TRƯỜNG CAO ĐẲNG NGHỀ CÔNG NGHIỆP HÀ NỘI
Tác giả
Vũ Thị Kim Phượng
Bùi Quang Ngọc

GIÁO TRÌNH
LẬP TRÌNH CƠ SỞ DỮ LIỆU
(Lưu hành nội bộ)

Hà Nội năm 2011

Chương 1
TỔNG QUAN VỀ LẬP TRÌNH VỚI NGÔN NGỮ VISUAL BASIC .NET

MỤC TIÊU CỦA CHƯƠNG

- Về kiến thức

Giới thiệu các khái niệm cơ bản và cấu trúc của .Net Framework, đồng thời thông qua việc trình bày một ứng dụng để giới thiệu về Visual Studio .Net.

- Về thái độ:

Giúp sinh viên chú ý và tiếp nhận ngôn ngữ lập trình Visual Basic.NET như một công cụ lập trình hiện đại và được ứng dụng nhiều trong thực tế.

- Về kỹ năng

Sau khi kết thúc bài học sinh viên có thể cài đặt phần mềm Visual Studio 2005 và chạy được các chương trình 1.4.2; 1.4.3; 1.4.4.

NỘI DUNG BÀI GIẢNG LÝ THUYẾT

1.1 Visual Studio.NET và .NET Framework

1.1.1 Giới thiệu

Trong thời đại công nghệ thông tin, dữ liệu trở nên quan trọng đến nỗi người ta mong muốn tất cả mọi thứ như điện thoại di động, máy tính xách tay, các máy PDA (Personal Digital Assistant) đều phải kết nối với nhau để chia sẽ dữ liệu và việc sử dụng các phần mềm để quản lý, sử dụng những dữ liệu đó là "không biên giới". Ứng dụng phải sẵn sàng để sử dụng từ trên máy tính cũng như trên điện thoại di động 24/24 giờ, ít lỗi, xử lý nhanh và bảo mật chặt chẽ.

Các yêu cầu này làm đau đầu những chuyên gia phát triển ứng dụng khi phần mềm chủ yếu viết cho hệ thống này không chạy trên một hệ thống khác bởi nhiều lý do như khác biệt về hệ điều hành, khác biệt về chuẩn giao tiếp dữ liệu, mạng. Thời gian và chi phí càng trở nên quý báu vì bạn không phải là người duy nhất biết lập trình. Làm sao sử dụng lại những ứng dụng đã viết để mở rộng thêm nhưng vẫn tương thích với những kỹ thuật mới?

Sun Microsystems đi đầu trong việc cung cấp giải pháp với Java. Java chạy ổn định trên các hệ điều hành Unix hay Solaris của Sun từ máy chủ tới các thiết bị cầm tay hay thậm chí trên các hệ điều hành Windows của Microsoft (một ví dụ rõ ràng đó là hầu hết các điện thoại di động thế hệ mới đều có phần mềm viết bằng Java). Kiến trúc lập trình dựa trên Java bytecode và thi hành trên máy ảo Java (JVM – Java Virtual Marchine) cho phép các ứng dụng Java chạy trên bất cứ hệ điều hành nào. Mô hình lập trình thuần hướng đối tượng của Java giúp các lập trình viên tùy ý sử dụng lại và mở rộng các đối tượng có sẵn. Các nhà cung cấp công cụ lập trình

dựa vào đây để gắn vào các môi trường phát triển ứng dụng bằng Java của mình đủ các thư viện lập trình nhằm hỗ trợ các lập trình viên.

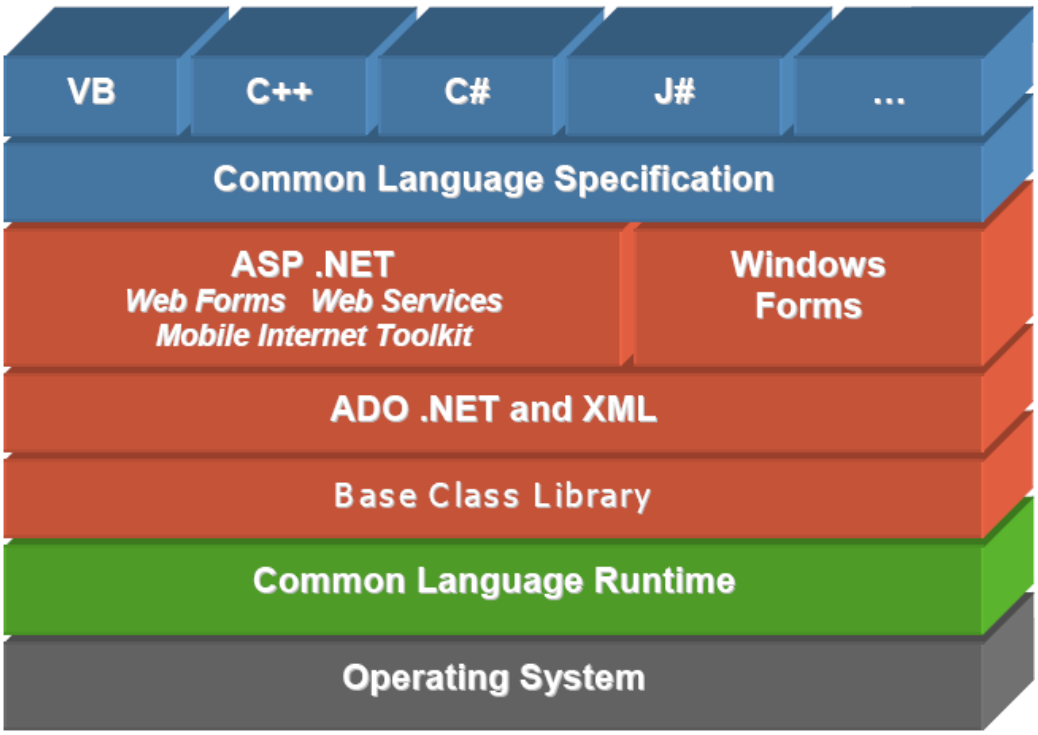
Sức mạnh của Java dường như quá lớn đến nỗi Microsoft từng phải chống trả bằng cách loại bỏ Java Virtual Marchine khỏi các phiên bản hệ điều hành Windows mới của mình như Windows XP. Tuy nhiên, Microsoft thừa hiểu rằng dù không cung cấp JVM, Sun cũng có thể tự cung cấp các JVM package cho những người dùng Windows. Đó là lý do tại sao nhà khổng lồ quyết định bắt tay xây dựng lại từ đầu một nền tảng phát triển ứng dụng mới: Microsoft.NET Framework.

Hiểu một cách đơn giản thì .NET Framework là bộ thư viện dành cho các lập trình viên .NET. **Framework** chính là một tập hợp hay thư viện các lớp đối tượng hỗ trợ người lập trình khi xây dựng ứng dụng. Bộ thư viện của .NET Framework bao gồm hơn 5000 lớp đối tượng đủ sức hỗ trợ hầu hết các yêu cầu từ phía lập trình viên. Công nghệ mã nguồn mở được đưa vào .NET và trong .NET, mọi thành phần đều có thể kế thừa và mở rộng.

Ngày 13/02/2002, Microsoft chính thức giới thiệu bộ công cụ lập trình mới của mình – Visual Studio.NET dựa trên công nghệ Microsoft .NET. Đó là một môi trường phát triển ứng dụng sử dụng giao diện đồ hoạ, tích hợp nhiều chức năng, tiện ích khác nhau để hỗ trợ tối đa cho các lập trình viên.

.NET Framework là thành phần quan trọng nhất trong kỹ thuật phát triển ứng dụng dựa trên .NET. Visual Studio sẽ giúp người lập trình nắm bắt và tận dụng tốt hơn những chức năng của .NET Framework.

1.1.2 Cấu trúc .NET Framework



Common language Specification:

Vai trò của thành phần này là đảm bảo sự tương tác giữa các đối tượng bất chấp chúng được xây dựng trong ngôn ngữ nào, miễn là chúng cung cấp được những thành phần chung của các ngôn ngữ muốn tương tác

ASP.NET

Bộ thư viện các lớp đối tượng dùng trong việc xây dựng các ứng dụng Web. Ứng dụng web xây dựng bằng ASP.NET tận dụng được toàn bộ khả năng của .NET Framework. Bên cạnh đó là một "phong cách" lập trình mới mà Microsoft đặt cho nó một tên gọi rất kêu : code behind. Đây là cách mà lập trình viên xây dựng các ứng dụng Windows based thường sử dụng – giao diện và lệnh được tách riêng.

Web services có thể hiểu khá sát nghĩa là các dịch vụ được cung cấp qua Web (hay Internet). Dịch vụ được coi là Web service không nhằm vào người dùng mà nhằm vào người xây dựng phần mềm. Web service có thể dùng để cung cấp các dữ liệu hay một chức năng tính toán.

Ví dụ, công ty du lịch của bạn đang sử dụng một hệ thống phần mềm để ghi nhận thông tin về khách du lịch đăng ký đi các tour. Để thực hiện việc đặt phòng khách sạn tại địa điểm du lịch, công ty cần biết thông tin về phòng trống tại các khách sạn. Khách sạn có thể cung cấp một Web service để cho biết thông tin về các phòng trống tại một thời điểm. Dựa vào đó, phần mềm của bạn sẽ biết rằng liệu có đủ chỗ để đặt phòng cho khách du lịch không? Nếu đủ, phần mềm lại có thể dùng một Web service khác cung cấp chức năng đặt phòng để thuê khách sạn. Điểm lợi của Web service ở đây là bạn không cần một người làm việc liên lạc với khách sạn để hỏi thông tin phòng, sau đó, với đủ các thông tin về nhiều loại phòng người đó sẽ xác định loại phòng nào cần đặt, số lượng đặt bao nhiêu, đủ hay không đủ rồi lại liên lạc lại với khách sạn để đặt phòng. Đừng quên là khách sạn lúc này cũng cần có người để làm việc với nhân viên của bạn và chưa chắc họ có thể liên lạc thành công.

Web service được cung cấp dựa vào ASP.NET và sự hỗ trợ từ phía hệ điều hành của Internet Information Server.

Windows Form

Bộ thư viện về Window form gồm các lớp đối tượng dành cho việc xây dựng các ứng dụng Windows based.

ADO.NET and XML

Bộ thư viện này gồm các lớp dùng để xử lý dữ liệu. ADO.NET thay thế ADO để trong việc thao tác với các dữ liệu thông thường. Các lớp đối tượng XML được cung cấp để bạn xử lý các dữ liệu theo định dạng mới: XML. Các ví dụ cho bộ thư viện này là SqlDataAdapter, SqlCommand, DataSet, XMLReader, XMLWriter,…

Base Class Library

Đây là thư viện các lớp cơ bản nhất, được dùng trong khi lập trình hay bản thân những người xây dựng .NET Framework cũng phải dùng nó để xây dựng các lớp cao hơn. Ví dụ các lớp trong thư viện này là String, Integer, Exception,…

Common Language Runtime

Là thành phần "kết nối" giữa các phần khác trong .NET Framework với hệ điều hành. Common Language Runtime (CLR) giữ vai trò quản lý việc thi hành các ứng dụng viết bằng .NET trên Windows.

CLR sẽ thông dịch các lời gọi từ chương trình cho Windows thi hành, đảm bảo ứng dụng không chiếm dụng và sử dụng tràn lan tài nguyên của hệ thống. Nó cũng không cho phép các lệnh "nguy hiểm" được thi hành. Các chức năng này được thực thi bởi các thành phần bên trong CLR như Class loader, Just In Time compiler, Garbage collector, Exception handler, COM marshaller, Security engine,…

Trong các phiên bản hệ điều hành Windows mới như XP.NET và Windows 2003, CLR được gắn kèm với hệ điều hành. Điều này đảm bảo ứng dụng viết ra trên máy tính của chúng ta sẽ chạy trên máy tính khác mà không cần cài đặt, các bước thực hiện chỉ đơn giản là một lệnh copy của DOS!

Operating System

.NET Framework cần được cài đặt và sử dụng trên một hệ điều hành. Hiện tại, .NET Framework chỉ có khả năng làm việc trên các hệ điều hành Microsoft Win32 và Win64 mà thôi. Trong thời gian tới, Microsoft sẽ đưa hệ thống này lên Windows CE cho các thiết bị cầm tay và có thể mở rộng cho các hệ điều hành khác như Unix.

Với vai trò quản lý việc xây dựng và thi hành ứng dụng, .NET Framework cung cấp các lớp đối tượng (Class) để NLT có thể gọi thi hành các chức năng mà đối tượng đó cung cấp. Tuy nhiên, lời gọi này có được "hưởng ứng" hay không còn tùy thuộc vào khả năng của hệ điều hành đang chạy ứng dụng.

Các chức năng đơn giản như hiển thị một hộp thông báo (Messagebox) sẽ được .NET Framework sử dụng các hàm API của Windows. Chức năng phức tạp hơn như sử dụng các COMponent sẽ yêu cầu Windows phải cài đặt Microsoft Transaction Server (MTS) hay các chức năng trên Web cần Windows phải cài đặt Internet Information Server (IIS).

Như vậy, việc lựa chọn một hệ điều hành để cài đặt và sử dụng .NET Framework cũng không kém phần quan trọng. Cài đặt .NET Framework trên các hệ điều hành Windows 2000, 2000 Server, XP, XP.NET, 2003 Server sẽ đơn giản và tiện dụng hơn trong khi lập trình.

1.1.3 Một số đặc trưng của Visual Studio .NET

Điểm đặc trưng của Microsoft Visual Studio là tất cả các ngôn ngữ lập trình trong .NET Framework đều có chung một IDE (Integrated Development Environment), trình gỡ lỗi, trình duyệt project và solution, class view, cửa sổ thuộc tính, hộp công cụ, menu và toolbar.

Ngoài ra còn phải kể đến một số các đặc trưng sau

1. Từ khóa và cú pháp lệnh được tô sáng.

2. Tự động hoàn thành các cú pháp lệnh khi người lập trình đánh dấu chấm với objects, namespace, enum và khi sử dụng từ khóa New.

3. Trình duyệt project, solution cho phép quản lý các ứng dụng chứa nhiều file với khuôn dạng khác nhau.

4. Cho phép người sử dụng xây dựng giao diện chỉ với thao tác kéo và thả trên form.

5. Cửa sổ thuộc tính cho phép thiết lập các giá trị cho các thuộc tính khác nhau của các điều khiển trên form hoặc trên trang web.

6. Trình gỡ rối cho phép gỡ lỗi của chương trình bằng cách thiết lập các điểm break point khi theo dõi quá trình hoạt động của chương trình.

7. Trình biên dịch trực tiếp (Hot compiler) cho phép kiểm tra cú pháp của dòng mã lệnh và thông báo các lỗi được phát hiện ngay khi người sử dụng nhập dòng lệnh vào từ bàn phím.

8. Người sử dụng được trợ giúp bằng trình Dynamic Help sử dụng MSDN (Microsoft Development Network library).

9. Biên dịch và xây dựng các ứng dụngCompilating and building applications.

10. Cho phép thi hành ứng dụng có/ không bộ gỡ rối (debugger).

11. Triển khai ứng dụng .NET của người sử dụng trên Internet hoặc CD.

Project và Solutions

Một Project là sự kết hợp của các file thực thi chương trình và file thư viện để tạo nên một ứng dụng hoặc một mô đun. Thông tin về project thường được lưu trữ trong các file có phần mở rộng là .vbproj (VB.NET) hoặc csproj (C#). Có thể kể đến một số loại project trong Visual Studio .NET như Console, Windows Application, ASP.NET, Class Librraries,...

Trái lại, một solution thường là sự kết hợp nhiều dự án khác nhau để tạo thành một số ứng dụng. Ví dụ trong một solution có thể bao gồm một project là ASP.NET WEB Application và một Windows Form project. Thông tin của một solution được lưu trữ trong các file .sln và được quản lý nhờ Visual Studio.NET Solution Explorer.

Toolbox, Properties và Class View Tabs

Tất cả các ngôn ngữ lập trình trong bộ Visual Studio .NET đều dùng chung một bộ công cụ (toolbox). Bộ công cụ này (thường xuất hiện bên tay trái màn hình) chứa một số các điều khiển chung cho các ứng dụng windows, web và dữ liệu như texbox, checkbox, tree view, list box, menus, file open dialog

Properties Tab - cửa sổ thuộc tính (thường xuất hiện bên tay phải) cho phép thiết lập các thuộc tính cho form và điều khiển tại thời điểm thiết kế mà không cần viết code.

1.2 Cài đặt phần mềm Visual Studio 2005

Microsoft Visual Studio 2005 là một bộ sản phẩm cung cấp 3 ngôn ngữ lập trình Visual Basic.NET (VB.NET), C# (C Sharp), Visual C++.NET và Visual J#.NET. Thêm vào đó là Integrated Development Environment (**IDE**) giúp lập trình dễ đàng, thoải mái. IDE không những cung cấp mọi công cu lập trình cần thiết mà còn giúp kiểm tra nguồn mã hay tạo giao diện Windows trực quan, truy tìm các tập tin liên hệ đến dự án và nhiều thứ khác nữa.

1.2.1 Yêu cầu phần cứng

System Requirements for Installing Visual Studio 2005	
Processor	Minimum:600 MHz Pentium processor Recommended:1 GHz Pentium processor
Operating System	• Microsoft Windows 2000 Professional SP4 •Microsoft Windows XP Professional SP2
RAM	Minimum: 192 MB Recommended:256 MB
Hard Disk	Without MSDN: 2 GB of available space required on installation drive. With MSDN: 3.8 GB of available space required on installation drive.

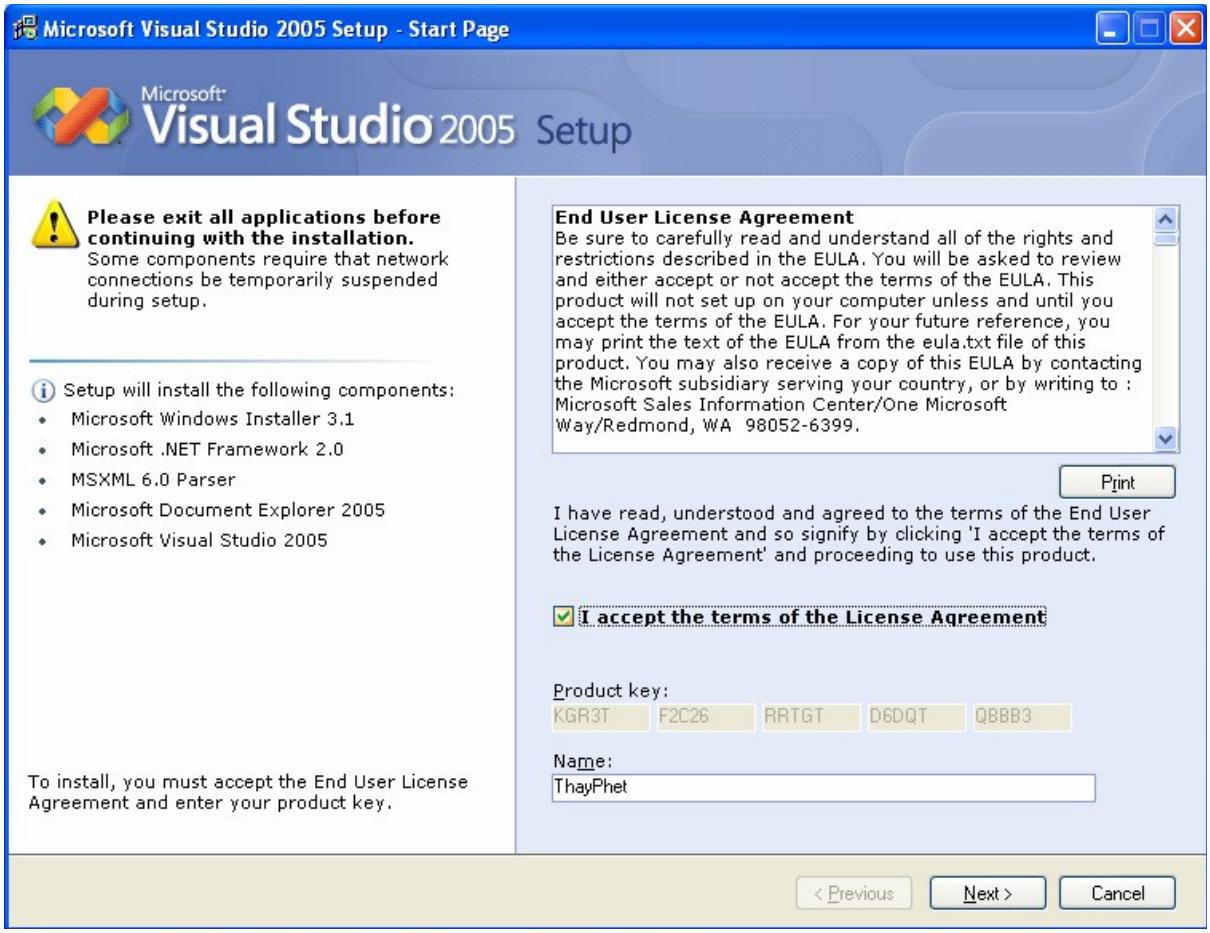
1.2.2 Cài đặt:

Microsoft Visual Studio 2005 thường gồm 4 đĩa CD. Trước khi cài đặt bạn phải chạy các file trên 3 đĩa để giải nén vào một thư mục trên đĩa cứng. Sau khi giải nén xong, xong sẽ có hai thư mục chính: thư mục VS (chứa các file cài Visual Studio .Net 1.17GB) và thư mục MSDN (chứa các thư viện hỗ trợ việc sử dụng Visual Studio.NET 1.55GB).

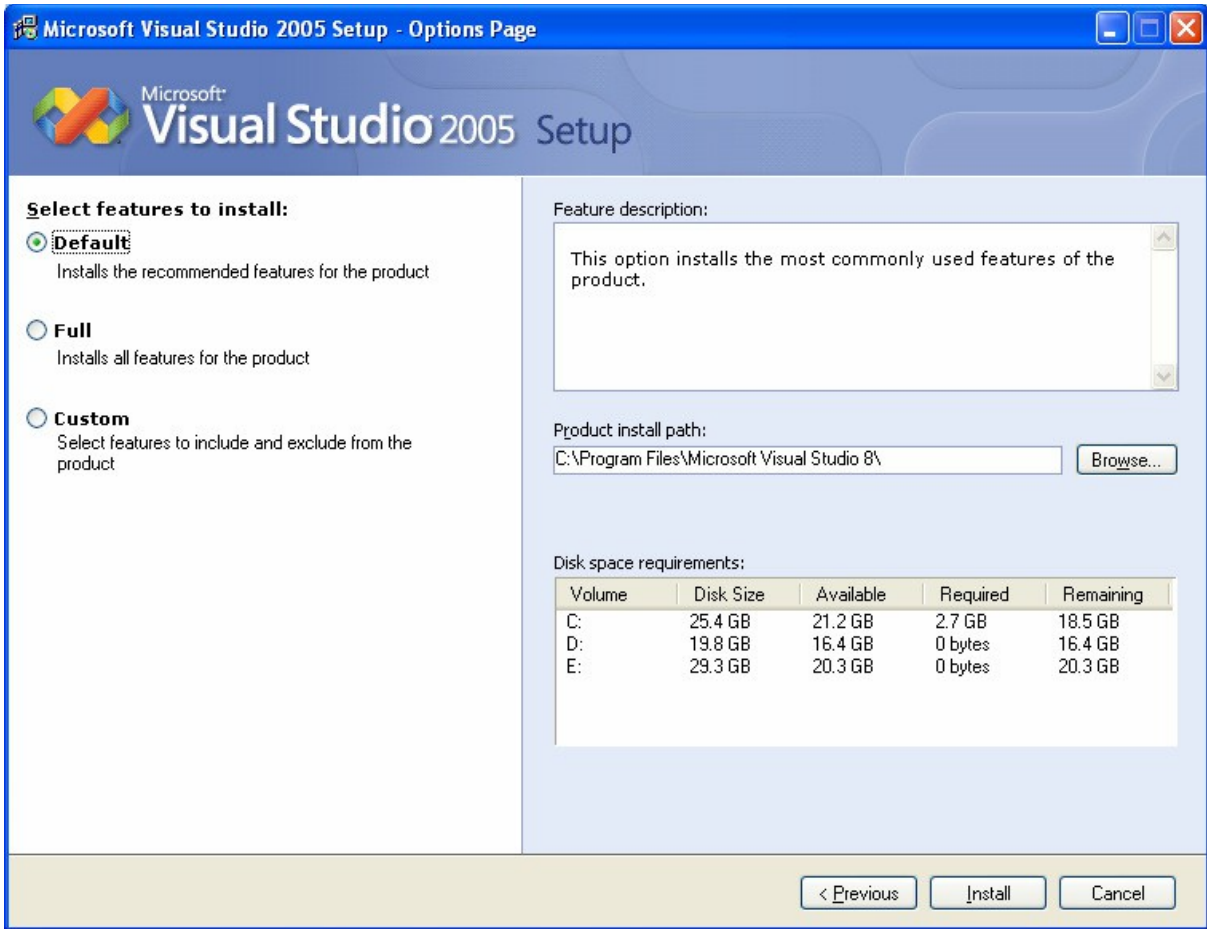
Để cài đặt, ta chạy file Setup.exe trong thư mục VS sau đó chọn chức năng Install Visual Studio 2005.

- Hộp thoại dưới đây sẽ xuất hiện trên màn hình:

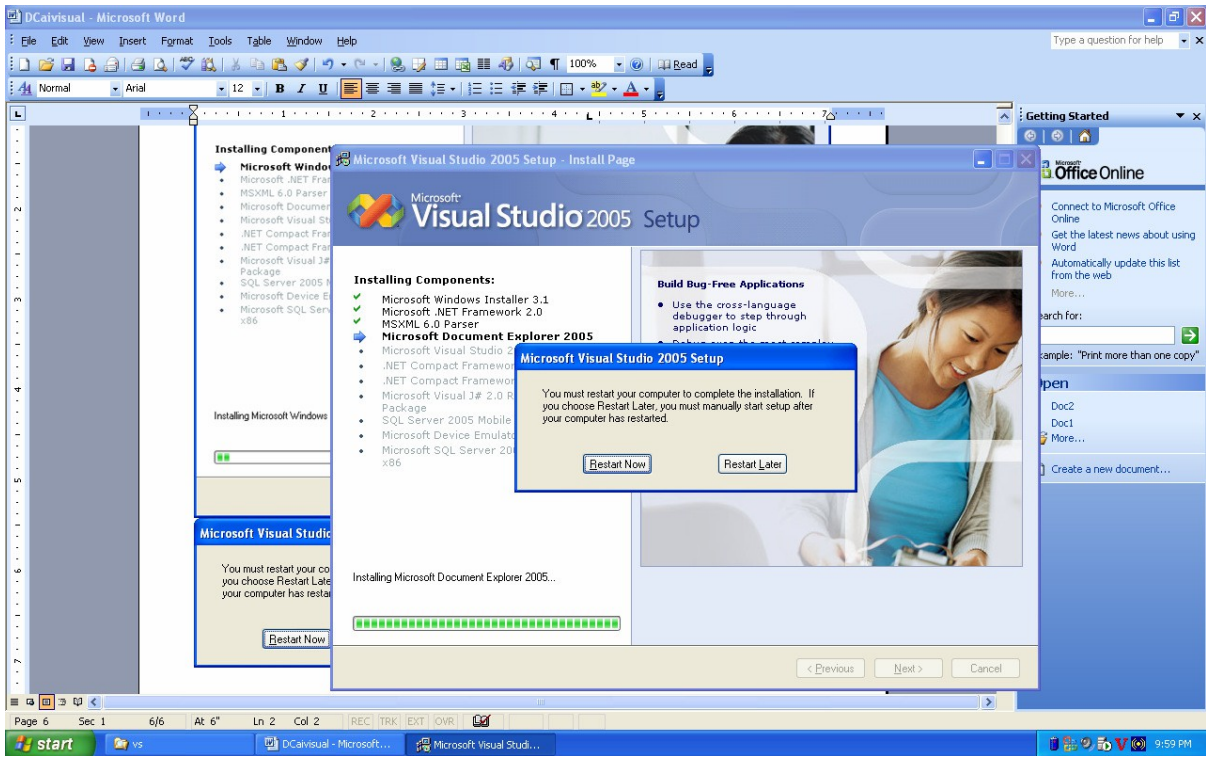
- Click chọn “Next”. Xuất hiện cửa sổ yêu cầu đăng ký



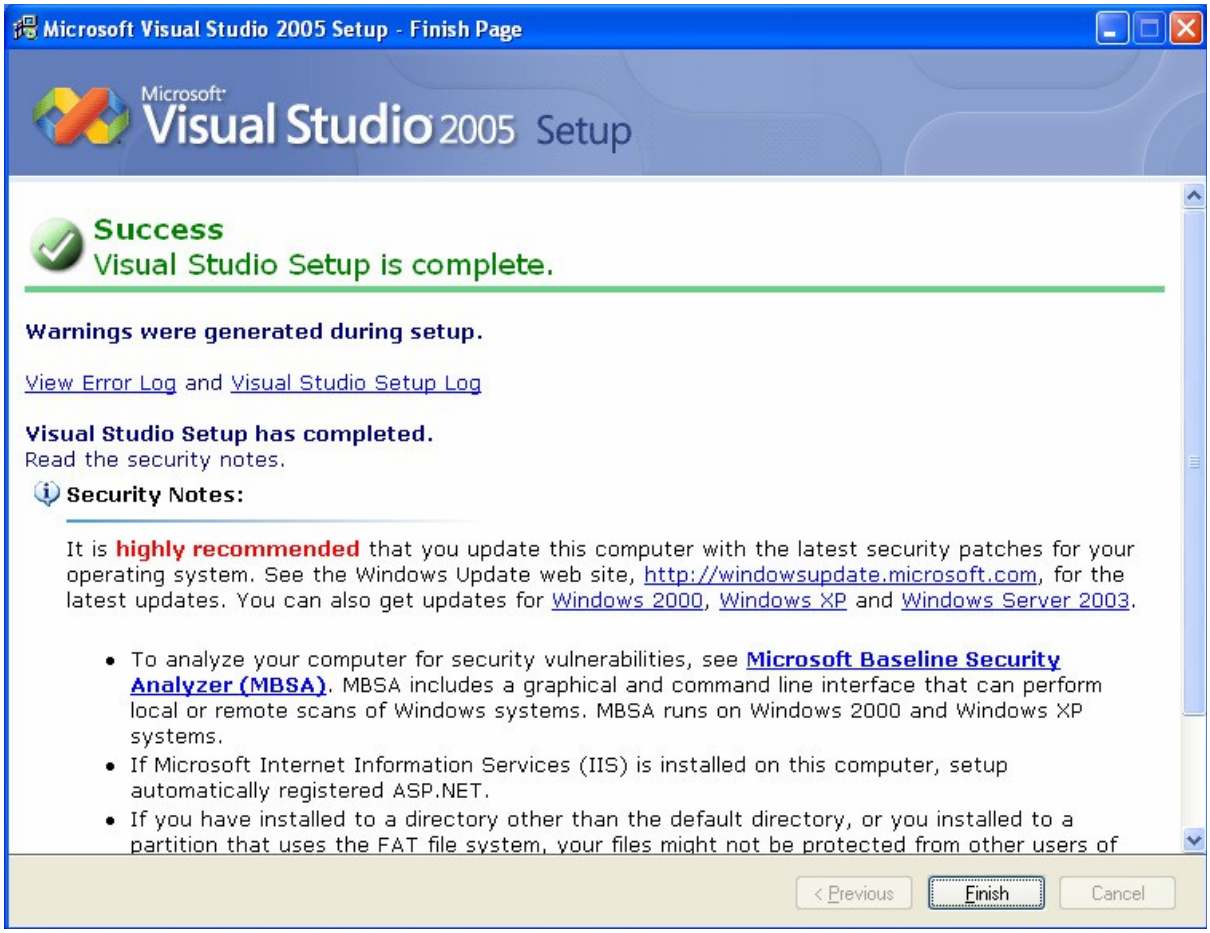
- Đăng ký bằng cách lựa chọn click chuột vào “I accept the term of the License Agreement”. Sau đó click chọn “Next” để sang trang tiếp theo.



- Chọn kiểu cài đặt Default (ngầm định) hay Full (đầy đủ) hay Custom (tự thiết lập các thông số); sau đó click vào nút Install và làm theo các hướng dẫn trên màn hình. Quá trình cài đặt sẽ yêu cầu bạn khởi động lại máy trước khi kết thúc.



Tùy theo cấu hình từng máy mà thời gian cài đặt có thể mất từ 25-35 phút.



1.3 Visual Basic.NET

1.3.1 Các phương pháp lập trình trong VB.NET

Phương pháp lập trình hướng lệnh

Trong phương pháp này người ta xem chương trình là tập hợp các lệnh. Khi đó việc viết chương trình là xác định xem chương trình gồm những lệnh nào, thứ tự thực hiện của chúng ra sao.

Phương pháp lập trình hướng thủ tục

Trong phương pháp này chương trình được xem là một hệ thống các thủ tục (sub và function). Trong đó, mỗi thủ tục là một dãy các lệnh được sắp thứ tự. Khi đó, việc viết chương trình là xác định xem chương trình gồm các thủ tục nào, mối quan hệ giữa chúng ra sao?

Phương pháp lập trình hướng đơn thể

Trong phương pháp này chương trình được xem là một hệ thống các đơn thể, mỗi đơn thể là một hệ thống các thủ tục và hàm. Khi đó, việc viết chương trình

là xác định xem chương trình gồm những đơn thể nào? Đơn thể nào đã có sẵn, đơn thể nào phải đi mua, đơn thể nào phải tự viết.

Trong VB.NET đơn thể được xem là một trong các cấu trúc Module, Class, Structure.

Phương pháp lập trình hướng đối tượng

Trong phương pháp này người ta xem chương trình là một hệ thống các đối tượng, mỗi một đối tượng là sự bao bọc bên trong nó 2 thành phần:

- Dữ liệu: là các thông tin về chính đối tượng. Trong một số sách, thành phần này còn được gọi là thành phần thuộc tính, thông tin .

- Hành động: là các khả năng mà đối tượng có thể thực hiện. Thành phần này còn có các tên như sau: phương thức, hàm thành phần, hành vi.

Mỗi một đối tượng sẽ được cài đặt trong chương trình với dạng đơn thể chứa dữ liệu. Thêm vào đó tính chất kế thừa cho phép chúng ta xây dựng đối tượng mới dựa trên cơ sở đối tượng đã có.

1.3.2 Visual Basic.NET

Nhiều lập trình viên đã quen với ngôn ngữ lập trình Visual Basic do Microsoft phát triển dựa trên ngôn ngữ BASIC từ năm 1964. Từ khi ra đời đến nay, Visual Basic đã phát triển qua nhiều thế hệ và kết thúc ở phiên bản VB 6.0 với rất nhiều modules, công cụ hay ứng dụng được bổ sung vào và đặc biệt là phương pháp kết nối với cơ sở dữ liệu qua sự kết hợp của ADO (Active Data Object). Tuy nhiên một trong những nhược điểm của VB 6.0 là không cung ứng tất cả các đặc trưng của một ngôn ngữ lập trình hướng đối tượng (Object Oriented Language - OOL) như các ngôn ngữ C++ hay Java.

Thay vì cải thiện hay vá víu thêm thắt vào VB 6.0, Microsoft đã xóa bỏ tất cả để làm lại từ đầu các ngôn ngữ lập trình hướng đối tượng rất hùng mạnh. Đó là các ngôn ngữ lập trình Visual Basic .NET và C# (C Sharp).

Có thể nói Visual Basic.NET (VB.NET) là một ngôn ngữ lập trình hướng đối tượng do Microsoft thiết kế lại từ con số 0. VB.NET không kế thừa hay bổ sung, phát triển từ VB 6.0 mà nó là một ngôn ngữ lập trình hoàn toàn mới trên nền Microsoft .NET FrameWork. VB.NET hỗ trợ đầy đủ các đặc trưng của một ngôn ngữ hướng đối tượng như là trừu tượng, bao đóng, kế thừa, đa hình, đa luồng và cấu trúc xử lý các exception.

VB.NET là một ngôn ngữ lập trình cho phép người sử dụng thiết lập các ứng dụng theo 3 loại:

- Ứng dụng Console là các chương trình chạy trên hệ điều hành MS-DOS thông qua trình biên dịch Visual Studio 2005 Command Prompt.

- Ứng dụng Windows Form là các ứng dụng chạy trên hệ điều hành Windows với các biểu mẫu (form) và các điều khiển (button, texbox, label,...)

- Ứng dụng ASP.NET gồm WEB Form và WEB Services.

Chương 2
NGÔN NGỮ LẬP TRÌNH VISUAL BASIC .NET

MỤC TIÊU CỦA CHƯƠNG

- Về kiến thức

Giới thiệu các kiểu dữ liệu cơ bản và kiểu dữ liệu có cấu trúc, cách khai báo và sử dụng các biến, cách viết và sử dụng các lệnh trong VB.NET.

- Về thái độ:

Giúp sinh viên hệ thống và củng cố lại các khái niệm chính trong lập trình hướng lệnh như nhập xuất dữ liệu, hoạt động của các cấu trúc rẽ nhánh và cấu trúc lặp.

- Về kỹ năng

Sau khi kết thúc bài học sinh viên giải được các ví dụ trong phần bài tập bằng cách viết các đoạn chương trình và xây dựng các thủ tục, hàm bằng ngôn ngữ VB.NET.

NỘI DUNG BÀI GIẢNG LÝ THUYẾT

2.1 Các kiểu dữ liệu và đặc điểm

2.1.1 Các kiểu dữ liệu

Các kiểu dữ liệu của .Net được mô tả chi tiết trong một cấu trúc gọi là Common Type System (CTS). CTS định nghĩa các kiểu dữ liệu, cách thức sử dụng, cách thức được quản lý lúc thực thi và cùng với Common Language Specification đóng một vai trò quan trọng trong việc trao đổi giữa các ngôn ngữ lập trình trong .Net.

Common Type System có chức năng:

Thi tế phm tộn nết nả cho ph ép tương tác giữa các ngôn ngữ lập trình, bảo toàn giá trị của dữ liệu khi có sự trao đổi dữ liệu giữa các ngôn ngữ và bảo đảm việc thực hiện câu lệnh được tối ưu

Cung c pấm tộn ô hình hướng đối tượng cho các ngôn ngữ lập trình.

Đưa ra những quy tắc để các ngôn ngữ lập trình phải tuân thủ nhằm bảo đảm các thành phần viết trên các ngôn ngữ khác nhau có thể tương tác với nhau.

Các kiểu dữ liệu trong VB.NET được chia thành 2 loại chính đó là

1. Value type (data types, Structure and Enumeration)

2. Reference Type (objects, delegates)

Bảng 2-1 mô tả các loại dữ liệu thuộc kiểu value type.

Kiểu dữ	Kiểu dữ	Kích	Mô tả

liệu trong VB.NET	liệu tương ứng trong .NET	cỡ	
Boolean	Boolean	1	Biểu diễn giá trị lôgic True hoặc False
Char	Char	2	Biểu diễn một ký tự Unicode (giá trị từ 0 đến 65535 không dấu)
String	String (Class)		0 đến khoảng 2 tỷ ký tự Unicode
DateTime	DateTime	8	0:00:00 ngày 01 tháng Giêng 0001 đến 23:59:59 ngày 31 tháng Mười Hai 9999.
Số nguyên			
Byte	Byte	1	0 đến 255
Integer	Int32	2	Từ -2 147 483 648 đến 2 147 483 647
Long	Int64	4	Từ -9 223 372 036 854 775 808 đến 9 223 372 036 854 775 807
Short	Int16	8	-32 768 to 32 767
Số thực			
Decimal	Decimal	12	Biểu diễn giá trị từ 1.0×10^{-28} đến 7.9×10^{28} với độ chính xác 28-29 chữ số thập phân.
Double	Double	8	Biểu diễn giá trị từ $\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$ với độ chính xác 15-16 chữ số thập phân
Single	Single	4	Biểu diễn giá trị từ $\pm 1.5 \times 10^{-45}$ đến $\pm 3.4 \times 10^{38}$ với độ chính xác 7 chữ số thập phân

Bảng 2-1 Các kiểu dữ liệu cơ bản trong VB.NET

2.1.2 Đặc điểm của các kiểu dữ liệu

Các kiểu dữ liệu mặc nhiên phát sinh từ lớp SysTem.Object. Ngoài các phương thức kế thừa từ lớp SysTem.Oject, các biến kiểu dữ liệu còn có các phương thức và thuộc tính đặc thù.

Các phương thức chung kế thừa từ SysTem.Object bao gồm

Equals: H tổ trợ cệso s ánh giữa hai object.

Finalize: Th cựhi nệc ác thao tác xóa bỏ trước khi object được tự động xóa bỏ.

GetHashCode: Ph át sinh một số tương ứng với giá trị của object.

GetType: Trả về kiểu của object.

ToString: Trả ra chuỗi ký tự nội dung mô tả một thể hiện của lớp.

Dưới đây là các bảng liệt kê những phương thức và thuộc tính đặc thù của các kiểu dữ liệu. Do các phương thức có nhiều cách sử dụng khác nhau, nên trong các phần nói về phương thức chỉ mô tả công dụng

a. Kiểu String

Thuộc tính

Tên	Mô tả
Chars(i)	Trả về ký tự tại vị trí chỉ ra trong biến. Thuộc tính có tính chỉ đọc
Length	Trả về số ký tự trong biến.

Phương thức

Tên	Mô tả
Clone	Trả về một tham chiếu của biến.
Compare	Phương thức so sánh hai tham số kiểu String dựa vào thứ tự các ký tự theo ngôn ngữ qui định trong Regional Settings của từng ký tự và trả về: -1 khi chuỗi thứ nhất nhỏ hơn chuỗi thứ 2 0 khi chuỗi thứ nhất bằng chuỗi thứ 1 khi chuỗi thứ nhất lớn hơn chuỗi thứ 2 Ngoài ra có thể có tham số qui định có phân biệt chữ Hoa chữ thường, v.v…
CompareOrdinal	So sánh hai tham số kiểu String dựa theo bảng mã các ký tự của các tham số. Hàm trả về hiệu của mã tham số thứ nhất và mã tham số thứ hai.
Concat	Nối các tham số lại với nhau và trả về chuỗi nối.
Copy	Tạo một thể hiện mới kiểu String có giá trị như tham số chuỗi truyền vào.
CopyTo	Sao chép một số ký tự chỉ ra từ một vị trí trên biến vào một vị trí chỉ ra trên mảng ký tự với số lượng ký tự truyền vào.
EndsWith	Trả về True/False cho biết các ký tự cuối của biến có khớp với chuỗi chỉ ra không.
Format	Thay thế phần biểu thức định dạng trong chuỗi bằng các các giá trị tương ứng đã được định dạng theo biểu thức.
IndexOf	Trả về vị trí đầu tiên tìm thấy chuỗi hoặc ký tự truyền vào trên biến, ; có thể sử dụng thêm vị trí bắt đầu tìm, trả về vị trí lần tìm thấy thứ mấy.
IndexOfAny	Trả về vị trí tìm thấy đầu tiên trên biến bất kỳ ký tự nào trong mảng

Tên	Mô tả
	ký tự truyền vào; có thể sử dụng thêm vị trí bắt đầu tìm, trả về vị trí lần tìm thấy thứ mấy.
Insert	Chèn vào một giá trị String truyền vào tại vị trí chỉ định trên biến.
Join	Nối các phần tử của mảng String truyền vào thành một chuỗi duy nhất với dấu nối là chuỗi dấu ngăn cách chỉ ra (separator)
LastIndexOf	Trả về vị trí tìm thấy cuối cùng trên biến, chuỗi hoặc ký tự truyền vào; có thể sử dụng thêm vị trí bắt đầu tìm, trả về vị trí lần tìm thấy thứ mấy.
LastIndexOfAny	Trả về vị trí tìm thấy cuối cùng trên biến bất kỳ ký tự nào trong mảng ký tự truyền vào; có thể sử dụng thêm vị trí bắt đầu tìm, trả về vị trí lần tìm thấy thứ mấy.
PadLeft	Nối thêm bên trái ký tự truyền vào với số lần sao cho độ dài tổng cộng bằng độ dài chỉ ra. Nếu độ dài tổng cộng chỉ ra nhỏ hơn độ dài của biến, không ký tự nào được thêm vào.
PadRight	Nối thêm bên phải ký tự truyền vào với số lần sao cho độ dài tổng cộng bằng độ dài chỉ ra. Nếu độ dài tổng cộng chỉ ra nhỏ hơn độ dài của biến, không ký tự nào được thêm vào.
Remove	Xóa bỏ một số ký tự chỉ ra khỏi biến từ vị trí truyền vào.
Replace	Thay thế tất cả ký tự hay chuỗi tìm thấy trên biến bằng ký tự hay chuỗi truyền vào.
Split	Trả về một mảng String với các phần tử chứa các chuỗi con được ngắt ra từ biến tùy theo ký tự ngăn cách truyền vào.
StartsWith	Cho biết trị bắt đầu của biến có khớp với chuỗi truyền vào.
Substring	Trả về một chuỗi con từ biến.
ToLower	Trả về bản sao của biến với các ký tự in thường.
ToUpper	Trả về bản sao của biến với các ký tự in HOA.
Trim	Trả về biến đã loại bỏ tất cả các ký tự từ đầu đến cuối của biến khớp với mảng ký tự truyền vào.
TrimEnd	Trả về biến đã loại bỏ tất cả các ký tự từ vị trí cuối của biến khớp với mảng ký tự truyền vào.
TrimStart	Trả về biến đã loại bỏ tất cả các ký tự từ vị trí đầu của biến khớp với mảng ký tự truyền vào.

b. Kiểu DateTime

Thuộc tính

Tên	Mô tả
MaxValue	Hiển thị giá trị lớn nhất của kiểu DateTime (chỉ đọc).
MinValue	Hiển thị giá trị nhỏ nhất của kiểu DateTime (chỉ đọc).

Phương thức

Tên	Mô tả
Date	Trả về giá trị ngày tháng năm của biến.
Day	Trả về giá trị ngày trong tháng của biến.
DayOfWeek	Trả về giá trị ngày trong tuần của biến, với ngày đầu tiên là Chủ nhật có giá trị là 0.
DayOfYear	Trả về giá trị ngày trong năm của biến.
Hour	Trả về giá trị giờ của biến.
Millisecond	Trả về giá trị phần ngàn giây của biến.
Minute	Trả về giá trị phút của biến.
Month	Trả về tháng của biến.
Now	Trả về giá trị ngày giờ hiện hành của hệ thống.
Second	Trả về giá trị giây của biến.
TimeOfDay	Trả về giá trị giờ phút giây của biến.
Today	Trả về ngày hiện hành.
Year	Trả về năm của biến.
AddDays	Thêm số ngày truyền vào cho giá trị của biến.
AddHours	Thêm số giờ truyền vào cho giá trị của biến.
AddMilliseconds	Thêm số phần ngàn giây truyền vào cho giá trị của biến.
AddMinutes	Thêm số phút truyền vào cho giá trị của biến.
AddMonths	Thêm số tháng truyền vào cho giá trị của biến.
AddSeconds	Thêm số giây truyền vào cho giá trị của biến.
AddYears	Thêm số năm truyền vào cho giá trị của biến.
Compare	So sánh hai biến ngày giờ và cho biết biến nào lớn hơn.
CompareTo	So sánh biến với một tham số Object.
DaysInMonth	Cho biết số ngày trong tháng theo tham số tháng, năm truyền vào.
IsLeapYear	Cho biết giá trị năm truyền vào (dạng yyyy) có phải là năm nhuận hay không.
Subtract	Trừ một giá trị thời gian khỏi biến.
ToLongDateString	Chuyển giá trị biến ra định dạng Long Date.
ToLongTimeString	Chuyển giá trị biến ra định dạng Long Time.
ToShortDateString	Chuyển giá trị biến ra định dạng Short Date.
ToShortTimeString	Chuyển giá trị biến ra định dạng Short Time.
ToString	Trả về chuỗi trị của biến theo định dạng truyền vào

c. Kiểu Number

Thuộc tính

Tên	Mô tả

MaxValue	Hiển thị giá trị lớn nhất của kiểu (chỉ đọc).
MinValue	Hiển thị giá trị nhỏ nhất của kiểu (chỉ đọc).

Ngoại trừ kiểu String, các kiểu khác khi muốn chuyển sang kiểu chuỗi đều có thể dùng phương thức ToString (kế thừa từ lớp Object) để chuyển đổi và định dạng cùng lúc. Cú pháp sử dụng:

```
ToString()
ToString(<biểu thức định dạng>)
```

Dưới đây là bảng biểu thức định dạng

Biểu thức	**Ý nghĩa**	**Ví dụ**
c, C	Định dạng tiền tệ	12345.67 ToString("C") hiển thị $ 12,345.67
e, E	Định dạng số khoa học.	12345.67 ToString("E") hiển thị 1.234567E+0004
f, F	Định dạng cố định	12345.67 ToString("F") hiển thị 12345.67 (với 2 số lẻ)
g, G	Định dạng tổng quát	12345.67 ToString("G") hiển thị 12345.67 tùy theo gía trị có thể hiện thị dưới dạng E hoặc F
n, N	Định dạng số	12345.67 ToString("N") hiển thị 12,345.67
p, P	Định dạng phần trăm	0.45 ToString("P") hiển thị 45 %
x, X	Định dạng Thập lục phân	250 ToString("X") hiển thị FA

Bảng 2-2 Biểu thức định dạng

Ngoài ra chúng ta cũng có thể sử dụng các ký tự sau đây để lập biểu thức định dạng

Biểu thức	**Ý nghĩa**	**Ví dụ**
0	Số không giữ chỗ	123 ToString("0000") hiển thị 0123
#	Số bất kỳ giữ chỗ	123 ToString("####") hiển thị 123
.	Dấu phần lẻ	123 ToString("####.00") hiển thị 123.00
,	Dấu chia cụm ba số	12345 ToString("#,###") hiển thị 12,345
%	Dấu phần trăm	0.45 ToString("# %") hiển thị 45 %
E+0,E-0,e+0, e-0	Dấu hiển thị số khoa học	12345678 ToString ("#.#######E+000") hiển thị 1.2345678E+007
\	Ký tự literal	123456 ToString("\# #,###") hiển thị # 123,456
; Ký tự ngăn cách	Ký tự ngăn cách vùng	Với ToString("dương #,###;âm #,###; số không")

Biểu thức	Ý nghĩa	Ví dụ
vùng		-123456 hiển thị âm 123,456 0 hiển thị số không

Bảng 2-3 Ký tự định dạng biểu thức

2.1.3 Biểu diễn các giá trị

Bảng dưới đây mô tả cách biểu diễn các giá trị cụ thể theo các kiểu dữ liệu trong VB.NET

Kiểu dữ liệu	Biểu diễn	Ví dụ
Boolean	True, False	Dim bFlag As Boolean = False
Char	C	Dim chVal As Char = "X"C
Date	# #	Dim datMillen As Date = #01/01/2001#
Decimal	D	Dim decValue As Decimal = 6.14D
Double	Bất kỳ một số viết kiểu dấu phẩy động nào hoặc R	Dim dblValue As Double = 6.142 Dim dblValue As Double = 6.142R
Integer	Bất kỳ một số nguyên nào thuộc khoảng (-2,147,483,648 đến 2,147,483,647), hoặc I	Dim iValue As Integer = 362 Dim iValue As Integer = 362I Dim iValue As Integer = &H16AI (hệ 16) Dim iValue As Integer = &O552I (hệ 8)
Long	Bất kỳ một số nguyên nào ngoài khoảng (-9,223,372,036,854,775,808 đến -2,147,483,649 hoặc ngoài khoảng từ 2,147,483,648 đến 9,223,372,036,854,775,807), hoặc L	Dim lValue As Long = 362L Dim lValue As Long = &H16AL (hệ 16) Dim lValue As Long = &O552L (hệ 8)
Short	S	Dim shValue As Short = 362S Dim shValue As Short = &H16AS (hệ 16) Dim shValue As Short = &O552S (hệ 8)
Single	F	Dim sngValue As Single = 6.142F
String	" "	Dim strValue As String = "This is a string"

Bảng 2-4 Cách viết các giá trị theo từng kiểu dữ liệu

2.2 Biến, hằng

2.2.1 Biến và tính chất

Biến là một thực thể với 6 tính chất sau:

Name: Tên của biến. Tên của biến phải là định danh hợp lệ trong **VB.Net**, nghĩa là phải bắt đầu bằng một chữ cái hoặc ký tự _ và không được trùng với các từ khóa của**VB.Net**. Trường hợp muốn dùng từ khóa làm tên biến phải được dùng trong ngoặc vuông như **[String]**, **[Boolean]**, … Tên biến nên có ý nghĩa gợi nhớ đến nội dung trong nó như *Don_gia*, *So_luong_xuat*.

Address: Địa chỉ vùng nhớ nơi lưu giữ giá trị của biến. Trong thời gian sống của chương trình, địa chỉ của biến có thể thay đổi.

Type: Kiểu của biến, còn gọi là kiểu dữ liệu. Mỗi biến phải thuộc về một kiểu dữ liệu trong **Common Type System**.

Value: Giá trị. Giá trị của biến phải phù hợp với kiểu dữ liệu của biến.

Scope: Phạm vi sử dụng của biến.

Mỗi biến có một phạm vi sử dụng là phạm vi trong chương trình nơi biến được nhìn nhận đối với câu lệnh. Có các loại phạm vi sau:

Cấp độ	**Mô tả**
Block Scope - Phạm vi khối lệnh	Chỉ được nhìn nhận trong khối lệnh mà biến được khai báo
Procedure Scope - Phạm vi thủ tục	Cho phép truy cập tại bất kỳ một dòng lệnh nào bên trong thủ tục mà biến được khai báo
Module Scope - Phạm vi Module	Cho phép truy cập tại bất kỳ một dòng code nào trong module, class hoặc structure nơi biến được khai báo
Namespace Scope - Phạm vi Namespace	Cho phép truy cập tại bất kỳ một dòng code nào của namespace nơi biến được khai báo

LifeTime: Thời gian tồn tại của biến.

Trong khi phạm vi sử dụng của biến xác định nơi chốn biến được phép sử dụng, thì thời gian tồn tại của biến xác định khoảng thời gian biến có thể lưu giữ giá trị.

Biến có phạm vi Module có thời gian tồn tại là thời gian ứng dụng đang thực hiện. Biến có phạm vi khối lệnh, thủ tục chỉ tồn tại trong khi thủ tục đang thực hiện. Biến này sẽ được khởi tạo theo giá trị mặc định của kiểu dữ liệu khi thủ tục bắt đầu thực hiện và chấm dứt khi thủ tục kết thúc.

2.2.2 Khai báo và khởi tạo giá trị cho biến

Khai báo

Trong VB.NET các biến được khai báo với cú pháp sau:

```
Dim tên_biến AS kiểu_dữ_liệu
```

Trong đó tên biến là một chuỗi được bắt đầu bởi một chữ cái, không chứa dấu cách và là duy nhất trong một phạm vi.

- Ví dụ: Khai báo biến `i` thuộc kiểu integer

```
Dim i as integer
```

- Để nhấn mạnh vai trò của hàm tạo (constructor), chúng ta có thể viết:

```
Dim x as Integer = New Integer()
```

- Khi khai báo nhiều biến trên cùng dòng và không chỉ ra kiểu của biến, biến sẽ lấy kiểu dữ liệu của biến khai báo dữ liệu tường minh tiếp sau đó

```
Dim x as Integer, a, b, c as Long
```

Các biến a, b, c đều cùng có kiểu Long

Khai báo và khởi tạo giá trị

- Có thể khai báo và khởi tạo giá trị cho biến cùng lúc:

```
Dim x as Integer = 100, y as Integer = 200
```

Trong cách này, phải khai báo tường minh kiểu dữ liệu cho từng biến.

Khai báo biến sử dụng các ký tự hậu tố

Có thể khai báo biến bằng cách thêm vào sau tên biến một ký tự (hậu tố) xác định kiểu dữ liệu của biến.

Ví dụ: `Dim x%` tương ứng với lệnh khai báo `Dim x As Integer`

Các ký tự hậu tố được chỉ ra trong bảng dưới đây:

Kiểu dữ liệu	**Ký tự**	**Ví dụ**
Decimal	@	`Dim decValue@ = 132.24`
Double	#	`Dim dblValue# = .0000001327`
Integer	%	`Dim iCount% = 100`
Long	&	`Dim lLimit& = 1000000`
Single	!	`Dim sngValue! = 3.1417`
String	$	`Dim strInput$ = ""`

Bảng 2-5 Các ký tự hậu tố xác định kiểu dữ liệu

2.2.3 Xác định phạm vi biến (tầm vực của biến)

Phạm vi khối lệnh

Trong VB.NET khối lệnh là tập hợp các câu lệnh thuộc về một trong các cấu trúc điều khiển sau:

- `Do .. Loop`
- `For [Each] .. Next`
- `If .. End If`
- `Select .. End Select`
- `SyncLock .. End SyncLock`
- `Try .. End Try`
- `While .. End While`
- `With .. End With`

Một biến được khai báo trong khối lệnh sẽ có tầm vực khối lệnh nghĩa là nó chỉ được phép sử dụng trong khối lệnh này.

Phạm vi thủ tục

Có hai loại thủ tục trong VB.NET đó là sub và function. Các biến được khai báo trong thủ tục và nằm ngoài bất kỳ khối lệnh nào của thủ tục là các biến có phạm vi thủ tục. Chúng còn được gọi là các biến cục bộ.

Phạm vi Module (phạm vi đơn thể)

Biến được khai báo trong đơn thể và không nằm trong bất kỳ khối lệnh hay thủ tục nào của đơn thể là các biến có phạm vi đơn thể.

Các biến có phạm vi đơn thể được khai báo bằng cách thay từ khóa `Dim` bằng từ khóa `Private`.

Phạm vi Namespace

Các biến có phạm vi đơn thể và được khai báo với từ khóa Public hoặc Friend được gọi là các biến có phạm vi Namespace.

2.2.4 Hằng và số ngẫu nhiên

Hằng

Hằng là đại lượng có giá trị không đổi trong một phạm vi nào đó. Trong VB.NET các hằng số được khai báo với từ khóa Const và phải được khởi tạo giá trị ngay khi khai báo.

Cú pháp khai báo hằng:

```
Const <Tên_hằng> As <Kiểu_dữ_liệu>=<Biểu_thức>
```

Ví dụ:

```
Const Max as Interger=100
```

Số ngẫu nhiên

- Sử dụng đối tượng Random để khai báo số ngẫu nhiên

- Dùng phương thức Next của đối tượng Random để khai báo khoảng mà số ngẫu nhiên lấy ra

Ví dụ:

```
Dim rd As Random
Set rd =New Random
Dim rdSo As Integer
rdSo = rd.Next (1,1000)
```

2.2.5 Chuyển đổi các kiểu dữ liệu

Option Strict ON|OFF

Khai báo Option Strict On|Off là một chỉ thị không cho phép các chuyển đổi kiểu làm mất dữ liệu. Nhưng chúng ta có thể thực hiện các chuyển đổi mở rộng như chuyển biến kiểu Integer sang kiểu Long.

Ví dụ: Đoạn code sau không có lỗi vì kiểu integer có thể chuyển sang kiểu Long là một kiểu chuyển đổi mở rộng được gọi là *implicit conversion*.

```
Dim a As Integer = 5
Dim b As Long = a
```

Song khi đảo ngược lại kiểu dữ liệu của hai biến a và b như sau

```
Dim a As Long = 5
Dim b As Integer = a
```

Thì sẽ gây ra lỗi trường hợp Option Strict là ON.

Cũng như vậy, khi khai báo này bật (On) sẽ không cho phép tự động chuyển đổi kiểu chuỗi sang kiểu số hay ngược lại. Thay vào đó ta phải sử dụng các hàm convert.

Ví dụ khi Option Strict là ON, với hai biến x và y được khai báo như sau

```
Dim x as string, y as integer
```

Thì câu lệnh `x=y` sẽ gây ra lỗi cú pháp. Thay vào đó ta phải dùng lệnh `x=CStr(y)` hoặc `x=y.ToString`

Các hàm chuyển đổi dữ liệu

Ta có thể sử dụng các hàm trong bảng dưới đây để chuyển đổi giữa các kiểu dữ liệu. Kiểu chuyển đổi này được gọi là *explicit conversion* vì trong kiểu chuyển đổi này lập trình viên đã chỉ định rõ cần chuyển về kiểu dữ liệu nào.

Hàm chuyển đổi	**Chuyển đổi số về kiểu**
CBool	Boolean

Hàm chuyển đổi	Chuyển đổi số về kiểu
CByte	Byte
CChar	Char
CDate	Date
CDbl	Double
CDec	Decimal
CInt	Integer
CLng	Long
CObj	Object
CSng	Single
CStr	String

Bảng 2-6 Các hàm chuyển đổi dữ liệu

2.3 Toán tử

Toán tử là ký hiệu chỉ ra phép toán nào được thực hiện trên các toán hạng (có thể là một hoặc hai toán hạng)

2.3.1 Toán tử toán học

Ký hiệu	Mô tả
+	(cộng)
-	(trừ)
*	(nhân)
/	(chia)
\	(chia lấy phần nguyên)
Mod	chia lấy phần dư của số nguyên
^	(lũy thừa)

2.3.2 Toán tử nối chuỗi

Toán tử chỉ dành cho toán hạng kiểu String với hai toán tử là & (ampersand) và + (cộng). Kết quả là một trị String gồm các ký tự của toán hạng thứ nhất tiếp theo sau là các ký tự của toán hạng thứ hai.

2.3.3 Toán tử gán

Ký hiệu	Mô tả
=	Gán toán hạng thứ hai cho toán hạng thứ nhất
+=	Cộng hoặc nối chuỗi toán hạng sau vào toán hạng đầu và gán

	kết quả cho toán hạng đầu
-=	Trừ toán hạng sau khỏi toán hạng đầu và gán hiệu cho toán hạng đầu
*=	Nhân hai toán hạng với nhau và gán tích cho toán hạng đầu
/=	Chia toán hạng đầu cho toán hạng sau và gán thương cho toán hạng đầu
\=	Thực hiện phép toán \ giữa toán hạng đầu và toán hạng sau và gán kết quả cho toán hạng đầu
^=	Tính lũy thừa toán hạng đầu với số mũ là toán hạng sau và gán kết quả cho toán hạng đầu
&=	Nối chuỗi toán hạng sau vào toán hạng đầu và gán kết quả cho toán hạng đầu

2.3.4 Toán tử so sánh

Ký hiệu	**Mô tả**
=	Bằng
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
>	Lớn hơn
<	Nhỏ hơn
<>	Khác
TypeOf … Is …	So sánh kiểu của biến kiểu tham chiếu thứ nhất có trùng kiểu trên toán hạng thứ hai, nếu trùng trả về True, ngược lại False
Is	Toán tử dành cho toán hạng kiểu tham chiếu, trả về True nếu hai toán hạng cùng tham chiếu đến một đối tượng, ngược lại là False)
Like	Toán tử dành cho toán hạng kiểu String, trả về True nếu toán hạng thứ nhất trùng với mẫu (pattern) của toán hạng thứ hai, ngược lại là False.

2.3.5 Toán tử luận lý và bitwise

Toán tử luận lý trả về giá trị True, False

Ký hiệu	**Mô tả**
Not	Trả về giá trị ngược lại của toán hạng
And	Trả về True (1) khi và chỉ khi hai toán hạng cùng là True (1)
AndAlso	Trả về giá trị như And nhưng khi toán hạng thứ nhất là False (0) sẽ không kiểm tra toán hạng thứ hai và trả về False
Or	Trả về False (0) khi và chỉ khi hai toán hạng cùng là False (0)

Ký hiệu	Mô tả
OrElse	Trả về giá trị như Or nhưng khi toán hạng thứ nhất là True (1) sẽ không kiểm tra toán hạng thứ hai và trả về True (1)
Xor	Trả về True (1) khi và chỉ khi có 1 toán hạng là True (1)
Not	Trả về giá trị ngược lại của toán hạng

2.4 Các lệnh VB.NET

2.4.1 Chú thích và cách viết lệnh

Thông thường, mỗi lệnh VB.NET được viết trên một dòng. Trình biên dịch sẽ coi ký hiệu xuống dòng cũng là ký hiệu kết thúc câu lệnh. Trong trường hợp muốn viết một câu lệnh trên nhiều dòng, trước khi xuống dòng cần bổ sung thêm ký tự gạch dưới.

Ví dụ: Câu lệnh sau sẽ sinh ra lỗi

```
Dim myName As String = "My name is
Faraz Rasheed"
```

Ta có thể sửa thành

```
Dim myName As String = "My name is" & _
"Faraz Rasheed"
```

Để viết thêm các chú thích trong chương trình, ta sử dụng dấu nháy đơn ở đầu dòng chú thích

Để viết nhiều lệnh trên cùng một dòng ta sử dụng dấu hai chấm ngăn cách các câu lệnh.

Ví dụ: Đoạn mã sau:

```
X=1
Y=2
```

Có thể viết liền một dòng:

```
X=1 : y=2
```

2.4.2 Nhập/xuất dữ liệu

a. Nhập/xuất dữ liệu với Console

- Xuất dữ liệu:

```
Console.Write(value)
```

- Nhập một ký tự

```
Console.Read()
```

- Nhập một dòng:

```
Console.Readline()
```

Ví dụ 2-1: Chương trình dưới đây sẽ nhập tên, điểm toán, điểm văn từ màn hình console sau đó hiển thị lại tên và điểm trung bình.

```
Imports System
Public Module Module1
  Sub Main()
     Dim hoten as string
     Dim toan,van, diemtb as decimal
     Console.Write("Nhap ho ten:")
     HoTen = Console.ReadLine()
     Console.Write("Nhap diem toan:")
     Toan = Console.ReadLine()
     Console.Write("Nhap diem van:")
     Van = Console.ReadLine()
     DiemTB = (Toan+Van)/2
     Console.WriteLine("Ho ten :" & HoTen)
     Console.WriteLine("Diem Trung Binh:" & DiemTB)
  End Sub
End Module
```

b. InputBox- Hộp nhập liệu

InputBox là một hàm cho phép hiển thị một hộp thoại có chứa một thông báo và một textbox, chờ người sử dụng nhập vào một chuỗi hoặc nhấn một nút lệnh và sau đó trả lại chuỗi đã nhập trong textbox.

Cú pháp:

```
InputBox( Prompt,[Title],[DefaultResponse], [Xpos], [Ypos])
```

Trong đó

- Prompt (tham số bắt buộc) là chuỗi thông báo hiển thị trong hộp thoại. Trường hợp thông báo trên nhiều dòng cần dùng biểu thức Chr(13)&Chr(10) để ngắt dòng.

- Title (tham số tùy chọn) là tiêu đề của hộp thoại. Nếu không chỉ định thì tên của ứng dụng sẽ được dùng làm tiêu đề.

- DefaultResponse (tham số tùy chọn) là chuỗi hiển thị trong textbox khi người sử dụng không nhập giá trị gì và nút ngầm định được đáp ứng. Nếu không chỉ định thì textbox bỏ trống

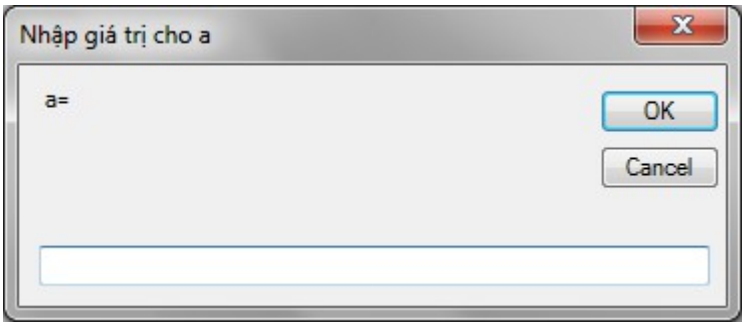
- Xpos (tham số tùy chọn) là một giá trị số xác định khoảng cách giữa mép trái của hộp thoại với mép trái màn hình. Nếu không chỉ định thì hộp thoại do inputbox tạo ra sẽ được căn giữa màn hình theo chiều ngang.

- Ypos (tham số tùy chọn) là một giá trị số xác định khoảng cách giữa mép trên của hộp thoại với đỉnh màn hình. Nếu không chỉ định thì hộp thoại do inputbox tạo ra sẽ được căn giữa màn hình theo chiều dọc.

Ví dụ: Để nhập giá trị cho biến a chúng ta có thể sử dụng hàm InputBox

```
a=InputBox("Nhập giá trị cho biến a")
```

Hộp thoại hiển thị trên màn hình như sau:



c. Hiển thị thông báo - Hàm Msgbox

Msgbox là một hàm hiển thị một thông báo trong một hộp thoại, chờ người sử dụng chọn một nút lệnh và trả về giá trị nguyên tương ứng với nút lệnh được chọn.

Cú pháp

```
MsgBox(Prompt[,Buttons][,Title)
```

Trong đó:

- Prompt (bắt buộc) là chuỗi được hiển thị trong hộp thông báo, độ dài tối đa 1024 ký tự; có thể dùng Chr(13) & Chr(10) để ngắt dòng.

- Buttons (tham số tùy chọn) là một biểu thức số tương ứng với tổng của tất cả các giá trị bao gồm số lượng và kiểu của các nút được hiển thị, biểu tượng trong hộp thông báo, nút bấm tùy chọn. Nếu tham số này được bỏ qua thì giá trị ngầm định là zero.

Các giá trị quy ước dạng của hộp thông báo được hiển thị trong bảng dưới đây:

Member	**Value**	**Description**
OKOnly	0	Chỉ hiển thị nút OK
OKCancel	1	Hiển thị nút OK và Cancel
AbortRetryIgnore	2	Hiển thị nút Abort, Retry và Ignore
YesNoCancel	3	Hiển thị nút Yes, No và Cancel
YesNo	4	Hiển thị nút Yes và No
RetryCancel	5	Hiển thị nút Retry và Cancel

Member	Value	Description
Critical	16	Hiển thị biểu tượng
Question	32	Hiển thị biểu tượng
Exclamation	48	Hiển thị biểu tượng
Information	64	Hiển thị biểu tượng
DefaultButton1	0	Nút bấm đầu tiên là ngầm định
DefaultButton2	256	Nút bấm thứ hai là ngầm định
DefaultButton3	512	Nút bấm thứ ba là ngầm định
AplicationModal	0	Ứng dụng là modal có nghĩa là người sử dụng phải nhấn một nút trong hộp thông báo mới có thể tiếp tục làm việc với ứng dụng
SystemModal	4096	System là modal. Mọi ứng dụng phải đợi cho đến khi người sử dụng đáp ứng hộp thông báo
MsgboxSetForegroun d	65536	Chỉ định hộp thông báo như màu nền của window
MsgBoxRight	524288	Text được căn phải
MsgBoxRtlReading	1048576	Text xuất hiện theo thứ tự từ phải sang trái

Bảng 2-6 Các giá trị xác định dạng hộp thoại Msgbox

Nhóm thứ nhất có giá trị từ 0 đến 5 mô tả số lượng và kiểu của các button xuất hiện trong hộp thông báo. Nhóm thứ hai (16,32,48,64) mô tả kiểu của các icon. Nhóm thứ ba (0, 256, 512) xác định nút bấm nào là ngầm định. Nhóm thứ 4 (0,4096) xác định kiểu đáp ứng của hộp thông báo và nhóm thứ 5 chỉ định màu cũng như kiểu căn lề của dòng text trong hộp thông báo.

- Title (tham số tùy chọn) là tiêu đề của hộp thông báo. Nếu bỏ qua tham số này thì tên của ứng dụng sẽ được dùng làm tiêu đề.

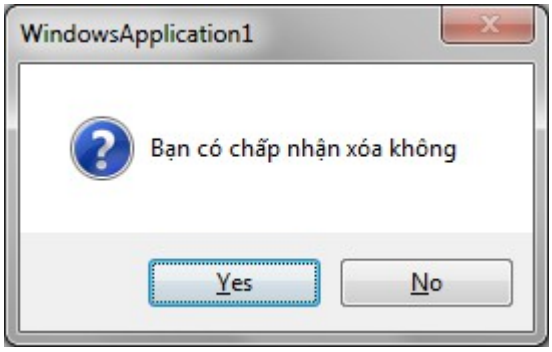
- Giá trị trả về

Hằng số	Giá trị
OK	1
Cancel	2

Hằng số	Giá trị
Abort	3
Retry	4
Ignore	5
Yes	6
No	7

Ví dụ: Để hiển thị hộp thông báo như dưới đây (gồm hai nút Yes, No và icon Question) ta sử dụng lệnh

```
MsgBox("Bạn có chấp nhận xóa không", 36)
```



Trong đó 36 là tổng của 4 (nút YesNo) và 32 (biểu tượng Question).

Muốn lấy giá trị trả về tương ứng với nút người sử dụng đã bấm ta sử dụng lệnh

```
X=MsgBox("Bạn có chấp nhận xóa không", 36)
```

Khi đó nếu người sử dụng nhấn nút Yes ta sẽ có giá trị của x bằng 6; nếu nhấn nút No ta có giá trị của x bằng 7

d. Hiển thị thông báo - MessageBox.Show

Ngoài hàm msgbox, ta có thể sử dụng phương thức Show của lớp MessageBox để hiển thị một hộp thông báo với cú pháp như sau:

```
MessageBox.Show(text,caption,button,icon,defaultButton
,option,displayHelpButton)
```

Trong đó:

- Text: String là nội dung dòng thông báo

- Caption: String là tiêu đề của hộp thoại. Nếu không chỉ định thì tiêu đề sẽ là tên của ứng dụng.

- Button: xác định các nút được hiển thị trong hộp thông báo và thuộc về một trong các giá trị sau:

+ MessageBoxButtons.AbortRetryIgnore

+ MessageBoxButtons.OK

+ MessageBoxButtons.OKCancel

+ MessageBoxButtons.RetryCancel

+ MessageBoxButtons.YesNo

+ MessageBoxButtons.YesNoCancel

- Icon: Biểu tượng xuất hiện trong hộp thoại; nhận một trong các giá trị sau:

+ MessageBoxIcon.Asterisk

+ MessageBoxIcon.Error

+ MessageBoxIcon.Exclamation

+ MessageBoxIcon.Hand

+ MessageBoxIcon.Information

+ MessageBoxIcon.None

+ MessageBoxIcon.Question

+ MessageBoxIcon.Stop

+ MessageBoxIcon.Warning

- Default button: Nút bấm ngầm định nhận 1 trong 3 giá trị Button1, Button2, Button3

- Option: kiểu căn lề của hộp thoại

- DisplayHelpButton (True,False) cho phép hiển thị hay không hiển thị nút Help.

Ví dụ: Lệnh

```
MessageBox.Show("Xóa bản ghi hiện thời?", "Thông báo",
MessageBoxButtons.YesNo, MessageBoxIcon.Question)
```

Sẽ hiển thị hộp thông báo dưới đây

2.4.3 Các lệnh rẽ nhánh

a. Lệnh If

Mẫu 1:

```
If <điều kiện> Then
' Các câu lệnh
End if
```

Trong đó <điều kiện> có thể là biểu thức trả về giá trị True/False hoặc là một giá trị số. Giá trị số khác 0 tương ứng với True, ngược lại là False.

Sử dụng cú pháp này, người lập trình muốn khai báo với trình biên dịch rằng các câu lệnh trong vùng If … End If chỉ được thực hiện nếu như <điều kiện> là đúng.

Nếu chỉ có một lệnh được thực hiện khi điều kiện đúng ta có thể viết lại mẫu 1 như sau:

```
If <điều kiện> then Câu_lệnh
```

Mẫu 2:

```
If <điều kiện> Then
    ' Các câu lệnh khi điều kiện đúng
Else
    ' Các câu lệnh khi điều kiện sai
End if
```

Nếu chỉ có một câu lệnh được thực hiện trong mỗi trường hợp khi điều kiện đúng hoặc sai ta có thể viết như sau:

```
If <điều kiện> Then Câu_lệnh1 Else Câu_lệnh2
```

Trong đó câu_lệnh1 là lệnh được thực hiện khi điều kiện đúng còn câu_lệnh 2 là lệnh được thực hiện khi điều kiện sai.

Ví dụ: Đoạn mã dưới đây:

```
If x>0 then
        Msgbox ("Số dương")
Else
        Msgbox ("Số âm")
End if
```

Có thể được viết như sau:

```
If x>0 then Msgbox ("Số dương") Else Msgbox ("Số âm")
```

Trong trường hợp nhiều điều kiện, chúng ta sử dụng mẫu 3 dưới đây:

Mẫu 3:

```
If <điều kiện 1> Then
    ...
ElseIf <điều kiện 2> Then
    ...
ElseIf <điều kiện n> Then
    ...
Else
    ...
End If
```

Chú ý: Các mệnh đề If … Then … Else có thể lồng nhau.

Ví dụ 2-2: Đoạn chương trình dưới đây sử dụng hàm InputBox để nhập giá trị cho 2 số nguyên x và y sau đó dùng MsgBox để đưa ra thông báo về số lớn nhất và số nhỏ nhất.

```
Dim x%, y%
x = InputBox("Nhập số thứ nhất")
y = InputBox("Nhập số thứ hai")
If x > y Then
   MsgBox("Số lớn nhất là " & x & Chr(13) & Chr(10) _
   & "Số nhỏ nhất là" & y)
Else
   MsgBox("Số lớn nhất là " & y & Chr(13) & Chr(10) _
   & "Số nhỏ nhất là" & x)
End If
```

Trong ví dụ trên biểu thức Chr(13) & Chr(10) được dùng để tách chuỗi thông báo trong hộp thoại thành 2 dòng. Dấu _ dùng để xuống dòng khi chưa kết thúc câu lệnh.

b. Lệnh Case

```
Select Case <biểu thức>
  Case <danh sách 1>
    ' Các lệnh thực hiện trong nhánh 1
  Case <danh sách 2>
    ' Các lệnh thực hiện trong nhánh 2
  ...
  Case Else
    ' Các lệnh thực hiện khi không có nhánh nào ở trên được chọn
End Select
```

Danh sách được đưa ra để làm điều kiện rẽ nhánh có thể rơi vào một trong các trường hợp sau:

- biểu thức
- biểu thức 1, biểu thức 2,...
- biểu thức 1 TO biểu thức 2
- IS <phép toán so sánh> biểu thức

Ví dụ 2-3: Đoạn chương trình sau dùng InputBox để nhập vào giá trị của một tháng trong một năm và hiển thị số ngày của tháng đó.

```
Dim thang, nam, songay As Integer
thang = InputBox("Tháng") : nam = InputBox("Năm:")
Select Case thang
  Case 1, 3, 5, 7, 8, 10, 12
    songay = 31
  Case 4, 6, 9, 11
    songay = 30
  Case Else
    If (nam Mod 400=0) Or (nam Mod 4=0 And nam Mod 100 <>0) _
    Then songay = 29 Else songay = 28
End Select
MsgBox("Tháng " & thang & " năm " & nam & " có " & _
songay & " ngày")
```

Ví dụ 2-4: Nhập vào tuổi của một người và in ra thông báo Thiếu niên nếu tuổi nhỏ hơn 15; Thanh niên nếu tuổi từ 15 đến 25; Trung niên nếu tuổi từ 26 đến 45; Hoa niên nếu tuổi lớn hơn 26

```
Dim tuoi%
tuoi = InputBox("Tuổi:")
Select Case tuoi
   Case Is < 15
      MsgBox("Thiếu niên")
   Case 15 To 25
      MsgBox("Thanh niên")
   Case 26 To 45
      MsgBox("Trung niên")
   Case Is > 46
      MsgBox("Hoa niên")
End Select
```

2.4.4 Câu lệnh lặp

a. For .. Next

Cú pháp:

```
For <biến đếm>  = <giá trị  đầu> To <giá trị  cuối> [Step <bước>]
  ' Các câu lệnh
  [Exit For]
  ' Các câu lệnh
Next [biến đếm]
```

Các câu lệnh trong vùng For … Next chỉ được thực hiện nếu <biến đếm> có giá trị trong đoạn [<giá trị đầu>, <giá trị cuối>]

Sau mỗi lần thực hiện <biến đếm> sẽ được tăng thêm <bước>. Nếu không chỉ định, <bước> có giá trị là 1.

Nếu <bước> có trị > 0, cấu trúc chỉ thực hiện khi <giá trị đầu> <= <giá trị cuối>

Nếu <bước> có trị < 0, cấu trúc chỉ thực hiện khi <giá trị đầu> >= <giá trị cuối>

Mệnh đề Exit For dùng để thoát ngang khỏi vòng lặp

Ví dụ 2-5: Tính số tiền nhận được sau N tháng gửi tiết kiệm biết số tiền đem gửi là S và lãi suất kép là H% một tháng.

```
Dim N%, i%
Dim H, S As Decimal
S = InputBox("Số tiền đem gửi:")
H = InputBox("Lãi suất hàng tháng: (%)")
N = InputBox("Số tháng gửi:")
For i = 1 To N
   S += H * S / 100
Next
MsgBox("Số tiền nhận được là " & S)
```

b. For Each … Next

Cú pháp:

```
For Each <phần tử> In <tập hợp>
' Các câu lệnh
[Exit For]
' Các câu lệnh
Next [phần tử]
```

Với cú pháp này, chương trình sẽ duyệt qua từng phần tử trong tập hợp đang duyệt. Chú ý kiểu của phần tử phải khai báo tương thích với kiểu của tập hợp đang duyệt. `Exit For` dùng để thoát khỏi vòng lặp For

Ví dụ: Đoạn chương trình dưới đây dùng cấu trúc For .. Each .. Next để duyệt qua lần lượt các phần tử trong một tập hợp nhằm tìm phần tử có chứa chuỗi "Hello". Đoạn mã này giả thiết là tập hợp thisCollection đã được thiết lập và mỗi phần tử của nó thuộc kiểu String.

```
Dim found As Boolean = False
Dim thisCollection As New Collection
For Each thisObject As String In thisCollection
    If thisObject = "Hello" Then
        found = True
        Exit For
    End If
Next thisObject
```

c. Do … Loop

Mẫu 1:

```
Do While <biểu thức logic>
' Các câu lệnh
Loop
```

Với cú pháp này, các câu lệnh đặt trong vùng Do While … Loop chỉ thực hiện bao lâu <biểu thức logic> có giá trị True. Sau mỗi lần thực hiện các câu lệnh trong vùng Do While...Loop, <biểu thức logic> sẽ được kiểm tra lại:

Nếu True, thì thực hiện lại vòng lặp

Nếu False, chấm dứt vòng lặp.

Cấu trúc này kiểm tra <biểu thức logic> trước khi thực hiện các lệnh nên sẽ không thực hiện lần nào nếu ngay lần đầu tiên <biểu thức logic> có trị False.

Mẫu 2:

```
Do
' Các câu lệnh
Loop While <biểu thức logic>
```

Tương tự Do While … Loop, các câu lệnh chỉ tiếp tục thực hiện khi <biểu thức logic> có giá trị True và sẽ kiểm tra lại <biểu thức logic> sau mỗi lần thực hiện.

Do kiểm tra sau khi thực hiện nên nếu ngay lần đầu <biểu thức logic> có trị False, các lệnh cũng được thực hiện một lần.

Mẫu 3:

```
Do Untile <biểu thức logic>
' Các câu lệnh
Loop
```

Mẫu 4:

```
Do
' Các câu lệnh
Loop Until <biểu thức logic>
```

Hai cú pháp này tương tự hai cú pháp trên (`Do While … Loop`, `Do … Loop While`), với một khác biệt là chỉ thực hiện hoặc tiếp tục thực hiện khi <biểu thức logic> là `False`.

Chúng ta có thể chấm dứt giữa chừng vòng lặp với lệnh Exit Do

d. While … End While

Cú pháp

```
While <biểu thức logic>
    ' Các câu lệnh
    [ Exit While ]
    ' Các câu lệnh
End While
```

Cách sử dụng như Do While … Loop

Ví dụ 2-6: Tìm ước số chung lớn nhất của 2 số m và n

Cách 1: - Dùng Do .. While

```
Dim n%, m%, UCLN%
n = InputBox("Nhập số thứ nhất:")
m = InputBox("Nhập số thứ hai")
Do While m <> n
        If m > n Then m = m-n Else n = n-m
Loop
UCLN = m
msgBox("USCLN= " & UCLN)
```

Cách 2: - Dùng Do ..Until

```
Do Until m = n
        If m > n Then m = m-n Else n = n-m
Loop
```

Cách 3: - Dùng While .. End While

```
While m <> n
        If m > n Then m = m - n Else n = n - m
End While
```

2.5 Dữ liệu có cấu trúc

2.5.1 Enum

Kiểu Enum là sự liên kết một tập hợp các trị hằng với các tên gợi nhớ. Các trị này mặc nhiên có kiểu Integer và chỉ có thể là kiểu Byte, Short, Long hoặc Integer. Kiểu Enum chỉ được tạo trong các Class hoặc Module.

Cú pháp:

```
[Phạm vi] [ Shadows ] Enum <tên> [ As <Kiểu DL>]
<tên thành phần thứ nhất> [= trị hằng 1]
<tên thành phần thứ hai> [= trị hằng 2]
...
End Enum
```

Trong đó phạm vi được xác định bởi một trong các từ khóa [Public | Protected | Friend | Protected Friend | Private]

Ví dụ:

```
Public Enum DoTuoi as Integer
   Nhidong  = 0
   Thieunien = 1
   Thanhnien = 2
   Trungnien = 3
End Enum
```

Khi không chỉ ra trị hằng, VB.NET sẽ gán trị cho thành phần đầu tiên là 0 và tăng dần cho các thành phần kế tiếp:

Ví dụ:

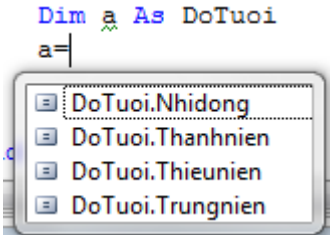
```
Public Enum DoTuoi as Integer
   Nhidong  ' mặc nhiên có trị  0
   Thieunien    ' mặc nhiên có trị  1
   Thanhnien    ' mặc nhiên có trị  2
   Trungnien    ' mặc nhiên có trị  3
End Enum
```

Nếu chỉ gán trị hằng cho thành phần đầu tiên, các thành phần kế tiếp sẽ nhận giá trị tăng dần:

```
Public Enum DoTuoi as Integer
   Nhidong = 100
```

```
   Thieunien    ' mặc nhiên có trị  101
   Thanhnien    ' mặc nhiên có trị  102
   Trungnien    ' mặc nhiên có trị  103
End Enum
```

Sau khi khai báo kiểu Enum, chúng ta có thể khai báo biến kiểu Enum cũng như sử dụng các thành phần của kiểu này thay cho các trị hằng.



2.5.2 Mảng

Mảng là tập hợp các biến có cùng kiểu dữ liệu, cùng tên nhưng có chỉ số khác nhau. Trong VB.Net, mảng có chỉ số bắt đầu là 0 và luôn luôn là mảng động. Chúng ta có các cách khai báo mảng như sau.

- Khai báo không khởi tạo kích thước và giá trị

```
Dim a() as Integer
```

Hoặc

```
Dim a as Integer()
```

- Khai báo có khởi tạo kích thước nhưng không khởi tạo giá trị ban đầu:

```
Dim a(6) as Integer
```

- Khai báo có khởi tạo kích thước và khởi tạo giá trị ban đầu:

```
Dim a() as Integer = {1,2,3,4,5,6,7}
```

Hoặc

```
Dim a() as Integer = New Integer(6){1,2,3,4,5,6,7}
```

Hoặc

```
Dim a() as Integer = New Integer (6) {}
```

Chú ý: Khi dấu { } rỗng, các phần tử có giá trị khởi tạo là giá trị mặc định của kiểu dữ liệu.

Mảng có kiểu tham chiếu nên khi gán hai biến mảng cho nhau, biến được gán sẽ là một tham chiếu đến mảng bên phải toán tử =, khác với trong VB6, là tạo ra một mảng mới có số phần tử mang trị giống nhau.

Mảng thuộc lớp System.Array nên có các thuộc tính và phương thức của lớp này. Sau đây là một số thuộc tính và phương thức đáng chú ý:

Thuộc tính

Tên	Mô tả
Length	Số phần tử của mảng
Rank	Số chiều của mảng

Phương thức:

Tên	Mô tả
BinarySearch	Tìm kiếm trên mảng một chiều đã được sắp xếp giá trị truyền vào, sử dụng thuật giải tìm kiếm nhị phân.
Clear	Gán các phần tử trong dãy chỉ ra bằng giá trị mặc định của kiểu dữ liệu các phần tử
Clone	Trả về bản sao cạn (shallow copy) của mảng. Bản sao này chỉ sao chép kiểu trị và kiểu tham chiếu nhưng không sao chép các đối tượng được tham chiếu đến.
Copy	Sao chép một phần của mảng vào mảng khác và thực hiện chuyển đổi kiểu nếu cần.
CopyTo	Sao chép toàn bộ các phần tử của mảng một chiều vào mảng một chiều được truyền vào bắt đầu từ vị trí chỉ ra.
GetLength	Trả về số phần tử của một chiều được chỉ ra trên mảng
GetLowerBound	Trả về chỉ số nhỏ nhất của một chiều được chỉ ra trên mảng
GetUpperBound	Trả về chỉ số lớn nhất của một chiều được chỉ ra trên mảng
GetValue	Trả về trị của một phần tử chỉ ra trên mảng
IndexOf	Trả về chỉ số của phần tử đầu tiên trên mảng một chiều (hoặc trên một vùng của mảng) trùng với giá trị truyền vào
LastIndexOf	Trả về chỉ số của phần tử cuối cùng trên mảng một chiều (hoặc trên một vùng của mảng) trùng với giá trị truyền vào
Reverse	Đảo ngược thứ tự các phần tử trên mảng một chiều hoặc trên một phần của mảng
SetValue	Gán trị cho một phần tử chỉ ra trên mảng
Sort	Sắp xếp các phần tử trong mảng một chiều

2.5.3 Structure

Khác với mảng, Structure, kiểu do người dùng định nghĩa (UDT: User Defined Type), là một cấu trúc gồm một hoặc nhiều thành phần có kiểu dữ liệu khác nhau. Tuy chúng ta có thể truy xuất riêng lẻ các thành phần nhưng Structure được xem như là một thực thể duy nhất. Trong phiên bản trước, UDT được khai báo với từ khóa Type … End Type. Trong VB.NET, cú pháp khai báo Structure như sau:

```
[Public|Private|Protected] Structure <tên structure>
```

```
   {Dim|Public|Private|Friend}<tên thành phần> As <kiểu dữ
liệu>
   ...
   {Dim|Public|Private|Friend}<tên thành phần N> As <kiểu dữ
liệu> End Structure
```

Với Structure, chúng ta được phép khai báo các phương thức. Sau đây là các đặc điểm của Structure:

C ó các thành phần, kể cả bộ khởi tạo, phương thức, thuộc tính, hằng, sự kiện.

C ó thể cài đặt các lớp giao tiếp (Interface).

C ó thể có các bộ khởi tạo chung, có hoặc không có tham số.

Structure l à kiểu trị.

T ất cả các thành phần của Structure mặc định là Public.

C ác thành phần của Structure không được khai báo với từ khóa Protected.

Structure kh ông thể kế thừa.

Mỗi Structure có một bộ khởi tạo mặc nhiên không tham số ban đầu. Bộ khởi tạo này sẽ khởi tạo mọi thành phần dữ liệu của Structure với giá trị mặc định của chúng. Chúng ta không thể định nghĩa lại chức năng này.

Vì Structure là kiểu trị (Value Type), nên mỗi biến Structure luôn luôn gắn liền với một thể hiện Structure.

2.6 Xử lý lỗi

2.6.1 Phân loại lỗi

Trong VB.NET, chúng ta có thể gặp các loại lỗi sau:

Syntax error

Lỗi cú pháp, còn gọi là lỗi trong lúc thiết kế. Những lỗi này dễ chỉnh sửa vì VB.NET sẽ kiểm tra cú pháp khi ta đang nhập từ bàn phím nên sẽ báo lỗi tức thời khi ta gõ sai hoặc dùng một từ không thích hợp.

Run-time error

Lỗi thực thi xảy ra khi chương trình đang thực thi. Đây là những lỗi khó xác định hơn lỗi cú pháp. Lỗi thực thi có thể từ các lý do khác nhau như:

M ở một tệp tin kh ông tồn tại

Truy xu ất một tệp tin mà người dùng kh ông có quyền trên đó

Truy xu ất dữ liệu một nguồn kh ông tồn tại trong CSDL

Chia cho s ố 0

Nh ập/xu ất dữ liệu cho một thiết bị đã hỏng hoặc đang bận v.v…

Logic error

Lỗi luận lý cũng xảy ra khi chương trình đang thực thi và được thể hiện dưới những hình thức hay những kết quả không mong đợi. Loại lỗi này thường do sai lầm trong thuật giải.

2.6.2 Xử lý lỗi

Một lỗi xảy ra khi chương trình đang chạy gọi là một Exception. Trong CLR, Exception là một đối tượng từ lớp System.Exception. Chúng ta cần lưu ý một lỗi xảy ra trong lúc thực thi không làm treo chương trình, nhưng nếu không được xử lý sẽ làm treo chương trình. CLR chỉ ra tình trạng lỗi qua lệnh Throw. Lệnh này sẽ đưa ra một đối tượng kiểu System.Exception chứa thông tin về lỗi đang xảy ra.

VB.NET cung cấp một cơ chế xử lý lỗi hoàn chỉnh với cú pháp sau:

```
Try
   ' các lệnh có khả  năng gây lỗi
   Catch
       ' các lệnh xử  lý khi lỗi xảy ra

   [Finally]
       ' các lệnh thực hiện sau cùng
End Try
```

Cấu trúc này cho phép chúng ta thử (Try) thực hiện một khối lệnh xem có gây lỗi không; nếu có sẽ bẫy và xử lý (Catch) lỗi.

Cấu trúc này chia làm các khối sau:

Khối Try:

Chứa các câu lệnh có khả năng gây lỗi

Khối Catch:

Các dòng lệnh để bẫy và xử lý lỗi phát sinh trên khối Try. Khối này gồm một loạt các lệnh bắt đầu với từ khóa Catch, biến kiểu Exception ứng với một kiểu Exception muốn bẫy và các lệnh xử lý. Dĩ nhiên, chúng ta có thể dùng một lệnh Catch cho các System.Exception, nhưng như thế sẽ không cung cấp thông tin đầy đủ cho người dùng về lỗi đang xảy ra cũng như hướng dẫn cách xử lý cụ thể cho mỗi tình huống. Ngoài những lỗi đã xử lý, có thể xảy ra những lỗi ngoài dự kiến, để xử lý các lỗi này, chúng ta nên đưa thêm một lệnh Catch để bẫy tất cả các trường hợp còn lại và xuất thông tin về lỗi xảy ra.

Khối Finally:

Khối tùy chọn, sau khi chạy qua các khối Try và Catch nếu không có chỉ định nào khác, khối Finally sẽ được thực hiện bất kể có xảy ra lỗi hay không.

Cuối cùng, cấu trúc bẫy và xử lý lỗi chấm dứt với từ khóa End Try. Cú pháp chung cho một cấu trúc xử lý lỗi như sau

```
Try
   ' khối lệnh có thể  gây lỗi
   Catch <biến1> As <Kiểu Exception> [When <biểu thức>]
      ' khối lệnh bẫy và xử  lý lỗi
   Catch <biến2> As <Kiểu Exception> [When <biểu thức>]
      ' khối lệnh bẫy và xử  lý lỗi
   Finally
      ' khối lệnh kết thúc
End Try
```

Ví dụ:

```
Dim d as Double, i as Integer
Try
   i = CInt(InputBox("Xin nhập một số  nguyên"))
   d = 42 \ i
   Catch ex As DivideByZeroException
      Messagebox.Show("Không thể  chia cho số  không")
   Catch ex As InvalidCastException
      Messagebox.Show("Xin nhập số  nguyên !")
End Try
```

Câu lệnh Catch có thể có nhiều cách sử dụng:

a. Bẫy không có điều kiện

```
Dim d as Double, i as Integer
Try
   i = CInt(InputBox(“Xin nhập một số  nguyên”))
   d = 42 \ i
   Catch
     Messagebox.Show(“Không thể  chia")
End Try
```

b. Bẫy với kiểu lỗi chung Exception

```
Dim d As Double, i As Integer
Try
   i = InputBox("Xin nhập một số  nguyên")
   d = 42 \ i
   Catch ex As Exception
      MessageBox.Show("Không thể  chia")
```

```
End Try
```

c. Bẫy với những kiểu Exception đặc biệt

Các kiểu Exception đặc biệt thường gặp:

Tên	Mô tả
ArgumentException	Tham số truyền không hợp lệ
DivideByZeroException	Chương trình thực hiện phép chia một số cho số không
OverFlowException	Kết quả của một phép toán hoặc của một phép chuyển đổi kiểu lớn hơn khả năng lưu giữ của biến
FieldAccessException	Chương trình truy xuất một field Private hoặc Protected của lớp
InvalidCastException	Chương trình đang cố thực hiện một chuyển đổi không hợp lệ
InvalidOperationException	Chương trình đang cố gọi một thủ tục không hợp lệ
MemberAccessException	Truy xuất một thành phần của một lớp bị thất bại
MethodAccessException	Chương trình đang cố gọi thủ tục Private hoặc Protected của lớp
NullReferenceException	Chương trình đang cố truy xuất một đối tượng không tồn tại
TypeUnloadException	Chương trình truy xuất một lớp chưa được tải lên vùng nhớ

d. Bẫy với điều kiện When

```
Dim d As Double, i As Integer
Try
   i = InputBox("Xin nhập một số  nguyên")
   d = 42 \ i
   Catch ex As Exception When i = 0
     MessageBox.Show("Không thể  chia cho số  không")
End Try
```

2.7 Một số chỉ thị của VB.NET

2.7.1 Chỉ thị Option Explicit ON\OFF

Khi Option Eplicit là ON, VB.NET không cho phép sử dụng các biến mà không khai báo trước.

2.7.2 Chỉ thị #Region … #End Region

Chỉ thị #Region … #End Region được dùng để đánh dấu một khối lệnh có thể thu gọn, giãn ra trên cửa sổ viết lệnh.

Cú pháp:

```
#Region <chuỗi định danh>
' khối lệnh
#End Region
```

Trong đó <chuỗi định danh>: bắt buộc, có giá trị kiểu String, là tiêu đề của khối lệnh

Ví dụ với đoạn mã sau:

```
#Region "Các khai báo"
' Đưa vào các dòng lệnh khai báo.
#End Region
```

Khi thu lại ta sẽ thấy trên cửa sổ lệnh:

```
+ Các khai báo
```

2.8 Thủ tục (Procedure)

2.8.1 Khái niệm

Thủ tục (*procedure)* là một khối lệnh Visual Basic được đặt trong cặp từ khóa khai báo (**Function**, **Sub**, **Operator**, **Get**, **Set**) và từ khóa kết thúc tương ứng với nó(**End**).

Thủ tục được thi hành tại bất kỳ một vị trí nào trong chương trình. Lệnh thi hành một thủ tục được xem như lời gọi thủ tục. Khi thực hiện xong các lệnh trong một thủ tục ứng dụng sẽ quay lại thực hiện câu lệnh tiếp theo câu lệnh đã gọi thủ tục. Thủ tục có thể được gọi trong một câu lệnh hoặc được gọi trong một biểu thức.

Trong VB.NET thủ tục được chia thành 5 loại: Sub Procedure, Function Procedure, Property Procedure, Operator Procedure,Generic Procedure.

2.8.2 Tham số của thủ tục

Định nghĩa

- Các thông số đầu vào của một thủ tục được gọi là tham số của thủ tục.

- Tham số (parameter) là các tên (định danh) được các thủ tục dùng để nhận thông tin đầu vào để thực hiện các xử lý. Khi một thủ tục được gọi thì nó sẽ đòi hỏi các đối số cụ thể tương ứng với các tham số và thông thường các đối số này phải được gán giá trị cụ thể trước khi thủ tục thực thi.

Phân loại tham số

Có hai loại tham số.

- Tham biến: thay đổi (về mặt giá trị sau lời gọi hàm). Tham số loại này trong VB.NET được khai báo với từ khóa ByRef.

- Tham trị: Không đổi (về mặt giá trị sau lời gọi hàm). Tham số loại này trong VB.NET được khai báo với từ khóa ByVal.

2.8.3. Sub Procedure

Sub là một dãy các câu lệnh để thực thi một công việc, một chức năng đặc thù nào đó, sub được xem như là một thành phần của chương trình. Nói một cách khác, sub là các câu lệnh được nhóm vào một khối và được đặt tên.

Các sub có thể được gọi để thi hành (thường là thông qua subname). Điều này cho phép gọi tới những thủ tục nhiều lần mà không cần phải lặp lại các khối lệnh giống nhau một khi đã hoàn tất việc viết mã lệnh cho các thủ tục đó chỉ một lần.

- Từ khóa để tham khảo khái niệm thủ tục trong MSDN là "Sub Procedures, Sub Statement".

Khai báo sub

Cấu trúc tổng quát của việc định nghĩa một thủ tục được đưa ra như sau:

```
[<attributelist>] [accessmodifier]
[proceduremodifiers] [Shared] [Shadows]
Sub name [(Of typeparamlist)] [(parameterlist)]
[Implements implementslist | Handles eventlist]
        các lệnh
        [ Exit Sub ]
        các lệnh
End Sub
```

Cú pháp thông dụng nhất để khai báo một thủ tục dạng sub là:

```
[Public|Protected|Private] Sub <Tên thủ tục> [(DS tham số)]
        <Khai báo biến>
        Câu lệnh 1
        Câu lệnh 2
        ...
End Sub
```

Trong đó

- Public, Protected, Private là các từ khóa xác định phạm vi của thủ tục

- Tên thủ tục đặt theo qui tắc giống tên biến. Trong VB.NET các thủ tục có thể trùng tên nhau song số tham số hoặc kiểu dữ liệu của các tham số phải khác nhau.

- Danh sách tham số: phải xác định rõ là tham biến hay tham trị.

Ví dụ: Thủ tục nhập tên, điểm toán và điểm văn của một sinh viên:

```
Public Sub Nhap(ByRef HoTen As String,ByRef Toan As _
Integer,ByRef Van As Integer)
  Console.Write("Nhap ho ten:")
  HoTen = Console.ReadLine()
  Console.Write("Nhap toan:")
  Toan = Console.ReadLine()
  Console.Write("Nhap van:")
  Van = Console.ReadLine()
End Sub
```

Trong thủ tục này, giá trị của các tham số HoTen, Toan, Van sau khi kết thúc thủ tục đã được thay bằng các giá trị mới nhập từ bàn phím do đó chúng phải là tham biến và được khai báo với từ khóa ByRef.

Ví dụ: Thủ tục xuất ra màn hình Console tên và điểm toán, điểm văn, điểm trung bình của sinh viên:

```
Public Sub Xuat(ByVal HoTen As String, ByVal Toan As  _
Integer,ByVal Van As Integer, ByVal DiemTrungBinh As Double)
  Console.WriteLine("Ho ten :" & HoTen)
  Console.WriteLine("Toan :" & Toan)
  Console.WriteLine("Van:" & Van)
  Console.WriteLine("Diem Trung Binh:" & DiemTrungBinh)
End Sub
```

Trong thủ tục Xuat; các tham số HoTen, Toan, Van, DiemTrungBinh không thay đổi giá trị do đó có thể khai báo chúng là tham trị bằng cách sử dụng từ khóa ByVal.

Lời gọi thủ tục:

- Thủ tục sử dụng như một lệnh bằng cách viết tên của thủ tục cùng với danh sách tham số.

- Có thể sử dụng câu lệnh Call để gọi thủ tục.

- Thông thường, thủ tục sẽ kết thúc và trở về đơn thể gọi nó khi gặp từ khóa End Sub. Tuy nhiên có thể thoát trực tiếp khỏi thủ tục tại bất kỳ một vị trí nào trong thân nó bằng câu lệnh Exit Sub hoặc Return.

Ví dụ 2-8: Chương trình nhập họ tên, điểm toán, điểm văn của một sinh viên và xuất ra màn hình các thông tin về họ tên, điểm toán, điểm văn và điểm trung bình của sinh viên đó.

Chương trình được xây dựng gồm 1 đơn thể - Module chứa 3 thủ tục là Nhap (nhập dữ liệu), Xuat (hiển thị dữ liệu), XuLy(tính điểm trung bình). Thủ tục Main() chứa các lệnh cần thiết để khai báo biến, gọi và truyền tham số cho các thủ tục.

```
Imports System
Public Module Module1

Public Sub Nhap(ByRef HoTen As String,ByRef Toan As _
Integer,ByRef Van As Integer)
  Console.Write("Nhap ho ten:")
  HoTen = Console.ReadLine()
  Console.Write("Nhap toan:")
  Toan = Console.ReadLine()
  Console.Write("Nhap van:")
  Van = Console.ReadLine()
End Sub

Public Sub XuLy(ByVal Toan As Integer, ByVal Van As _
Integer,ByRef DiemTrungBinh As Double)
  DiemTrungBinh = (Toan + Van) / 2
End Sub

Public Sub Xuat(ByVal HoTen As String, ByVal Toan As _
Integer,ByVal Van As Integer, ByVal DiemTrungBinh As Double)
  Console.WriteLine("Ho ten :" & HoTen)
  Console.WriteLine("Toan :" & Toan)
  Console.WriteLine("Van:" & Van)
  Console.WriteLine("Diem Trung Binh:" & DiemTrungBinh)
End Sub

Sub Main()
  Dim ht As String = ""
```

```
  Dim t,v As Integer
  Dim tb As Double
  Nhap(ht, t, v)
  XuLy(t, v, tb)
  Xuat(ht, t, v, tb)
End Sub

End Module
```

2.8.4 Function Procedure (Hàm)

Một hàm là dãy các lệnh để thực thi một thao tác đặc thù nào đó như là một phần của chương trình lớn hơn. Nói một cách khác hàm là các câu lệnh được nhóm vào một khối, được đặt tên và có một giá trị trả về.

Các hàm có thể được gọi để thi hành (thường là thông qua tên của hàm). Điều này cho phép gọi tới hàm nhiều lần mà không cần phải lặp lại các khối mã giống nhau một khi đã hoàn tất việc viết mã cho các hàm đó chỉ một lần.

Trong VB.NET quan niệm hàm là một thủ tục có giá trị trả về.

Từ khóa để tham khảo về hàm trong MSDN là "Function Statement".

Cú pháp khai báo hàm

Hàm được định nghĩa và khai báo tương tự như thủ tục.

```
[<attributelist>][accessmodifier][proceduremodifiers]
[Shared][Shadows]
Function name [(Of typeparamlist)] [(parameterlist)]
[As returntype]
   [Implements implementslist | Handles eventlist]
   [ statements ]
   [ Exit Function ]
   [ statements ]
End Function
```

Thông dụng nhất của việc khai báo hàm là như sau:

```
[Public, Protected, Private] Function <Tên Hàm>
[(Danh sách các tham số)] As Kiểu dữ liệu trả về
   <Khai báo các biến>
   Các câu lệnh
   [Exit Function]
   Các câu lệnh
   ...
```

```
  End Function
```

Trong đó

- Public, Protected, Private là các từ khóa xác định phạm vi của hàm.

- Tên hàm đặt theo qui tắc giống tên sub; có thể trùng tên nhau miễn là khác về kiểu dữ liệu, số tham số.

- Danh sách tham số: phải xác định rõ là tham biến hay tham trị.

- Kiểu dữ liệu trả về: Là kiểu dữ liệu của biểu thức mà hàm trả về.

- Lệnh `Exit Function` dùng để thoát ngang một hàm.

- Nếu muốn hàm trả về một giá trị ta dùng lệnh `return biểu thức.`

Ví dụ: Thủ tục XuLy để tính điểm trung bình trong ví dụ 3-3 có thể xây dựng bằng hàm như sau:

```
Public Function XuLy(ByVal Toan As Integer, ByVal Van _
As Integer) As Double
  return (Toan+Van)/2
End Function
```

Hàm trả về giá trị (Toan+Van)/2 bằng câu lệnh return.

Trong trường hợp này Sub Main() được viết như sau:

```
Sub Main()
  Dim ht As String = ""
  Dim t,v As Integer
  Dim tb As Double
  Nhap(ht, t, v)
  tb=XuLy(t, v)
  Xuat(ht, t, v, tb)
End Sub
```

Lời gọi hàm

- Hàm được sử dụng trong biểu thức bằng cách viết tên hàm kèm theo danh sách các tham số . Ví dụ ta có thể viết `tb=XuLy(Toan,Van)`

- Trong trường hợp không cần quan tâm đến giá trị của hàm, ta có thể gọi hàm như cách gọi của thủ tục.

NỘI DUNG PHẦN THẢO LUẬN

1. Phương pháp lập trình hướng thủ tục và cách xây dựng.

2. Tham số, tham biến, giá trị trả về của sub, function

3. Cấu trúc và các giải thuật trên cấu trúc

TÓM TẮT NỘI DUNG CỐT LÕI

Trong chương này sinh viên cần chú ý đến các nội dung sau:

• Khai báo và sử dụng các biến

• Cách viết các hằng giá trị trong VB.NET

• Toán tử

• Lệnh rẽ nhánh, Vòng lặp

• Mảng, Cấu trúc

• Sub, Function

BÀI TẬP ỨNG DỤNG

Dùng ngôn ngữ lập trình VB.NET để tạo các ứng dụng Console giải các bài tập dưới đây:

2.1. Xây dựng, biên dịch và chạy thử các chương trình tương ứng với ví dụ 2-1, 2-2, 2-3, 2-4, 2-5, 2-6.

2.2. Viết chương trình nhập vào một phân số. Hãy cho biết phân số đó là phân số âm hay dương hay bằng không.

2.3. Viết chương trình nhập tọa độ hai điểm trong không gian. Tính khoảng cách giữa chúng và xuất kết quả.

2.4. Viết chương trình giải phương trình bậc nhất ax+b=0

2.5. Viết chương trình giải phương trình bậc hai $ax^2+bx+c=0$

2.6. Viết chương trình nhập vào một số nguyên và kiểm tra xem đó có phải là số chính phương hay không.

2.7. Viết chương trình nhập vào một số kiểu Long và kiểm tra xem số đó là số nguyên tố hay hợp số.

2.8. Xây dựng hàm `Public Function SoNT (ByVal n as Long) as Boolean`. Hàm trả về True nếu n là số nguyên tố và trả về giá trị False nếu n là hợp số.

HƯỚNG DẪN TỰ HỌC Ở NHÀ

1. Bài tập 2.1:

* Chạy ví dụ 2-1 (chương trình nhập tên, điểm toán, điểm văn và in ra tên, điểm trung bình).

- Dùng Notepad biên soạn file có tên C:\vd1.vb với nội dung là chương trình trong ví dụ 2-1.

- Khởi động Visual Studio 2005 Command prompt

- Biên dịch chương trình vd1.vb thành file exe bằng lệnh vbc C:\vd1.vb và chạy chương trình từ file exe này.

* Sử dụng lệnh Console.Write() thay cho MsgBox và Console.Readline() thay cho InputBox để viết lại các chương trình tương ứng với các ví dụ 2-2, 2-3, 2-4, 2-5, 2-6. Sau đó dùng Visual Studio 2005 Command prompt để biên dịch và chạy thử các chương trình này.

2. Bài tập 2.2: Sử dụng hai biến a và b cho tử số và mẫu số. Nhập dữ liệu lần lượt cho từng biến sau đó kiểm tra các trường hợp. Chú ý kiểm tra mẫu số phải khác 0 trước khi thực hiện phép chia.

3. Bài tập 2.5: Sử dụng hàm SQRT(d) để lấy căn bậc 2 của d (delta). Chú ý bổ sung lệnh `Imports System.Math` vào đầu chương trình sau lệnh `Imports System` để có thể sử dụng được các hàm toán học này.

4. Bài tập 2.6: Sử dụng hàm ROUND (làm tròn) và SQRT (tính căn bậc 2). Hai hàm này nằm trong Namespace System.Math.

Chương 3
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG VB.NET

MỤC TIÊU CỦA CHƯƠNG

- Về kiến thức

+ Củng cố các kiến thức về lập trình hướng đối tượng.

+ Trang bị cho sinh viên cách khai báo lớp, đối tượng, cách xây dựng các phương thức và thuộc tính trong VB.NET

- Về kỹ năng

+ Sau khi kết thúc bài sinh viên hiểu được cách vận dụng lập trình hướng đối tượng trong VB.NET

+ Tự xây dựng được một số lớp, đối tượng, phương thức đơn giản.

+ Kết hợp cú pháp lập trình VB.NET đã học trong chương 2 với phương pháp lập trình hướng đối tượng để viết các ứng dụng Console giải quyết các bài tập được cho.

- Về thái độ:

Giúp sinh viên có sự so sánh tương quan giữa VB.NET với một số ngôn ngữ lập trình hướng đối tượng khác như C++

NỘI DUNG BÀI GIẢNG LÝ THUYẾT

3.1 Tổng quan về lập trình hướng đối tượng

Tư tưởng chính của lập trình hướng đối tượng là xây dựng một chương trình dựa trên sự phối hợp hoạt động của các đối tượng. Một đối tượng bao gồm hai thành phần chính là thông tin lưu trữ và các thao tác xử lý. Trong thế giới thực, đối tượng là thực thể tồn tại như con người, xe, máy tính, v.v…Trong ngôn ngữ lập trình, đối tượng có thể là màn hình, điều khiển v.v…

Lập trình hướng đối tượng là kiểu lập trình nhằm vào sự tương tác giữa các đối tượng. Mỗi đối tượng có những thuộc tính (thông tin lưu trữ), những phương thức xác định các chức năng của đối tượng. Bên cạnh đó, đối tượng cũng có khả năng phát sinh các sự kiện khi thay đổi thông tin, thực hiện một chức năng hay khi đối tượng khác tác động vào. Tất cả những thuộc tính, phương thức và sự kiện tạo nên cấu trúc của đối tượng. Có bốn ý niệm trong Lập trình hướng đối tượng bao gồm:

Abstraction: T ính trừu tượng

Encapsulation: T ính bao bọc

Inheritance: T ính kế thừa

Polymorphism: T ính đa hình

Mỗi ý niệm đều có vai trò quan trọng trong lập trình hướng đối tượng.

3.1.1 Tính trừu tượng

Chúng ta thường lẫn lộn giữa lớp (Class) và đối tượng (Object). Cần phân biệt lớp là một ý niệm trừu tượng, còn đối tượng là một thể hiện của lớp.

Ví dụ Class ConNgười là một ý niệm trừu tượng, nhưng NguyễnVăn A là một đối tượng cụ thể. Từ những đối tượng giống nhau, chúng ta có thể trừu tượng hóa thành một lớp đối tượng.

Tính trừu tượng cho phép chúng ta loại bỏ tính chất phức tạp của đối tượng bằng cách chỉ đưa ra các thuộc tính và phương thức cần thiết của đối tượng trong lập trình.

3.1.2 Tính bao bọc

Mỗi Class được xây dựng để thực hiện một nhóm chức năng đặc trưng của riêng Class, trong trường hợp một đối tượng thuộc Class cần thực hiện một chức năng không nằm trong khả năng vì chức năng đó thuộc về một đối tượng thuộc Class khác, nó sẽ yêu cầu đối tượng đó đảm nhận thực hiện công việc. Một điểm quan trọng trong cách giao tiếp giữa các đối tượng là một đối tượng sẽ không được truy xuất trực tiếp vào thành phần dữ liệu của đối tượng khác cũng như không đưa thành phần dữ liệu của mình cho đối tượng khác một cách trực tiếp. Tất cả mọi thao tác truy xuất vào thành phần dữ liệu từ đối tượng này qua đối tượng khác phải được thực hiện bởi các phương thức (method) của chính đối tượng chứa dữ liệu. Đây cũng chính là một tính chất quan trọng trong lập trình hướng đối tượng gọi là tính bao bọc (encapsulation) dữ liệu.

Tính bao bọc cho phép dấu thông tin của đối tượng bằng cách kết hợp thông tin vớiø các phương thức liên quan đến thông tin trong đối tượng.

Ví dụ: Xe hơi có các chức năng (phương thức phô diễn bên ngoài) như Ngừng, Chạy tới, Chạy lùi. Đây là những gì cần thiết cho Tài xế khi tương tác với Xe hơi. Xe hơi có thể có một đối tượng Động cơ nhưng Tài xế không cần phải quan tâm. Tất cả những gì cần quan tâm là những chức năng để có thể vận hành xe. Do đó, khi thay một Động cơ khác, Tài xế vẫn sử dụng các chức năng cũ để vận hành Xe hơi bao lâu các phương thức phô diễn bên ngoài (Interface) không bị thay đổi.

3.1.3 Tính kế thừa

Tính kế thừa là khả năng cho phép ta xây dựng một lớp mới dựa trên các định nghĩa của một lớp đã có. Lớp đã có gọi là lớp Cha, lớp mới phát sinh gọi là lớp Con và đương nhiên kế thừa tất cả các thành phần của lớp Cha, có thể mở rộng công năng các thành phần kế thừa cũng như bổ sung thêm các thành phần mới

3.1.4 Tính đa hình

Tính đa hình là khả năng một ngôn ngữ xử lý các đối tượng hữu quan theo cùng một cách. Tính đa hình thể hiện dưới nhiều hình thức:

*** Kết nối trễ - Late Binding**

Đây là khả năng cho phép người lập trình gọi trước một phương thức của đối tượng, tuy chưa xác định được đối tượng. Đến khi thực hiện, chương trình mới xác định được đối tượng và gọi phương thức tương ứng của đối tượng đó. Kết nối trễ giúp chương trình được uyển chuyển chỉ yêu cầu đối tượng cung cấp đúng phương thức cần thiết là đủ.

Ví dụ: Chúng ta có lớp Xe với phương thức Chạy và các lớp Xe đạp, Xe hơi, Xe đẩy cùng phát sinh từ lớp Xe.

Chúng ta chưa biết sẽ sử dụng xe gì để di chuyển vì tùy thuộc tình hình có sẵn xe nào nên gọi trước phương thức Chạy. Khi chương trình thực thi, tùy theo đối tượng của lớp nào được đưa ra mà phương thức Chạy của đối tượng đó được gọi.

*** Nạp chồng - Overloading**

Khả năng cho phép một lớp có nhiều thuộc tính, phương thức cùng tên nhưng với các tham số khác nhau về loại cũng như về số lượng. Khi được gọi, dựa vào tham số truyền vào, thuộc tính hay phương thức tương ứng sẽ được thực hiện.

*** Ghi chồng - Overriding**

Hình thức này áp dụng cho lớp Con đối với lớp Cha. Lớp Con được phép có một phương thức cùng tên, cùng số tham số có kiểu dữ liệu như phương thức của lớp Cha hoặc những lớp trước đó nữa (lớp phát sinh ra lớp Cha …) với cài đặt khác đi. Lúc thực thi, nếu lớp Con không có phương thức riêng, phương thức của lớp Cha sẽ được gọi, ngược lại nếu lớp Con có phương thức riêng thì phương thức này sẽ được gọi.

3.2 Lớp đối tượng

3.2.1 Khai báo lớp

Lớp hiểu một cách đơn giản là sự tích hợp giữa hai thành phần: thành phần dữ liệu và thành phần xử lý.

Cú pháp khai báo lớp

```
[AccessModifier][Keyword] Class _ ClassName [Implements
InterfaceName]
        'Declare properties and methods
End Class
```

Trong đó

- 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, sử dụng một trong các 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

- Từ khóa để tham khảo khái niệm class trong MSDN là "Class statement"

3.2.2 Đối tượng

Đối tượng là 1 sự thể hiện của lớp. Một lớp có thể có nhiều sự thể hiện khác nhau.

Cú pháp khai báo đối tượng:

```
Dim variablename As [New] { objectclass | Object }
```

Ví dụ minh hoạ:

```
Dim a As new CHocSinh
Dim x As new ChocSinh
Dim y As new CHocSinh
```

Trong ví dụ trên ta nói a,x,y là 3 đối tượng thuộc class CHocSinh. Nói cách khác, trong ví dụ này, lớp CHocSinh có 3 thể hiện.

Chú ý:

- Có thể có nhiều đối tượng cùng thuộc về một lớp.

- Từ khóa để tham khảo khái niệm đối tượng trong MSDN là "object variables, declaring"

3.2.3 Phương thức

Phương thức là chức năng mà đối tượng thuộc về lớp có thể thực hiện. Đó có thể là một thủ tục hoặc một hàm.

3.2.4 Ví dụ về thiết kế lớp

Viết chương trình nhập họ tên, điểm toán, điểm văn của một học sinh. Tính điểm trung bình và xuất kết quả ra Console.

Đơn thể lớp học sinh:

```
Public Class CHocSinh
   Private HoTen As String

   Private Toan As Integer
   Private Van As Integer
   Private DiemTrungBinh As Double

   Public Sub New()
       Return
   End Sub

   Public Sub Nhap()
       Console.Write("Nhap ho ten:")
       HoTen = Console.ReadLine()
       Console.Write("Nhap toan:")
       Toan = Console.ReadLine()
       Console.Write("Nhap van:")
       Van = Console.ReadLine()
   End Sub

   Public Sub XuLy()
       DiemTrungBinh = (Toan + Van) / 2

   End Sub

   Public Sub Xuat()
       Console.WriteLine("Ho ten :" & HoTen)
       Console.WriteLine("Toan :" & Toan)
       Console.WriteLine("Van:" & Van)
       Console.WriteLine("Diem Trung Binh:" &
DiemTrungBinh)
   End Sub
End Class
```

Đơn thể chính:

```
Module Module1
   Sub Main()
       Dim hs As New CHocSinh

       hs.Nhap()
       hs.XuLy()
       hs.Xuat()
   End Sub
End Module
```

3.3 Phương thức thiết lập và phương thức phá hủy

3.3.1 Phương thức thiết lập

Các phương thức thiết lập của một lớp có nhiệm vụ thiết lập thông tin ban đầu cho các đối tượng thuộc về lớp ngay khi đối tượng được khai báo.

a. Các đặc điểm của phương thức thiết lập

- Phương thức thiết lập của lớp được định nghĩa thông qua toán tử new.
- Không có giá trị trả về.
- Được tự động gọi thực hiện ngay khi đối tượng được khai báo.
- Có thể có nhiều phương thức thiết lập trong một lớp.
- Trong một quá trình sống của đối tượng thì chỉ có 1 lần duy nhất phương thức thiết lập được gọi thực hiện đó là khi đối tượng được khai báo.
- Các phương thức thiết lập của lớp thuộc nhóm các phương thức khởi tạo.
- Từ khóa để tham khảo trong MSDN là "New constructor".

Chú ý:

Khi khai báo một lớp đối tượng trong VB.NET ta phải định nghĩa ít nhất một phương thức thiết lập cho lớp đó. Thông thường các lập trình viên chọn là phương thức thiết lập mặc định (không nhận tham số đầu vào). Tuy nhiên, đây không phải là phương thức thiết lập bắt buộc phải định nghĩa.

b. Phân loại phương thức thiết lập

Để đơn giản ta có thể chia các phương thức thiết lập của một lớp thành 3 nhóm như sau:

- Phương thức thiết lập mặc định.
- Phương thức thiết lập sao chép.
- Phương thức thiết lập nhận tham số đầu vào.

Về mặt nguyên tắc có bao nhiêu phương thức khởi tạo thì có bấy nhiêu phương thức thiết lập, phương thức khởi tạo mặc định thì tương ứng với phương thức thiết lập mặc định, phương thức khởi tạo dựa vào 1 đối tượng khác tương ứng 1 phương thức thiết lập sao chép, các phương thức khởi tạo còn lại tương ứng với lại phương thức thiết lập nhận tham số đầu vào.

c. Ứng dụng phương thức thiết lập

Ví dụ: Cần xác định và cài đặt các phương thức thiết lập cho lớp phân số.

Lớp phân số có hai thuộc tính tử số và mẫu số.

- Phương thức thiết lập mặc định: tử số được lấy mặc định là 0 và mẫu số được lấy mặc định là 1.

- Phương thức thiết lập khi biết tử số: tử số được gán giá trị tương ứng với giá trị của đối số đầu vào và mẫu số được lấy mặc định là 1.

- Phương thức thiết lập khi biết đầy đủ thông tin: tử số và mẫu số được gán giá trị tương ứng với giá trị của các đối số đầu vào.

- Phương thức thiết lập sao chép: nhận tham số đầu vào là một đối tượng cùng thuộc về lớp phân số và tạo ra một đối tượng phân số mới giống hoàn toàn đối tượng phân số đối số tương ứng.

Chương trình minh họa

ĐƠN THỂ LỚP PHÂN SỐ:

```
Public Class CphanSo

'Các thuộc tính
   Private Tu As Integer

   Private Mau As Integer

' Phương thức thiết lập mặc định

   Public Sub New()

       Tu = 0

       Mau = 1

   End Sub

' Phương thức thiết lập khi biết tử số
   Public Sub New(ByVal t As Integer)
       Tu = t
       Mau = 1
   End Sub

' Phương thức thiết lập khi biết đầy đủ thông tin
   Public Sub New(ByVal t As Integer,

       ByVal m As Integer)

       Tu = t
       Mau = m

   End Sub

' Phương thức thiết lập sao chép
   Public Sub New(ByRef ps As CPhanSo)

       Tu = ps.Tu
```

```
        Mau = ps.Mau
    End Sub

    Public Sub Nhap()
        Console.Write("Nhap tu: ")
        Tu = Console.ReadLine()
        Console.Write("Nhap mau: ")
        Mau = Console.ReadLine()
    End Sub


    Public Sub Xuat()
        Console.WriteLine(Tu & "/" & Mau)

    End Sub
End Class
```

HƯỚNG DẪN SỬ DỤNG CONSTRUCTOR

```
Module Module1
    Sub Main()

        Dim a As New CPhanSo

        a.Xuat()

        Dim b As New CPhanSo(1)
        b.Xuat()

        Dim c As New CPhanSo(1, 2)

        c.Xuat()
        Dim d As New CPhanSo(c)

        d.Xuat()
    End Sub
End Module
```

3.3.2 Phương thức phá hủy

Phương thức phá huỷ có nhiệm vụ thu hồi lại tất cả các tài nguyên đã cấp phát cho đối tượng trong quá trình sống của đối tượng khi đối tượng hết phạm vi hoạt động.

Đặc điểm của phương thức phá hủy

- Phương thức phá hủy của lớp được định nghĩa thông qua toán tử Finalize.
- Không có giá trị trả về.
- Không có tham số đầu vào.
- Được tự động gọi thực hiện khi đối tượng hết phạm vi sử dụng.

- Phương thức phá huỷ thuộc nhóm các phương thức xử lý.

- Có và chỉ có duy nhất một phương thức phá huỷ trong 1 lớp mà thôi.

- Từ khóa để tham khảo trong MSDN là "Finalize destructor".

Chú ý: Một phương thức phá hủy Finalize không nên gọi exceptions, bởi vì ứng dụng không thể quản lý được và chương trình có thể bị ngắt giữa chừng.

3.4 Thuộc tính - Property

Câu lệnh Property được sử dụng để dùng để gán giá trị cho một thuộc tính, một biến thành phần hoặc lấy giá trị của một thuộc tính, một biến thành phần đã được khai báo trong Module, Class hoặc Structure.

Cú pháp khai báo thuộc tính:

```
' Khai báo biến lưu giữ giá trị của thuộc tính

Private mthuoctinh As <Kiểu dữ liệu>
   [<Từ khoá>] Property Thuoc_tinh() As <Kiểu dữ liệu>

   ' Truy xuất giá trị của thuộc tính
   Get
        Return mthuoctinh
   End Get

   ' Gán trị cho thuộc tính
   Set (ByVal Value As <Kiểu dữ liệu>)
        mthuoctinh = Value
   End Set

End Property
```

- Một thuộc tính có thể có thủ tục Get (đọc), thủ tục Set (ghi), hoặc cả hai

- Nếu không có các từ khóa chỉ phạm vi Public, Protected, Private, Friend thì thuộc tính (Property) của thủ tục Get và thủ tục Set sẽ mặc định là Public.

- Các từ khóa khai báo trong cú pháp trên bao gồm

*** Default:**

Khai báo thuộc tính mặc định. Các thuộc tính này phải có tham số và có thể gán và truy xuất không cần chỉ ra tên thuộc tính.

Một thuộc tính chỉ có thể là thuộc tính mặc định nếu thỏa các điều kiện:

Mỗi Class chỉ được có một thuộc tính mặc định, phải kể đến cả các thuộc tính kế thừa.

Thuộc tính mặc định không được là Shared hay Private

Nếu một thuộc tính nạp chồng (Overloaded) là mặc định thì tất cả các thuộc tính cùng tên cũng phải khai báo mặc định.

Thuộc tính mặc định phải có ít nhất một tham số

Cú pháp:

```
Private mthuoctinh As <Kiểu_DL>
Default Public Property Thuoc_tinh(Index as Integer) As
<Kiểu_DL>
   Get
        Return mthuoctinh
   End Get

   Set (ByVal Value As <Kiểu dữ liệu>)
        …
        mthuoctinh = Value
   End Set
End Property
```

*** ReadOnly**

Cho biết thuộc tính chỉ được phép đọc không cho phép gán.

Cú pháp:

```
Private mthuoctinh As <Kiểu dữ liệu>
Public ReadOnly Property Thuoc_tinh() As <Kiểu dữ liệu>
   Get
        Return mthuoctinh
   End Get
End Property
```

*** WriteOnly**

Cho biết thuộc tính chỉ được phép gán không cho phép đọc.

Cú pháp:

```
Private mthuoctinh As <Kiểu dữ liệu>
Public WriteOnly Property Thuoctinh() As String
   Set (ByVal Value As String)
        mthuoctinh = Value
   End Set
```

```
End Property
```

*** Overloads**

Cho biết thuộc tính này nạp chồng một hoặc nhiều thuộc tính có cùng tên được định nghĩa trên lớp cơ sở. Danh sách tham số trong thuộc tính này phải khác với danh sách tham số của mỗi thuộc tính nạp chồng khác về số lượng, hoặc về các kiểu dữ liệu hoặc cả hai.

Chúng ta không cần phải dùng từ khóa Overloads khi tạo các thuộc tính nạp chồng trong một lớp. Nhưng nếu đã khai báo cho một thì phải khai báo cho tất cả.

Không được phép sử dụng cả hai từ khóa sau một lượt: Overloads và Shadows trong cùng một thuộc tính.

*** Overrides**

Cho biết thuộc tính ghi chồng một thuộc tính cùng tên của lớp cơ sở. Số lượng tham số, kiểu dữ liệu của tham số cũng như kiểu giá trị trả về phải khớp với của lớp cơ sở.

*** Overridable**

Cho biết thuộc tính này được phép ghi chồng trong lớp Con.

*** NotOverridable**

Cho biết thuộc tính không được phép ghi chồng trong lớp Con. Mặc nhiên, các thuộc tính là không được phép ghi chồng.

*** MustOverride**

Cho biết thuộc tính không được cài đặt trong lớp và phải được cài đặt ở lớp Con.

*** Shadows**

Cho biết thuộc tính che lấp một thuộc tính có tên tương tự, hoặc một tập hợp các thuộc tính nạp chồng của lớp cơ sở. Tham số và giá trị trả về không nhất thiết phải như trong thuộc tính bị che. Thuộc tính bị che không còn giá trị trong lớp che nó.

*** Shared**

Cho biết thuộc tính được chia sẻ - nghĩa là thuộc tính không gắn chặt với một thể hiện nào của lớp nhưng được sử dụng chung giữa các thể hiện của một lớp.

Ví dụ: Sử dụng câu lệnh Property dùng để gán giá trị và lấy giá trị đối với thuộc tính intMaSo trong class CHocSinh.

Khai báo định nghĩa phương thức.

```
Public Class CHocSinh
   Private intMaSo As Integer
   Private strHoTen As String
   Private dblToan As Double
   Private dblVan As Double
   Private dblTrungBinh As Double
   Private intMaKhoi As Integer

   Property _MaSo() As Integer
      Get
        Return intMaSo
      End Get

      Set(ByVal value As Integer)
        intMaSo = value
      End Set
   End Property
End Class
```

Hướng dẫn sử dụng

```
Module Module1
   Sub Main()
        Dim hs As New ChocSinh
        hs._MaSo = 5
        Dim a As Integer = hs._MaSo
        ...
   End Sub
End Module
```

3.5 Khai báo sự kiện (Event)

Sự kiện là thông điệp do một đối tượng phát sinh cho biết một hành động đang xảy ra. Hành động có thể do sự tương tác của người dùng, do chương trình khác kích hoạt…Đối tượng kích hoạt biến cố được gọi là đối tượng gửi biến cố. Đối tượng bắt sự kiện và đáp ứng lại gọi là đối tượng nhận sự kiện. Đối tượng gửi không biết đối tượng nhận sự kiện do nó kích hoạt, nhưng giữa đối tượng gửi và đối tượng nhận có một đối tượng trung gian. Trong .NET Framework có một kiểu đặc biệt thích hợp cho chức năng của vai trò trung gian này là Delegate (ủy quyền).

Chức năng của sự kiện được xác định từ ba yếu tố liên quan: một đối tượng cung cấp dữ liệu sự kiện (event data), một event delegate và một đối tượng kích

hoạt sự kiện (sender). .NET Framework có một qui ước về việc đặt tên các lớp và các phương thức liên quan đến sự kiện như sau:

3.5.1 Phát sinh sự kiện

Để một class phát sinh sự kiện EventName cần có các yếu tố sau:

Một tham số sự kiện chứa các dữ liệu có tên EventNameEventArgs phát sinh từ lớp System.EventArgs

Một thành phần xử lý sự kiện có tên EventNameEventHandler. Đây là một thủ tục sẽ được gọi khi sự kiện xảy ra. Chúng ta có thể sử dụng bất kỳ thủ tục hợp lệ nào làm thành phần xử lý sự kiện nhưng không được là một hàm.

Một đối tượng phát sinh sự kiện. Đối tượng này phải cung cấp :

- Một khai báo sự kiện

```
[<Từ khoá>] Event EventName As EventNameEventHandler
```

- Một phương thức tên OnEventName phát sinh sự kiện

- Các đối tượng event delegate và đối tượng dữ liệu sự kiện (EventArgs) có thể phát sinh từ những lớp tương ứng có sẵn trong .NET.

Các từ khoá khai báo sự kiện có thể là:

Tên	**Mô tả**
Public	Sử dụng được ở mọi nơi. Sự kiện không có từ khóa mặc nhiên là Public.
Private	Chỉ truy xuất trong phạm vi khai báo.
Protected	Chỉ truy xuất trong phạm vi Class và SubClass.
Friend	Chỉ truy xuất trong phạm vi Project.
Protected Friend	Chỉ truy xuất trong phạm vi của Protected và Friend
Shadows	Cho biết sự kiện che mờ một thành phần có tên tương tự trong lớp cơ sở. Chúng ta có thể che mờ một thành phần loại này bằng một thành phần loại khác. Thành phần bị che mờ sẽ không còn tác dụng trong lớp kế thừa che mờ nó.

Các sự kiện trong VB.NET không hỗ trợ đúng nguyên tắc kế thừa. Một sự kiện khai báo trong Class nào chỉ được phép gọi phát sinh sự kiện (RaiseEvent) chỉ trong lớp đó mà thôi, không được gọi phát sinh kể cả trong các lớp kế thừa.

Cú pháp để gọi phát sinh sự kiện như sau

```
RaiseEvent <tên sự kiện>()
```

Ví dụ chúng ta có lớp Con_Nguoi và thuộc tính Chieu_Cao, khi chiều cao thay đổi sẽ phát sinh sự kiện Chieu_Cao_Thay_doi như sau

```
Public Class Con_Nguoi
   Private Cao As Single
   Public Event Chieu_Cao_Thay_doi()
   Public Property Chieu_Cao() As Single
     Get
        Return Cao
     End Get

    Set(ByVal Value As Single)
        Cao = Value
        RaiseEvent Chieu_Cao_Thay_doi()
    End Set
   End Property
End Class
```

Để sử dụng các sự kiện trên một biến, chúng ta sử dụng cú pháp sau khi khai báo biến:

```
Private WithEvents <tên biến> As <tên Class>
```

3.5.2. Kết hợp sự kiện với xử lý sự kiện

Các sự kiện được kết hợp với xử lý sự kiện bằng các lệnh: Handles hoặc AddHandler.

Từ khóa WithEvents và Handles cung cấp cách khai báo các xử lý sự kiện. Các sự kiện do một đối tượng phát sinh với từ khóa WithEvents có thể được xử lý bằng bất kỳ thủ tục nào được chỉ ra trong mệnh đề Handles của sự kiện. Tuy mệnh đề Handles là cách thức chuẩn để kết hợp một sự kiện với một xử lý sự kiện, nhưng lại giới hạn số các biến cố kết hợp với cùng xử lý sự kiện khi biên dịch.

Các lệnh AddHandler và RemoveHandler uyển chuyển hơn trong việc kết hợp sự kiện với xử lý sự kiện. Chúng cho phép chúng ta linh hoạt kết hợp và ngắt rời các sự kiện với các xử lý sự kiện lúc thực thi và chúng không đòi hỏi chúng ta phải khai báo biến với từ khóa WithEvents.

3.6 Từ khóa Me, Mybase, MyClass

3.6.1. Me

Từ khóa Me được dùng khi chúng ta chỉ rõ muốn dùng các thành phần của chính thể hiện Class nơi viết lệnh chứ không phải thành phần nào khác.

Ví dụ:

```
Public Class Con_nguoi
   Private ten as String Public Sub Lam_viec()
```

```
        Dim ten as String ' biến cục bộ  của Sub được sử  dụng
        ten = "Hùng"
        ' biến cấp Class được dùng
        Me.ten = "Hùng"
   End Sub
End Class
```

3.6.2. Mybase

Từ khóa Mybase được dùng trong Class kế thừa, khi chúng ta muốn dùng các phương thức của chính Class cơ sở.

Ví dụ:

Lớp `lopCha` với thủ tục `Gioi_thieu` và tạo tiếp Class `lopCon` kế thừa từ `lopCha`

```
Public Class lopCha
   Public Overridable Sub Gioi_thieu()
        Console.Write("Tôi là thể  hiện của lopCha")
   End Sub
End Class
```

```
Public Class lopCon
   Inherits lopCha
    Public Overrides Sub Gioi_thieu()
        Console.Write ("Tôi là thể  hiện của lopCon")
        Mybase.Gioi_thieu()
   End Sub
End Class
```

Khi gọi phương thức `Gioi_thieu` của `lopCon`, chúng ta sẽ có hai thông báo: một của chính `lopCon` và một của `lopCha`

Chú ý: Từ khóa Mybase

- Chỉ được dùng để tham chiếu đến Class trung gian và các thành phần được kế thừa của nó.

- Không phải là một đối tượng, nên thông thể gán trị, dùng làm tham số hoặc dùng trong toán tử Is

- Không được sử dụng trong các Standard Module.

3.6.3 MyClass

Từ khóa MyClass cho phép chúng ta gọi các phương thức Overridable của Class, dẫu các Class kế thừa đã có các phương thức Overrides tương ứng.

Ví dụ:

Lớp `lopCha` với thủ tục `Gioi_thieu` và tạo tiếp Class `lopCon` kế thừa từ `lopCha`

```
Public Class lopCha
   Public Sub Chao()
        Gioithieu()
   End Sub

   Public Overridable Sub Gioi_thieu()
        Console.Write("Tôi là thể  hiện của lopCha")
   End Sub
End Class
```

Và một Class `lopCon` kế thừa từ `lopCha` có thủ tục `Gioithieu` ghi đè, và kế thừa thủ tục `Chao` của `lopCha`

```
Public Class lopCon
  Inherits lopCha
  Public Overrides Sub Gioi_thieu()
        Console.Write(“Tôi là thể  hiện của lopCon”)
  End Sub
End Class
```

Với đoạn lệnh sau:

```
Dim a as New lopCon()
a.Chao() ' Tôi là thể  hiện của lopCon
```

Nhưng nếu thủ tục `Chao` của `lopCha` như sau:

```
Public Sub Chao()
        MyClass.Gioi_thieu()
End Sub
```

Thì :

```
a.Chao() ' Tôi là thể  hiện của lopCha
```

NỘI DUNG PHẦN THẢO LUẬN

1. Tạo lớp mới, tạo namespace mới.

2. Xây dựng hàm tạo, hàm hủy, định nghĩa toán tử cho lớp.

3. Kế thừa các lớp cơ sở

TÓM TẮT NỘI DUNG CỐT LÕI

Trong chương này sinh viên cần chú ý đến các nội dung sau:

• Khai báo lớp

• Khai báo phương thức

• Khái niệm sự kiện

• Sử dụng các từ khóa Me, Mybase, MyClass

BÀI TẬP ỨNG DỤNG

Hãy làm các bài tập dưới đây bằng phương pháp lập trình hướng đối tượng với VB.NET

3.1 Viết chương trình nhập vào một phân số. Hãy cho biết phân số đó là phân số âm hay dương hay bằng không.

3.2 Viết chương trình nhập tọa độ hai điểm trong không gian. Tính khoảng cách giữa chúng và xuất kết quả.

3.3 Viết chương trình nhập vào 2 phân số. Tìm phân số lớn nhất và kết quả.

3.4 Viết chương trình nhập vào 2 số phức. Tính tổng, hiệu, tích và xuất kết quả.

3.5 Viết chương trình nhập toạ độ 3 đỉnh A,B,C của 1 tam giác trong mặt phẳng Oxy. Tính chu vi của tam giác và xuất ra kết quả.

3.6 Viết chương trình nhập vào một ngày. Tìm ngày kế tiếp và xuất kết quả.

Chương 4
FORM VÀ CÁC ĐIỀU KHIỂN CONTROL

MỤC TIÊU CỦA CHƯƠNG

- Về kiến thức

+ Cung cấp cho sinh viên các khái niệm về form, điều khiển, thuộc tính, phương thức, biến cố;

+ Trang bị cho sinh viên kiến thức và cách làm việc với một số điều khiển thông dụng như form, label, textbox, button, listbox, combo box.

+ Giúp sinh viên xây dựng code cho các sự kiện với chuột và bàn phím.

- Về thái độ:

+ Giúp sinh viên tiếp cận với lập trình trực quan và nắm được các yêu cầu khi xây dựng giao diện của một chương trình.

- Về kỹ năng

+ Sinh viên tạo được các ứng dụng Windows Form bằng ngôn ngữ Visual Basic.NET.

+ Sinh viên tự thiết kế các biểu mẫu phục vụ cho việc giải các bài toán cơ bản.

+ Kết hợp các cấu trúc lệnh đã học để viết các đoạn code xử lý sự kiện chương trình.

+ Biết cách quản lý form và project của chương trình.

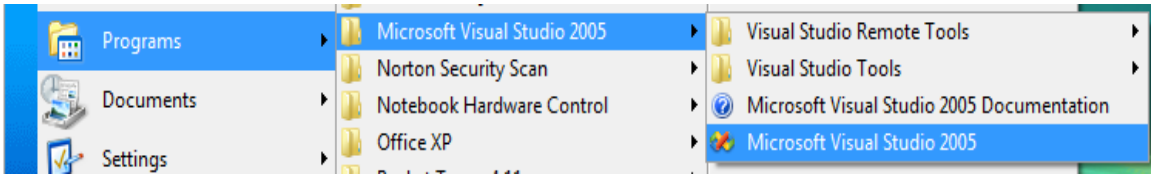
+ Biết cách chạy chương trình và cách phát hiện các lỗi cú pháp đơn giản.

NỘI DUNG BÀI GIẢNG LÝ THUYẾT

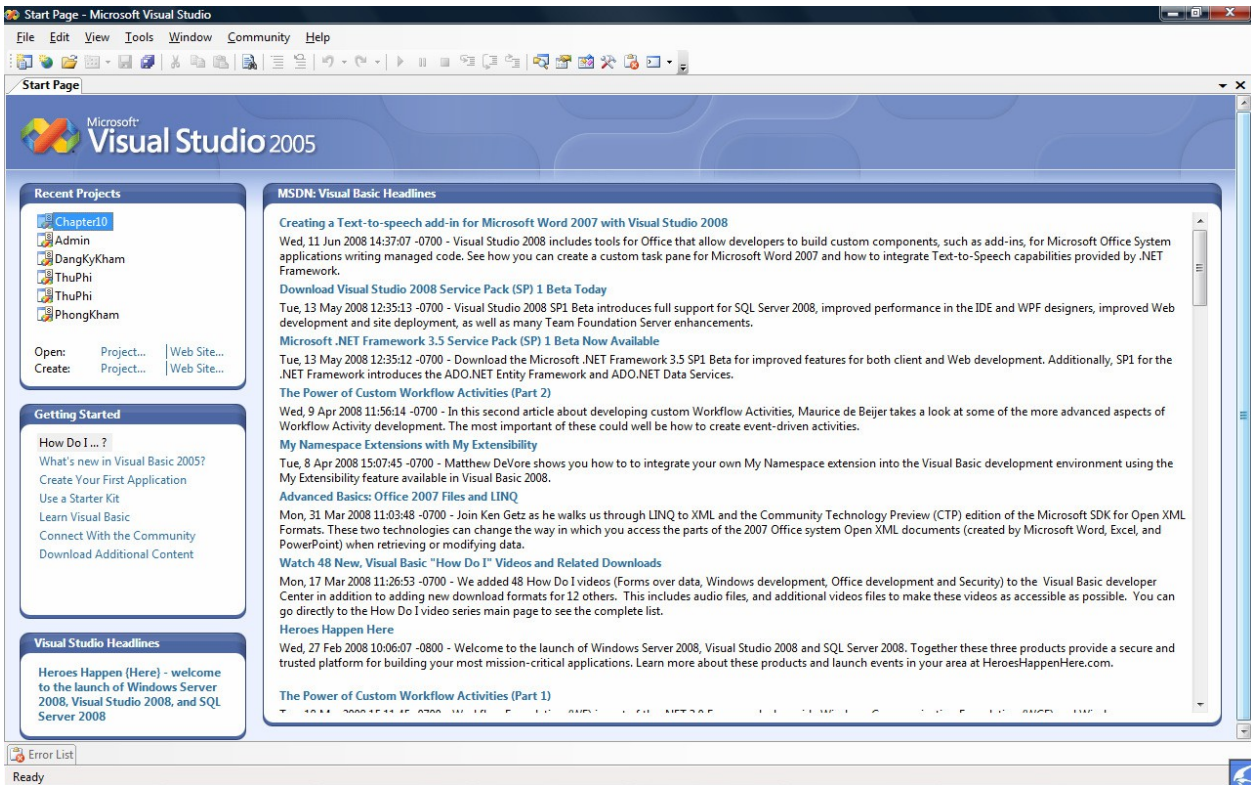
4.1 Tạo ứng dụng Windows Form

4.1.1 Mở một project mới

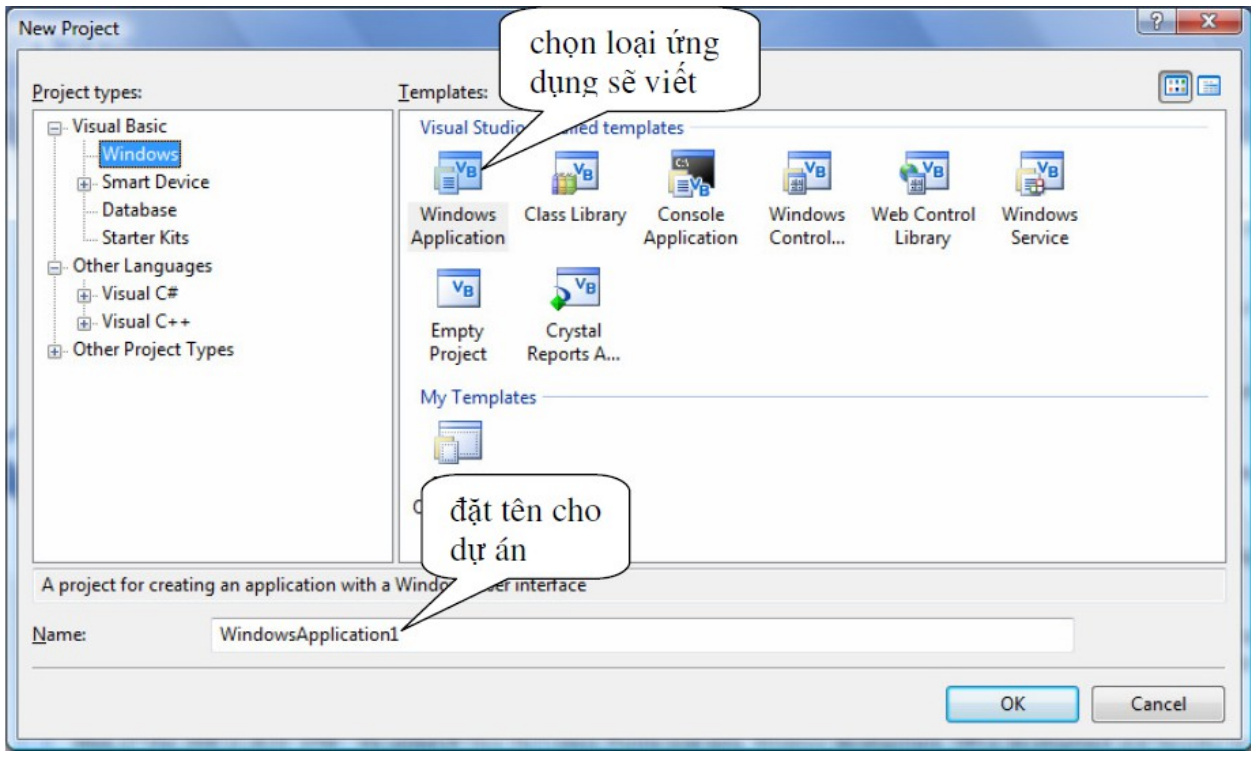
- Vào Start/Programs/Microsoft Visual Studio 2005/Microsoft Visual Studio 2005



- Sau khi khởi động nếu là lần đầu tiên thì chương trình sẽ hỏi bạn sử dụng ngôn ngữ gì (chọn Visual Basic). Nếu là lần thứ hai trở đi thì xuất hiện giao diện như hình dưới.

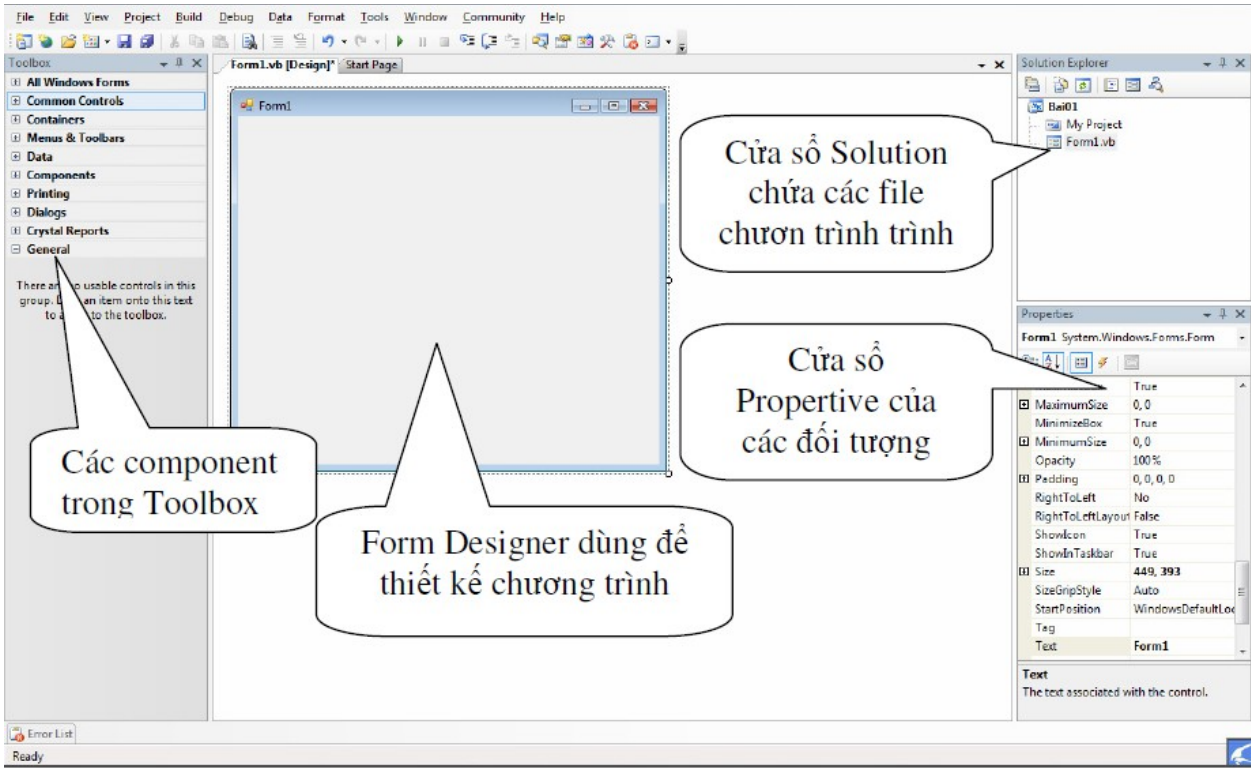


- Vào New/Project. Xuất hiện hộp thoại sau:



Chọn Visual Basic/Windows, chọn Windows Application trong cửa sổ bên phải sau đó đặt tên cho project mới (tên mặc định là WindowsApplication1), OK.

Màn hình làm việc sẽ xuất hiện như cửa sổ dưới đây:



Hộp công cụ (toolbox): chứa các biểu tượng tương ứng với những đối tượng điều khiển chuẩn bao gồm nhãn (label), hộp văn bản (textbox), nút lệnh (button),... Để bật tắt hộp công cụ ta vào Menu View, chọn Toolbox (hoặc nhấn Ctrl+Alt+X).

Thực đơn (Menu Bar): cho phép truy cập đến hầu hết các lệnh dùng điều khiển môi trường phát triển. Menu và các lệnh trên menu làm việc tương ứng như hầu hết các chương trình Windows thông thường khác.

Thiết kế chương trình (Form Designer): đây là đối tượng xây dựng và thiết kế màn hình giao tiếp của ứng dụng. Khi vừa tạo mới, màn hình giao tiếp có tên mặc định là Form1.vb và không chứa đối tượng điều khiển nào cả, nhiệm vụ của người lập trình là thiết kế các đối tượng điều khiển trên màn hình giao tiếp và sử dụng các dòng lệnh để xử lý các sự cố kiện thích hợp cho các đối tượng điều khiển đó.

Solution Explorer: Quản lý các ứng dụng hiển thị các màn hình giao tiếp (Forms), các reports và các module,... hiện có của ứng dụng. Ngoài ra, còn cho phép người lập trình thực hiện nhanh những thao tác như thêm, xóa các đối tượng này khỏi ứng dụng.

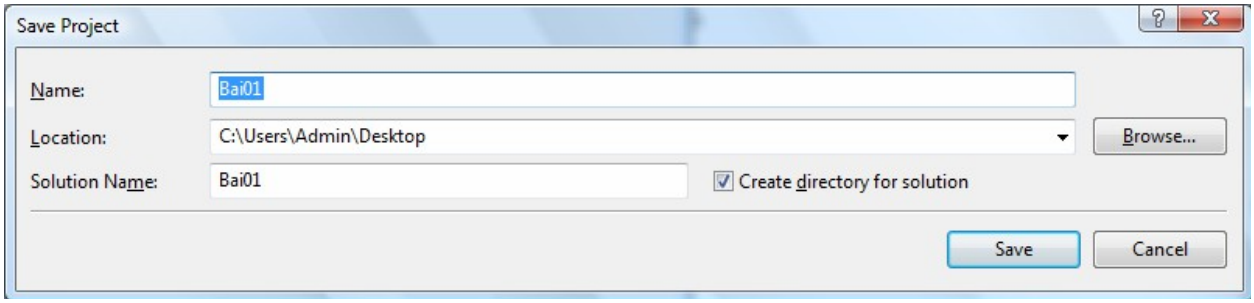
Cửa sổ thuộc tính (Properties): Cho phép định thuộc tính ban đầu cho các đối tượng bao gồm màn hình giao tiếp (Forms) và các điều khiển (Control).

Cửa sổ lệnh (Code Window): đây là cửa sổ cho phép khai báo các dòng lệnh xử lý biến cố cho màn hình giao tiếp và các đối tượng trên màn hình giao tiếp. Mặc nhiên cửa sổ này không được hiển thị, người lập trình có thể nhấn nút chuột phải lên màn hình giao tiếp và chọn **View Code** để hiển thị khi cần.

4.1.2 Thao tác với project

a. Lưu chương trình

- Chọn File/Save All



b. Mở một project đã có:

- Khởi động Visual Studio 2005
- Vào menu File / Open Project
- Chọn Project muốn mở trong hộp thoại Open , chọn OK.

c. Chạy chương trình

- Nhấn chuột vào nút Start hoặc nhấn F5. Chương trình sẽ tự động tạo ra file thực thi .exe trong thư mục Debug chứa trong dự án đã tạo.

Ta cũng có thể dịch thử chương trình vừa tạo bằng cách vào Build/Build và chọn tên dự án để dịch thử chương trình vừa tạo.

Trong trường hợp chương trình còn lỗi thì sẽ xuất hiện thông báo lỗi và chỉ ra dòng lệnh có chứa lỗi.

4.2 Biểu mẫu - Form

Các ứng dụng Windows Form giao tiếp với người dùng thông qua các biểu mẫu (hay còn gọi là cửa sổ, xuất phát từ chữ Form hay Windows); các điều khiển (Control) được đặt lên bên trên giúp cho biểu mẫu thực hiện được công việc đó. Biểu mẫu là các cửa sổ được lập trình nhằm hiển thị dữ liệu và nhận thông tin từ phía người dùng.

Mỗi biểu mẫu và điều khiển đều có các thuộc tính, phương thức và sự kiện gắn liền với chúng. Thuộc tính bao gồm các đặc trưng của một điều khiển; chúng góp phần tạo nên dáng vẻ riêng của điều khiển đó. Các thuộc tính có thể được thiết lập ở thời điểm thiết kế hoặc có thể thiết lập ở thời điểm chạy chương trình bằng cách xây dựng các câu lệnh (code) thích hợp. Phương thức là các tác vụ mà điều khiển có thể thực thi; thông thường là các hàm và thủ tục. Sự kiện hay biến cố là hành động của người dùng tác động lên điều khiển ở thời điểm chạy chương trình. Một ứng dụng Windows thường được thực hiện nhờ vào việc đáp ứng lại các sự kiện của người dùng.

4.2.1 Thuộc tính, biến cố và phương thức của biểu mẫu

a. Thuộc tính của biểu mẫu

Tên	**Ý nghĩa**
Name	Tên để gọi trong chương trình của form. Không sử dụng dấu cách và các ký tự đặc biệt
AccepButton	Tên nút lệnh sẽ được kích hoạt khi người sử dụng gõ Enter
Autosize	Tự động thay đổi kích cỡ
BackColor	Màu nền form
BackGroundImage	Ảnh nền form
Cancel Button	Tên nút lệnh sẽ được kích hoạt khi người sử dụng gõ ESC
ContextMenuStript	Tên menustrip xuất hiện khi người sử dụng nhấn phải chuột vào đối tượng
ControlBox	True/False - Hiển thị hoặc không hiển thị hộp Control của form
Cussor	Thay đổi hình dạng hiển thị của con trỏ chuột khi người sử dụng di chuyển chuột trên phạm vi điều khiển
Enabled	Ngầm định là True, cho phép điều khiển đáp ứng các biến cố
Font	Tên font
ForeColor	Màu font
FormBorderStyle	Kiểu đường viền và tiêu đề Form. Nếu là fixed thì không cho phép thay đổi kích cỡ
HelpButton	Ngầm định là False - không hiển thị nút Help trên form
Icon	Biểu tượng trên thanh tiêu đề
IsMidiContainer	Ngầm định là False - Form không phải là form MDI
KeyPreview	Ngầm định là False - Không cho phép điều khiển các sự kiện bàn phím
Locked	Ngầm định là False - Không cho phép di chuyển hoặc thay đổi kích cỡ điều khiển khi đang chạy chương trình
MaximizeBox	Có hiển thị nút Max của form hay không
MinimizeBox	Có hiển thị nút Min của form hay không

Tên	Ý nghĩa
StartPosition	Vị trí của form khi được kích hoạt
ShowIcon	Cho phép hiện/ẩn Icon
Text	Nội dung thanh tiêu đề
Top/Left	Vị trí form
TopMost	Form luôn nằm trên các form khác kể cả khi không nhận được Focus
Height/Width	Chiều cao/chiều rộng của form
Windows State	Kích cỡ form khi được kích hoạt

b. Biến cố trên form

Tên biến cố	Đáp ứng khi
KeyDown	Khi nhấn một phím bất kỳ trên bàn phím
KeyPress	Khi nhấn một phím nhưng chưa kết thúc
KeyUp	Khi kết thúc thao tác nhấn một phím
Load	Khi nạp form
LostFocus	Khi mất focus
MouseClick	Khi nhấn chuột
MouseDoubleClick	Nhấn đúp chuột
MouseMove	Di chuyển chuột
Move	Di chuyển form
Resize	Khi thay đổi kích cỡ form

c. Một số phương thức của form

- Hiển thị một form bất kỳ:

```
Tên_form.Show()
```

- Đóng form hiện thời

```
Me.Close()
```

- Đóng một form bất kỳ

```
Tên_form.Close()
```

4.2.2 Thao tác với form trong một ứng dụng

a. Thêm một form mới

- Nhấn phải chuột vào tên ứng dụng trong cửa sổ Solution Explorer. Nếu cửa sổ này không hiển thị thì chọn lệnh View/Solution Explorer

- Chọn Add, Windows Form để thêm một biểu mẫu vào ứng dụng.

b. Xóa một form

- Nhấn phải chuột vào tên form trong cửa sổ Solution Explorer

- Chọn lệnh Delete

c. Thay đổi form khởi động

Ngầm định, form được tạo ra đầu tiên (form1) sẽ là form được hiển thị khi ứng dụng kích hoạt. Tuy nhiên ta có thể chọn bất kỳ một form nào đảm nhiệm công việc này. Thực hiện bằng cách nhấn phải chuột vào tên ứng dụng trong cửa sổ Solution Explorer, chọn Properties. Trong tab Application, chọn tên form trong mục Startup Form.

4.2.3 Viết code cho các biến cố trên form

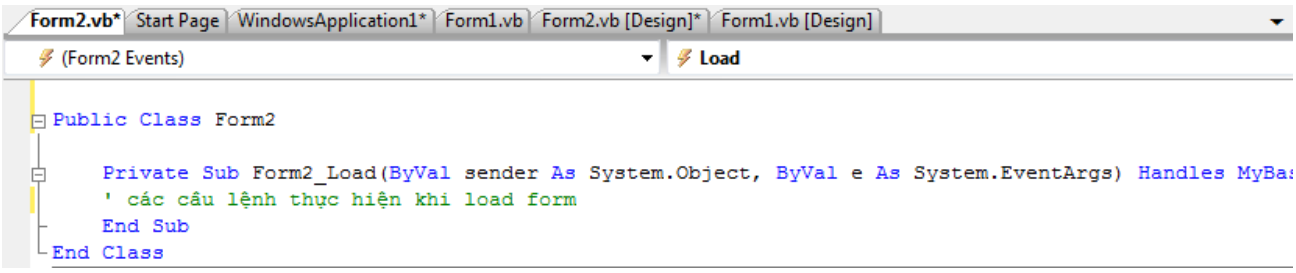
Thao tác chung để viết code cho các biến cố trên form hoặc trên một điều khiển bất kỳ được thực hiện thông qua 3 bước.

i. Gọi cửa sổ Code

ii. Chọn tên điều khiển và tên sự kiện

iii. Nhập đoạn code cần thiết.

Có nhiều cách để gọi cửa sổ Code. Ta có thể gọi bằng cách nhấn F7 hoặc nhấn phải chuột vào biểu mẫu và chọn View Code. Thao tác nhắp đúp vào biểu mẫu sẽ mở ra cửa sổ code của biến cố Form_Load.



Sau khi đã mở cửa sổ Code, ta chọn tên biểu mẫu (hoặc điều khiển) trên hộp bên trái. Các biến cố (sự kiện) được chọn trong mục bên phải.

Các câu lệnh cần thực hiện khi sự kiện được đáp ứng được trình bày dưới dạng một sub - procedure.

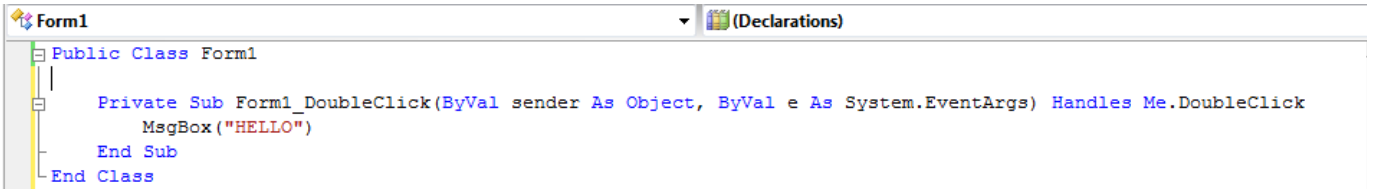
Ví dụ: Thiết kế form1 với tên form là Hello, tiêu đề là First Program, khi khởi động nằm chính giữa màn hình, con trỏ có hình bàn tay khi ở trên form và khi nhấn

đúp vào form xuất hiện dòng chữ Hello Programmer.

- Bước 1: Khởi động Visual Studio 2005
- Bước 2: Tạo ứng dụng mới lấy tên Hello
- Bước 3: Thiết lập các thuộc tính của form 1:
    + Name: hello
    + Curssor: chọn hình bàn tay Hand
    + StartPosition: Center Screen
    + Text: First Program
- Bước 4: Viết code:
    + Gọi cửa sổ code
    + Gọi biến cố Double Click
    + Nhập dòng mã:

```
Msgbox ("HELLO")
```

Đoạn code đầy đủ cho biến cố Double Click của form sẽ được hiển thị như sau:



Kết quả khi chạy chương trình ta thấy form nằm chính giữa màn hình và được thể hiện như trong hình 4.1 a. Khi nhắp đúp chuột vào một vị trí bất kỳ tại vùng làm việc của form ta thấy dòng chữ Hello xuất hiện dưới dạng một hộp thoại như trong hình 4.1b.

4.2.4 Form MDI

Trong các ứng dụng Windows ví dụ như Word chúng ta thường thấy các cửa sổ văn bản xuất hiện bên trong cửa sổ Word và không bao giờ vượt ra khỏi phạm vi của cửa sổ ứng dụng này. Đấy là một minh họa của form MDI Parent gọi tắt là MDI.

Form MDI là một loại form đặc biệt thường được sử dụng làm cửa sổ chính cho ứng dụng. Trong một ứng dụng chỉ có một cửa sổ MDI. Các cửa sổ con bên trong MDI chỉ có thể di chuyển trong phạm vi của cửa sổ MDI chứa nó.

Khi cửa sổ con bên trong MDI được maximize thì kích thước cũng chỉ bằng vùng làm việc của MDI. Lúc này tiêu đề của cửa sổ con được ghép với tiêu đề của cửa sổ MDI và nếu có menu thì menu của sổ con sẽ thay thế menu của cửa sổ MDI.

Khi cửa sổ con được minimize, biểu tượng của cửa sổ con nó nằm trong cửa sổ MDI.

Form MDI được thiết lập bằng hai cách. Cách thứ nhất ta có thể set thuộc tính IsMDIContainer của một form bất kỳ về True. Cách thứ 2 ta có thể bổ sung vào projetc một form MDI bằng cách nhấn phải chuột vào project, chọn Add, New Item và chọn MDI Parent form. Ngầm định các form MDI có nền màu xám đậm.

Sau khi đã thiết lập form MDI ta có thể chỉ định một form bất kỳ làm MDI con bằng cách set thuộc tính MDIparent của nó là tên của form MDI vừa thành lập.

Ngoài các thuộc tính như form SDI thông thường, form MDI có thêm thuộc tính ActivateForm xác định form đang được activate bên trong form MDI.

Ví dụ: Gán thuộc tính màu nền của form con đang được active là màu đỏ.

```
ActiveForm.Backcolor = VBRed
```

4.3 Label, Textbox, Button

4.3.1 Nhãn - Label

Nhãn là điều khiển dạng đồ họa cho phép người sử dụng hiển thị chuỗi ký tự trên biểu mẫu nhưng không thể thay đổi chuỗi ký tự đó một cách trực tiếp.

Thuộc tính của nhãn:

Tên	**Ý nghĩa**
Name	Tên của nhãn
BackColor	Màu nền của nhãn
Border Style	Kiểu đường viền của nhãn (none, fixsingle, fix3D)

Tên	Ý nghĩa
Image	Hình của nhãn
TabIndex	Số thứ tự của nhãn trên biểu mẫu (khi dùng phím Tab duyệt qua các điều khiển trên biểu mẫu).
Text	Chuỗi ký tự hiển thị trên nhãn. Mặc nhiên là "label.."
TextAlign	Căn lề của chuỗi ký tự
Visible	Ngầm định là True - cho phép hiển thị nhãn trên biểu mẫu

Phương thức của nhãn:

Location: di chuyển nhãn đến tọa độ X,Y

```
Location = new point (X,Y)
```

Biến cố tác động trên nhãn:

TextChanged: Xảy ra khi nhãn thay đổi giá trị (chuỗi text)

Click: Xảy ra khi người sử dụng nhắp chuột trên phạm vi của nhãn

Double Click: Xảy ra khi người sử dụng nhắp đúp chuột trên phạm vi của nhãn

4.3.2 Ô nhập liệu - Textbox

Ô nhập liệu là một điều khiển cho phép nhận thông tin do người dùng nhập vào. Đối với ô nhập liệu ta cũng có thể dùng để hiển thị thông tin, thông tin này được đưa vào tại thời điểm thiết kế hay thậm chí ở thời điểm thực thi ứng dụng.

Thuộc tính của textbox

Tên	Ý nghĩa
Name	Tên textbox
BackColor	Màu nền
BorderStyle	Xác định kiểu đường viền quanh textbox
CharacterCasing	Normal (mặc định); Upper (tự động đổi thành chữ hóa); Lower (tự động đổi thành chữ thường).
Font	Kiểu chữ
ForeColor	Màu chữ
Maxlength	Số ký tự tối đa được phép nhập vào textbox
Multiline	Ngầm định là False - không cho phép xuống dòng.
PasswordChar	Ký tự dùng để hiển thị thay cho những ký tự được nhập vào

Tên	Ý nghĩa
	textbox.
ReadOnly	Nếu là True - chỉ cho phép xem văn bản mà không được chỉnh sửa
Scrollbars	None (ngầm định), Horizontal (thanh cuộn ngang); Vertical (thanh cuộn dọc); Both (cả hai thanh cuộn).
Text	Dòng văn bản hiển thị trên textbox
Wordwrap	Nếu là True cho phép văn bản tự động xuống dòng

Phương thức:

- `Location`: di chuyển khung đến tọa độ X,Y
- `Focus`: thiết lập Focus cho textbox (chuyển con trỏ đến textbox để chờ nhập liệu)
- `Select(bắt đầu, số ký tự)`: chọn khối văn bản trong textbox từ vị trí `bắt đầu` với chiều dài chuỗi được chọn là `số ký tự`
- `SelectionLength`: độ dài khối văn bản được chọn
- `SelectionStart`: vị trí đầu khối văn bản được chọn
- `SelectionText`: truy xuất phần văn bản được chọn

Sự kiện:

- `KeyPress`: xảy ra khi người sử dụng nhấn một phím bất kỳ. Sự kiện này cho một mã ASCII (0-255) của phím vừa được nhấn. Trong ví dụ dưới đây Textbox chỉ nhận biết các phím là số (0-9) mà không nhận biết các phím khác:

```
Private Sub TextBox1_KeyPress(ByVal sender As Object, ByVal e
As _
System.Windows.Forms.KeyPressEventArgs) Handles
TextBox1.KeyPress
        If (e.KeyChar < ChrW(48) Or e.KeyChar > ChrW(57)) Then
            e.KeyChar = ChrW(0)
        End If
End Sub
```

- `KeyDown, KeyUp`: Mỗi sự kiện KeyPress để lại cho ta một cặp sự kiện KeyDown/KeyUp. Sự kiện này cho phép ta nhận biết được các phím đặc biệt trên bàn phím. Ví dụ dưới đây sẽ hiển thị tên các phím chức năng mà người sử dụng chương trình nhấn vào:

```
Private Sub TextBox1_KeyDown(ByVal sender As Object, ByVal e As
_
System.Windows.Forms.KeyEventArgs) Handles TextBox1.KeyDown
   If e.KeyCode >= Keys.F1 And e.KeyCode <= Keys.F12 Then
        MsgBox("Bạn vừa nhấn phím chức năng: F" & e.KeyCode -
111)
   End If
End Sub
```

4.3.3 Nút lệnh - Command Button

Nút lệnh là một điều khiển dùng để bắt đầu, ngắt hoặc kết thúc một quá trình. Người sử dụng luôn có thể chọn một nút lệnh nào đó bằng cách nhấn chuột trên nút lệnh đó. Hoặc có thể nhấn Enter để chọn nút lệnh khi nút đó đang có Focus.

Thuộc tính của nút

Tên	**Ý nghĩa**
Name	Tên nút
BackColor	Màu nền
BackgroundImage	Hình nền của nút
Enabled	Ngầm định là True - Nếu là False nút sẽ bị mờ đi và không có hiệu lực.
Font	Kiểu chữ
ForeColor	Màu chữ
Image	Hình của nút
ImageAlign	Kiểu căn lề của image
Locked	Khóa hay không khóa điều khiển button
Text	Dòng văn bản hiển thị trên nút
TextAlign	Căn lề chuỗi text trên nút
Visible	Mặc định là True cho phép hiển thị nút trên biểu mẫu

Phương thức

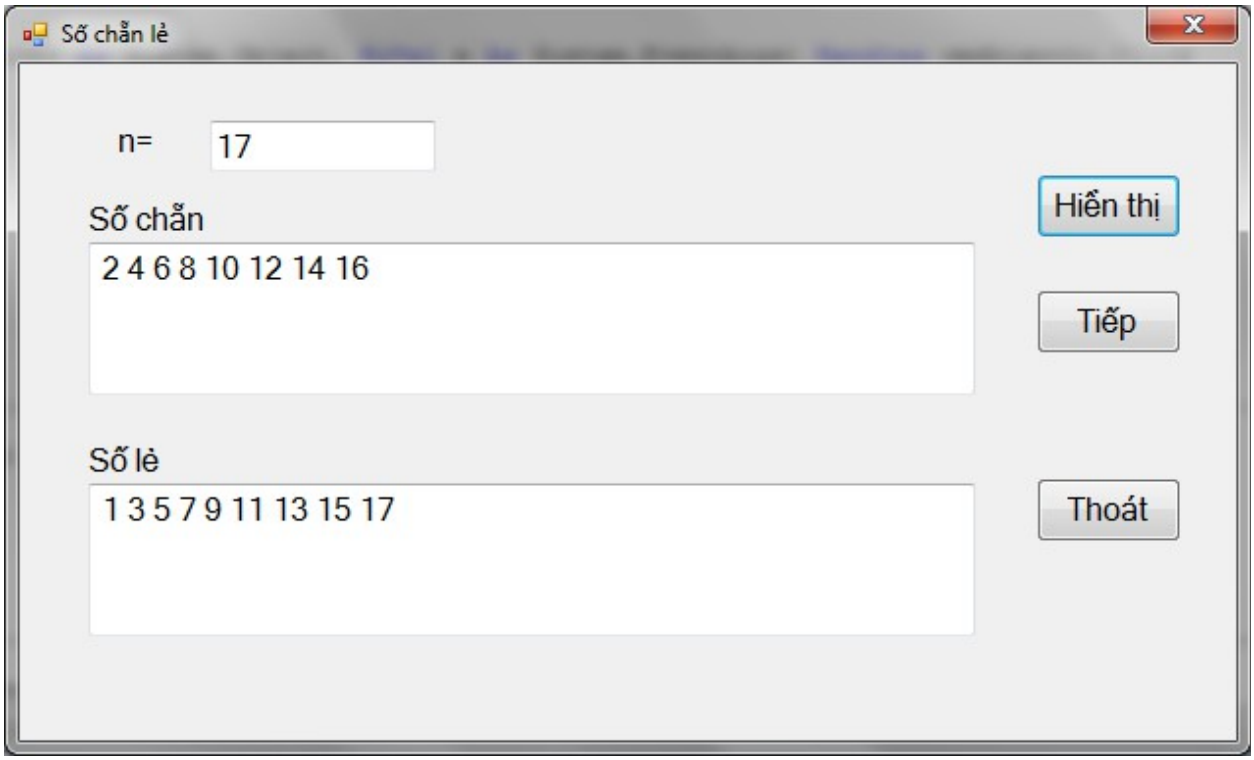
- `Location`: di chuyển nút đến tọa độ X,Y: location = new point (X,Y)

Sự kiện

- `Click`: Xuất hiện khi người sử dụng nhấn chuột vào nút lệnh hoặc Enter khi nút lệnh đang có Focus

Ví dụ: Nhập một số n từ bàn phím và in ra các số lẻ trong phạm vi n, các số chẵn trong phạm vi n

Thiết kế form như hình 4.2 với 3 nút lệnh Hiển thị, Tiếp và Thoát. Nút Hiển thị sẽ liệt kê các dãy số chẵn lẻ trong textbox tương ứng. Nút Tiếp cho phép xóa trắng các textbox nhập liệu, kết quả và chuyển con trỏ đến hộp n để nhập giá trị mới. Nút Thoát sẽ kết thúc chương trình.



Hình 4.2

1. Thiết kế

Điều khiển	**Thuộc tính**	**Giá trị**
Form:	Name	Frm1
	FormBorderStyle	FixedSingle
	MaximizeBox	False
	MinimizeBox	False
	StartPosition	CenterScreen
Label 1	Text	n=
Label 2	Text	Số chẵn

Điều khiển	Thuộc tính	Giá trị
Label 3	Text	Số lẻ
Textbox 1	Name	txtn
Textbox 2	Name	txtsc
Textbox 3	Name	txtsl
Button 1	Name	cmdhienthi
	Text	Hiển thị
Button 2	Name	cmdtiep
	Text	Tiếp
Button 3	Name	cmdthoat
	Text	Thoát

2. Viết Code

Biến cố Click của cmdhienthi

```
Dim i%, n%
n = CInt(txtn.Text)
For i = 1 To n
     If i Mod 2 = 1 Then
        txtsl.Text &= " " & i
     Else
        txtsc.Text &= " " & i
     End If
Next
```

Biến cố Click của cmdTiep

```
txtn.Clear()
txtsc.Text = ""
txtsl.Clear()
txtn.Focus()
```

Biến cố Click của cmdThoat

```
end
```

4.4 Check box, Radio Button

4.4.1 Hộp kiểm - Check box

Hộp kiểm hay còn gọi là hộp đánh dấu (Check box) là một điều khiển được hiển thị dưới dạng một ô vuông. Ô vuông này hiển thị dấu nếu như được chọn và để trắng nếu ô không được chọn. Điều khiển Check box được dùng khi muốn nhận thông tin từ người sử dụng theo kiểu Yes/No hoặc True/False. Ta cũng có thể gom nhiều điều khiển lại một nhóm (dùng công cụ Group box) để hiển thị nhiều khả năng lựa chọn. Khi một check box được chọn thì giá trị của nó là 1; ngược lại là 0.

Thuộc tính của check box:

- `Name`: tên của điều khiển
- `Checked`: giá trị hiện thời trên check box. Có thể nhận các giá trị Checked (True), Unchecked (False).
- `Text`: nội dung dòng chữ xuất hiện cạnh nút.

Sự kiện trên check box

- `Click:` xảy ra khi người sử dụng nhắp chuột trên check box

4.4.2 Nút chọn - Radio button

Công dụng của Radio button cũng tương tự như Check box. Điểm khác nhau chủ yếu giữa hai loại điều khiển này đó là trong cùng một nhóm (được tạo bởi group box hay picture box) tại mỗi thời điểm ta chỉ có thể chọn một radio button nhưng có thể đánh dấu chọn nhiều check box.

Thuộc tính của radio button:

- `Name`: tên của điều khiển
- `Checked`: giá trị hiện thời trên radio button. Có thể nhận các giá trị True hoặc False.
- `Text`: nội dung dòng chữ xuất hiện cạnh nút.

Sự kiện trên radio button

- `Click:` xảy ra khi người sử dụng nhắp chuột trên điều khiển

Hình 4.3 dưới đây minh hoạ các điều khiển radio button và check box

Hình 4.3

4.4.3 Ví dụ về form sử dụng điều khiển check box và radio button

Yêu cầu: Thiết kế form như hình 4.4. Form cho phép người sử dụng chọn một trong 3 màu chữ xanh (Blue), đỏ (Red), đen (Black) và chọn kiểu chữ là Đậm (Bold)

hoặc Nghiêng (Italic) hoặc gạch chân (Underline). Các kiểu chữ có thể kết hợp với nhau.

Hình 4.4 Form với các điều khiển check box và Radio button

1. Thiết kế:

Form:	+ Text: Điều khiển Check box và Radio button
Groupbox 1:	+ Text: Màu chữ
Groupbox 2	+ Text: Kiểu chữ
Radio button 1	+ Name: rdbxanh
	+ Checked: True
	+ Text: Xanh
Radio button 2	+ Name: rdbdo
	+ Text: Đỏ
Radio button 3	+ Name: rdbden
	+ Text: Đen
Check box 1	+ Name: chkdam
	+ Text: Đậm
Check box 2	+ Name: chknghieng
	+ Text: Nghiêng
Check box 3	+ Name: chkgc

	+ Text: Gạch chân
Textbox	+ Name: txtvbmau
	+ Font: Times New Roman, size 14
	+ Text: This is a sample text

2. Code:

Để điều khiển màu chữ chúng ta thiết lập giá trị màu cho thuộc tính Fore color của textbox txtvbmau. Các màu này lấy trong lớp Color.

Ví dụ: điều khiển màu chữ thành Blue khi nút Xanh được chọn (sự kiện Checkedchange của rdbxanh

```
txtvbmau.ForeColor = Color.Blue
```

Muốn điều khiển kiểu chữ (Font Style) ta phải truy cập vào phương thức New của lớp Font. Lớp này nằm trong namespace System.Drawing. Để sử dụng Namespace này ta bổ sung thêm câu lệnh Imports vào đầu cửa sổ Code (bên trên dòng lệnh Public Class form1).

```
Imports System.Drawing
```

Thay đổi kiểu chữ, font chữ ta sử dụng cú pháp sau:

```
Tên_điều_khiển.Font = new Font (font_name, font_style)
```

Trong đó font_name là tên một font hệ thống thuộc System.Drawing.Font; font_style thuộc System.Drawing.Font.Style. Font_Style có thể nhận các giá trị Bold (1), Italic (2), Regular (0), Strikeout (8) và Underline (4).

Ví dụ: muốn đổi kiểu chữ của textbox txtvbmau thành chữ đậm:

```
txtvbmau.Font = New Font(txtvbmau.Font,
FontStyle.Bold)
```

Muốn thêm (hoặc bớt) hiệu ứng font_style ta sử dụng các phép toán bit: OR (XOR).

Ví dụ: Thêm kiểu chữ đậm cho textbox txtvbmau:

```
txtvbmau.Font = New Font(txtvbmau.Font, txtvbmau.Font.Style Or _
FontStyle.Bold)
```

Bỏ hiệu ứng chữ đậm:

```
txtvbmau.Font = New Font(txtvbmau.Font, txtvbmau.Font.Style XOr
_ FontStyle.Bold)
```

Code cho sự kiện checkedchange của chkdam:

```
If chkdam.Checked = True Then
```

```
  txtvbmau.Font = New Font(txtvbmau.Font, txtvbmau.Font.Style Or
_
  FontStyle.Bold)
Else
  txtvbmau.Font= New Font(txtvbmau.Font, txtvbmau.Font.Style Xor
_
  FontStyle.Bold)
End If
```

4.5 Combo box, list box:

4.5.1 Danh sách lựa chọn - list box

Điều khiển này hiển thị một danh sách các đề mục mà ở đó người dùng có thể chọn lựa một hoặc nhiều đề mục. List Box giới thiệu với người dùng một danh sách các lựa chọn. Một cách mặc định các lựa chọn hiển thị theo chiều dọc trên một cột và bạn có thể thiết lập là hiển thị theo nhiều cột. Nếu số lượng các lựa chọn nhiều và không thể hiển thị hết trong danh sách thì một thanh trượt sẽ tự động xuất hiện trên điều khiển.

Thuộc tính:

- `Name:` tên của danh sách lựa chọn
- `Items:` các đề mục được cung cấp
- `SelectionMode`: thuộc tính này cho phép list box có được phép có nhiều lựa chọn khi thực thi hay không? Mặc định là One chỉ cho phép chọn một.
- `Sort:` List Box có sắp xếp hay không?

Phương thức:

- Thêm một phần tử vào List Box. Cú pháp

```
<Name>.Items.Add(Item As String, [Index])
```

Trong đó:

- Name: tên của list box

- Item: Biểu thức (chuỗi) cần thêm vào

- Index: Xác định vị trí đề mục mới được thêm vào. (giá trị 0 xác định cho vị trí đầu tiên). Khi không chỉ định rõ Index thì phần tử thêm vào là mục cuối cùng trong List box mới.

- Thêm các phần tử của một mảng vào List Box. Cú pháp

```
<Name>.Items.AddRange(Tên_mảng)
```

- Xóa một phần tử khỏi List Box.

  - Loại bỏ phần tử có chỉ số i

```
<Name>.Items.RemoveAt (i)
```

- Loại bỏ phần tử đang được chọn

```
<Name>.Items.Remove(<Name>.SelectedItem)
```

- Loại bỏ phần tử khi biết giá trị của nó

```
<Name>.Items.Remove(bt_chuỗi)
```

- Xóa tất cả các phần tử

```
<Name>.Items.Clear
```

◆ Thiết lập hoặc trả về thứ tự của mục chọn hiện thời. (Nếu không có mục nào được chọn trả về -1)

```
<Name>.SelectedINdex()
```

◆ Trả về giá trị của mục chọn thứ i

```
<Name>.Items(i).ToString
```

◆ Kiểm tra đề mục thứ i có được chọn hay không

```
<Name>.GetSelected(i)
```

Biến cố:

◆ `Click`: Được gọi khi người sử dụng nhắp chuột vào điều khiển

◆ `SelectedIndexchanged`: Được gọi khi người sử dụng thay đổi mục chọn trong điều khiển

4.5.2 Điều khiển hộp lựa chọn (Combo box)

Điều khiển Combo box có thể được xem là tích hợp giữa hai điều khiển Textbox và ListBox. Người dùng có thể chọn một đề mục bằng cách đánh chuỗi văn bản vào Combo box hoặc chọn một đề mục trong danh sách.

Điểm khác nhau cơ bản giữa Combo box và List box là điều khiển Combo chỉ gợi ý (hay đề nghị) các lựa chọn trong khi đó điều khiển List thì giới hạn các đề mục nhập vào tức là người dùng chỉ có thể chọn những đề mục có trong danh sách. Điều khiển Combo chứa cả ô nhập liệu nên người dùng có thể đưa vào một đề mục không có sẵn trong danh sách.

Có tất cả 3 dạng của điều khiển Combobox. Các dạng này có thể được chọn tại thời điểm thiết kế.

Kiểu	**Giá trị**	**Hằng**
Single	0	ComboBoxStyle.Single
Drop down	1	ComboBoxStyle.Dropdown
Drop down List	2	ComboBoxStyle.DropdownList

- Single: Người dùng có thể chọn từ danh sách các đè mục đã được hiển thị sẵn ngay bên dưới Combo box. Một thanh trượt sẽ xuất hiện khi còn đề mục chưa được hiển thị hết.

- Drop down: Đây là dạng ngầm định của combo. Người dùng có thể nhập vào trực tiếp hoặc chọn từ danh sách các đề mục.

- Drop down List: Dạng này gần giống List box. Một điểm khác biệt là các đề mục sẽ không hiển thị cho đến khi người dùng nhắp chuột lên mũi tên phía phải của điều khiển. Với dạng này người dùng không thể nhập vào trực tiếp một chuỗi không có trong danh sách.

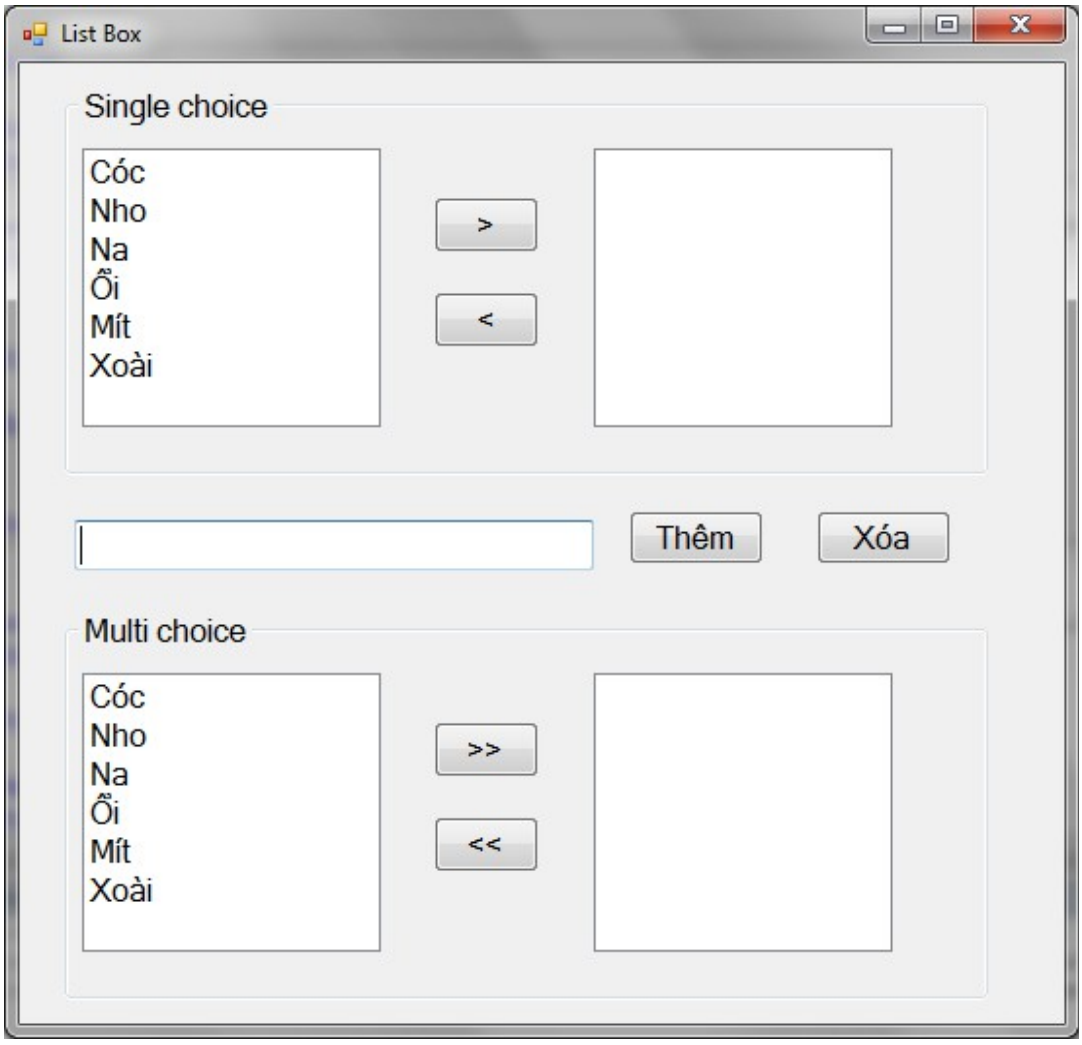
Các thuộc tính và phương thức của combo hoàn toàn giống với list box.

4.5.3 Ví dụ về sử dụng list box

Yêu cầu: Thiết kế form như hình 4.5. Form chứa 2 group box. Groupbox thứ nhất có 2 list box là `lst1` và `lstkq1` có thuộc tính Selection mode là One (chỉ cho phép chọn 1). Groupbox thứ hai có 2 list box là `lst2` và `lstkq2` có thuộc tính Selection mode là Multi Simple (cho phép chọn nhiều đề mục). Listbox `lst1` và `lst2` có chứa các chuỗi Cóc, Nho, Na, Ổi, Mít, Xoài.

Các button (>, >>, <, <<) được sử dụng để chuyển một hoặc nhiều đề mục được chọn từ listbox này sang listbox khác theo chiều tương ứng.

Button Thêm, Xóa dùng để thêm hoặc xóa đề mục trong `lst1` và `lst2`. Nội dung đề mục được thêm hoặc xóa do người sử dụng nhập vào textbox.



Hình 4.5 Form sử dụng điều khiển List box

Đoạn lệnh chuyển một mục chọn từ `lst1` sang `lstkq1`. (Nếu không có mục nào được chọn thì đưa ra thông báo):

```
If lst1.SelectedIndex() <> -1 Then
    lstkq1.Items.Add(lst1.SelectedItem().ToString)
    lst1.Items.Remove(lst1.SelectedItem())
Else
    MsgBox("Bạn phải chọn một mục")
End If
```

Đoạn lệnh bổ sung giá trị hiện có trong textbox 1 vào cả hai listbox `lst1` và `lst2`

```
lst1.Items.Add(TextBox1.Text)
lst2.Items.Add(TextBox1.Text)
```

Đoạn lệnh chuyển nội dung các mục được chọn trong `lst2` sang `lstkq2.`

Do `lst2` đặt thuộc tính Selection Mode là Multi Simple nên giá trị các mục được chọn sẽ là một tập hợp (được trả về bằng phương thức SelectedItems). Ta

dùng vòng lặp For each .. next để duyệt qua lần lượt từng đề mục được chọn trong tập này.

```
Dim chuoi As String

For Each chuoi In lst2.SelectedItems()
     lstkq2.Items.Add(chuoi)
Next

For Each chuoi In lstkq2.Items()
     lst2.Items.Remove(chuoi)
Next
```

4.6 Điều khiển thanh cuộn

Là một điều khiển có thanh trượt cho phép cuộn ngang (HscrollBar) hoặc cuộn dọc (VscrollBar) như một thiết bị nhập hoặc một thiết bị chỉ định cho số lượng hoặc vận tốc.

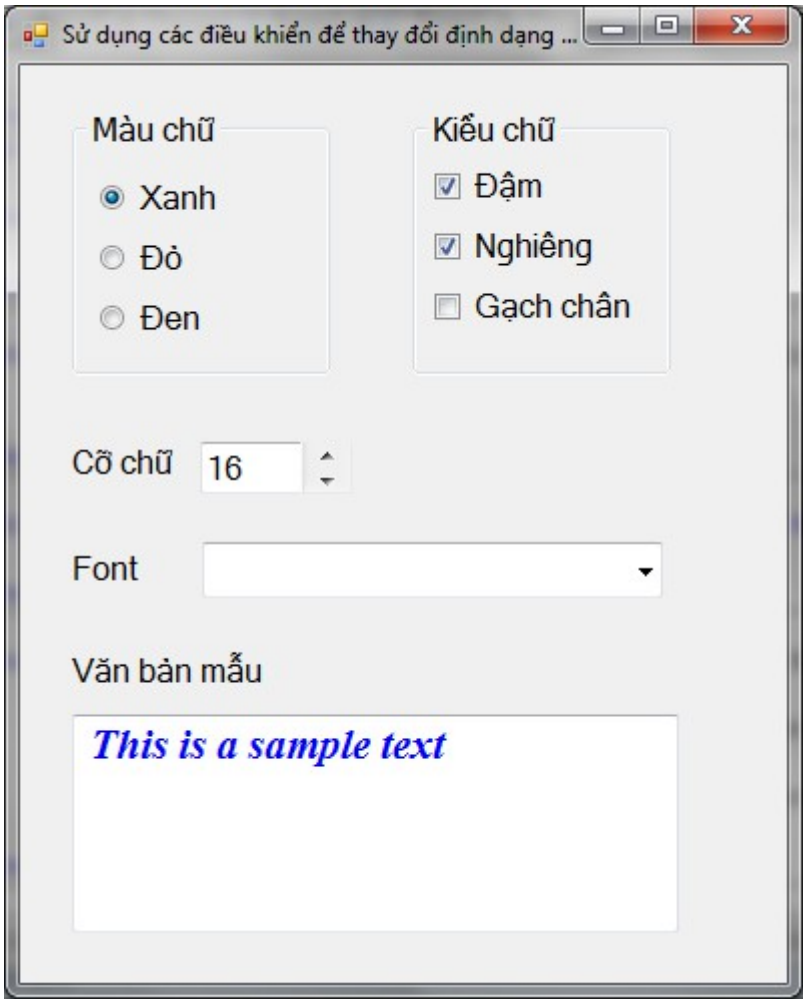
Thuộc tính:

- `Name:` Tên của thanh cuộn
- `Max:` Giá trị lớn nhất của thanh cuộn
- `Min:` Giá trị nhỏ nhất của thanh cuộn
- `Value:` Giá trị tại thời điểm hiện tại của thanh cuộn
- `Largerchange:` Xác định khoảng thay đổi khi người dùng ấn chuột lên mũi tên đầu thanh cuộn.
- `Smallchange`: Xác định khoảng thay đổi khi người dùng ấn chuột lên mũi tên cuối thanh cuộn.

Sự kiện:

- `Scroll`: Xảy ra mỗi khi người sử dụng di chuyển thanh cuộn.

Ví dụ: Trong hình 4-6 dưới đây có sử dụng một textbox và một thanh cuộn dọc để hiển thị giá trị của cỡ chữ. Cỡ chữ sẽ thay đổi khi người sử dụng điều khiển thanh cuộn hoặc khi người sử dụng nhập một giá trị vào textbox và nhấn Enter. Cỡ chữ ban đầu là 16.



Hình 4.6 Form sử dụng kết hợp các điều khiển để thay đổi định dạng văn bản

- Thuộc tính:

Textbox cỡ chữ: + Name: txtsize

Thanh cuộn dọc: + Name: VS1

- Thiết lập giá trị cho textbox và thanh cuộn ở thời điểm chạy chương trình (load form):

```
txtsize.Text = 16
VS1.Value = 16
```

Các giá trị này có thể thiết lập ngay tại thời điểm thiết kế bằng cách thay đổi các thuộc tính thích hợp.

- Đoạn mã cho phép thay đổi giá trị textbox và thay đổi cỡ chữ của hộp văn bản mẫu khi người sử dụng điều khiển thanh cuộn VS1 (sự kiện Scroll của VS1)

```
txtsize.Text = VS1.Value
txtvbmau.Font = New Font(txtvbmau.Font.Name, VS1.Value, _
txtvbmau.Font.Style)
```

- Đoạn mã cho phép thay đổi giá trị thanh cuộn và thay đổi cỡ chữ của hộp văn bản mẫu khi người sử dụng nhập một giá trị mới vào textbox và nhấn Enter (sự kiện KeyPress của txtsize):

```
If e.KeyChar = ChrW(13) Then
   VS1.Value = txtsize.Text
   txtvbmau.Font = New Font(txtvbmau.Font.Name, VS1.Value, _
   txtvbmau.Font.Style)
End If
```

Đoạn chương trình trên sử dụng lệnh khởi tạo font mới theo mẫu số 7 (trong tổng số 13 cú pháp).

```
Tên_điều_khiển.Font = New Font (Font_Name, Font_size,
                      Font_Style)
```

4.7 Menu và Toolbar

4.7.1 Menu

Menu là một loại điều khiển trong đó người sử dụng có thể lựa chọn các mục từ một danh sách cho trước.

Có hai loại menu chính đó là menu thả xuống (Drop Down Menu) và menu popup (Pop-Up Menu). Drop Down menu là menu thường đặt ở đầu biểu mẫu còn popup menu là menu bật ra khi nhấn chuột phải.

Thông thường menu Drop down hay được tạo ở form MDI của ứng dụng.

a. Tạo Drop Down Menu:

- Chọn biểu tượng MenuStrip trên Toolbox và kéo vào form. Một đối tượng Menu mới sẽ xuất hiện dưới cửa sổ thiết kế Form.

- Nhắp chuột vào mục Type Here và nhập tên mục chọn. VB.NET sẽ tự động tạo thêm các mục Type Here bên phải và bên dưới mục vừa được nhập để định nghĩa thêm các lệnh trên menu ngang và menu xổ xuống.

Để thiết lập phím nóng (hot-key) cho mục chọn, nhập thêm dấu & trước ký tự cần định dạng. Mục chọn sẽ được gọi khi người sử dụng nhấn Alt+hotkey.

Để tạo đường phân cách giữa các mục trong menu, nhấn phải chuột vào mục chọn và gọi lệnh Insert/Seperator.

Để tạo hình ảnh cho mục chọn, nhấn phải chuột vào tên mục và chọn lệnh Set image.

b. Tạo Pop-Up menu

- Chọn công cụ ContextMenuStrip và kéo vào form

- Nhập các mục chọn cho pop - up menu

- Muốn pop-up menu xuất hiện khi nhấn phải chuột ở đối tượng nào, ta khai báo thuộc tính ContextMenuStrip của đối tượng đó là tên của pop-up menu.

c. Thuộc tính và biến cố trên menu

Thuộc tính của menu:

- `Text`: Là chuỗi ký tự hiển thị trên menu
- `Name`: tên của menu dùng trong chương trình.
- `Shortcut`: dùng để thiết lập phím tắt gọi menu.
- `CheckedOnClick`: Nếu chọn thuộc tính này thì sẽ có một dấu hiển thị bên cạnh trái. Thuộc tính này không áp dụng cho những menu có chứa menu con.
- `Enabled`: Nếu là False thì mục chọn này sẽ bị xám đi và người sử dụng không thể lựa chọn.
- `Visible`: Nếu là False thì mục chọn không được hiển thị.

Biến cố trên menu:

- `Click`: Xảy ra khi người sử dụng nhắp chuột vào mục chọn.

4.7.2 Toolbar

Toolbar là thanh công cụ chứa các lệnh mà lập trình viên muốn đặt vào biểu mẫu. Để tạo Toolbar ta chọn Toolstrip và kéo vào form sau đó thiết lập các công cụ như đối với menu.

TÓM TẮT NỘI DUNG CỐT LÕI

Trong chương này sinh viên cần chú ý đến các nội dung sau:

- Tạo form MDI và menu, điều khiển các form trong ứng dụng
- Cách sử dụng label, textbox, button
- Khai thác list box và combo box
- Sử dụng thanh cuộn kết hợp với textbox.

Chương 5
WEB FORM VÀ ASP.NET

1 IIS và các khái niệm cơ bản.

1.1 IIS là gì?

IIS là viết tắt của từ (Internet Information Services)

IIS được đính kèm với các phiên bản của Windows.

Microsoft Internet Information Services (các dịch vụ cung cấp thông tin Internet) là các dịch vụ dành cho máy chủ chạy trên nền Hệ điều hành Window nhằm cung cấp và phân tán các thông tin lên mạng, nó bao gồm nhiều dịch vụ khác nhau như Web Server, FTP Server,...

Nó có thể được sử dụng để xuất bản nội dung của các trang Web lên Internet/Intranet bằng việc sử dụng "Phương thức chuyển giao siêu văn bản" - **Hypertext Transport Protocol** (HTTP).

Như vậy, sau khi bạn thiết kế xong các trang Web của mình, nếu bạn muốn đưa chúng lên mạng để mọi người có thể truy cập và xem chúng thì bạn phải nhờ đến một Web Server, ở đây là IIS.
Nếu không thì trang Web của bạn chỉ có thể được xem trên chính máy của bạn hoặc thông qua việc chia sẻ tệp (file sharing) như các tệp bất kỳ trong mạng nội bộ mà thôi.

IIS có thể làm được gì?

Nhiệm vụ của IIS là tiếp nhận yêu cầu của máy trạm và đáp ứng lại yêu cầu đó bằng cách gửi về máy trạm những thông tin mà máy trạm yêu cầu.
Bạn có thể sử dụng IIS để: · Xuất bản một Website của bạn trên Internet · Tạo các giao dịch thương mại điện tử trên Internet (hiện các catalog và nhận được các đơn đặt hàng từ người tiêu dùng) · Chia sẻ file dữ liệu thông qua giao thức FTP. · Cho phép người ở xa có thể truy xuất database của bạn (gọi là Database remote access).Và rất nhiều khả năng khác ...

IIS hoạt động như thế nào?

IIS sử dụng các giao thức mạng phổ biến là **HTTP** (Hyper Text Transfer Protocol) và **FPT** (File Transfer Protocol) và một số giao thức khác như SMTP, POP3,... để tiếp nhận yêu cầu và truyền tải thông tin trên mạng với các định dạng khác nhau.
Một trong những dịch vụ phổ biến nhất của IIS mà chúng ta quan tâm trong giáo trình này là dịch vụ WWW (World Wide Web), nói tắt là dịch vụ Web.
Dịch vụ Web sử dụng giao thức HTTP để tiếp nhận yêu cầu (Requests) của trình duyệt Web (Web browser) dưới dạng một địa chỉ URL (Uniform Resource Locator)

của một trang Web và IIS phản hồi lại các yêu cầu bằng cách gửi về cho Web browser nội dung của trang Web tương ứng.

1.2 Các khái niệm

Các thuật ngữ HTTP, URL, Hyperlink, FTP

HTTP (HyperText Transfer Protocol)

HTTP là viết tắt của HyperText Transfer Protocol, giao thức truyền tệp tin siêu văn bản. Trình duyệt web sử dụng giao thức này để truy xuất và tải về các trang thông tin và các hình ảnh từ máy chủ. Chính vì vậy mà bạn có thể thấy ở ở tiêu đề địa chỉ trang thông tin nào cũng mở đầu bằng http.

Ví dụ, bạn có thể sử dụng trình duyệt web truy xuất vào trang thông tin của Đại học Quốc gia Hà Nội bằng cách gõ vào ô địa chỉ http://www.vnu.edu.vn.

URL (Uniform Resource Locator)

URL (Uniform Resource Locator), bộ định vị tài nguyên thống nhất. Cấu trúc của URL bao gồm: Tên của giao thức (thường là HTTP hoặc FTP), sau đó là địa chỉ của máy tính mà bạn muốn kết nối đến, rồi đến vị trí của tài nguyên cần truy xuất.

Ví dụ về một URL là "fpt://ftp.vnu.edu.vn/ebooks/" sẽ hướng dẫn trình duyệt web của bạn sử dụng giao thức giao thức FTP để kết nối đến máy chủ của Đại học Quốc gia, vào thư mục ebooks (sách điện tử) để có thể tải về các tài liệu.

Siêu liên kết (Hyperlink)

Hyperlink, siêu liên kết, là một phần văn bản (hay hình ảnh) của trang Web, mà khi bạn nhấn chuột vào đó sẽ tự động thực hiện một trong các thao tác sau đây:

Đưa bạn đến phần khác của trang;

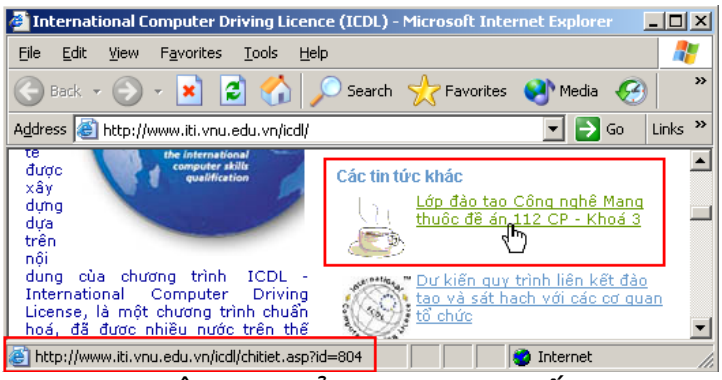
Đưa bạn đến một trang web khác trong cùng một website;

Đưa bạn đến một trang web khác trong website khác;

Cho phép bạn tải về (download) một tệp tin;

Chạy một ứng dụng, trình diễn một đoạn video hoặc âm thanh.

Hình ảnh minh hoạ dưới đây là một phần của trang web của Trung tâm Đào tạo và sát hạch, Viện Công nghệ Thông tin, ĐHQG Hà Nội. Những dòng chữ có gạch dưới chính là các siêu liên kết. Theo mặc định thì các ký tự trong siêu liên kết đó có màu xanh dương.



Hình 2: Chuột đang trỏ vào siêu liên kết, thanh trạng thái hiển thị đích đến

FTP (File Transfer Protocol)

FTP, giao thức truyền tệp tin, là cách thức để truyền dữ liệu từ nơi này đến nơi khác qua mạng Internet. Giao thức này thường được sử dụng để tải về hoặc đưa lên Internet các tệp tin có dung lượng lớn.

Bạn có thể không cần quan tâm cách thực hiện của FTP mà trong hầu hết các trường hợp, chỉ cần nhấn chuột vào một liên kết cho phép tải về tệp tin trong trang web thì trình duyệt sẽ thực hiện các thao tác truyền FTP cho bạn. Điều bạn cần quan tâm là cần phải tải về tệp tin có tên là gì hoặc mình sẽ đặt tên mới cho tệp tin là gì và sẽ lưu trữ nó ở đâu trong máy của mình.

Các chương trình FTP thông dụng giúp bạn có thể nhập vào địa chỉ của máy chủ cần truy xuất tới, tên và mật khẩu đăng nhập (nếu có) và các giao diện để bạn có thể dễ dàng tải về hoặc đưa lên các tệp tin của mình, ví dụ chương trình Total Commander, WSFTP, CuteFTP...

Trình duyệt web (Web browser) là gì?

Trình duyệt web là phần mềm giúp bạn có thể xem được thông tin từ các website trên Internet. Có rất nhiều trình duyệt web khác nhau, ví dụ như trình duyệt web Internet Explorer (IE), Netscape Navigator/Communicator (Netscape), Opera, MyIE2, Mozilla FireBird, Avant... trong đó phổ biến hơn cả là phần mềm trình duyệt IE

Mỗi phần mềm trình duyệt đều có các phiên bản khác nhau, phiên bản mới nhất là phiên bản có nhiều tính năng hơn các phiên bản trước đó. Tuy nhiên, các chức năng sử dụng cơ bản của trình duyệt như lùi (back), tiến (forward), làm tươi (refresh)... đều giống nhau và người dùng chỉ cần biết sử dụng một loại trình duyệt là có thể rất dễ dàng học cách sử dụng các trình duyệt khác để có thể truy xuất và xem các thông tin trên Internet.

2 Cấu trúc của một ứng dụng Web trong framework

Dù có nhiều biến thể, một ứng dụng Web thông thường được cấu trúc như một ứng dụng ba lớp. Ở dạng phổ biến nhất, một trình duyệt Web là lớp thứ nhất, một bộ máy sử dụng một vài công nghệ nội dung Web động (ASP.NET) là lớp giữa, và một cơ sở dữ liệu là lớp thứ ba. Trình duyệt sẽ gửi yêu cầu đến lớp giữa,

lớp giữa sẽ phục vụ bằng cách tạo ra truy vấn và cập nhật cơ sở dữ liệu và tạo ra giao diện người dùng.

ASP.NET

ASP.NET là một tập các lớp nằm trong thư viện lớp cơ sở. ASP.NET cung cấp một mô hình ứng dụng Web dưới dạng một tập các control (đối tượng điều khiển) và cơ sở hạ tầng giúp bạn tạo ra các ứng dụng Web một cách dễ dàng. Các control này được xây dựng cho các ứng dụng trên máy phục vụ (hay còn gọi là Web Forms) phản ánh những control giao diện người dùng HTML đặc thù như Listbox, Textbox và Button và một tập bổ sung các control Web phức tạp hơn như calendars chẳng hạn. Một đặc tính quan trọng của các control trên là chúng được viết để thích nghi với những khả năng của các ứng dụng máy khách. Nói cách khác các đối tượng điều khiển Web forms có thể "đánh hơi" thấy máy khách đang yêu cầu một trang (page) và trả lại người dùng một cách thích hợp.

Ngoài ra ASP.NET còn có khả năng như cập nhật ứng dụng, có thể mở rộng, quản lý và cất giữ trạng thái và nhiều tính năng cao cấp khác.

3 Biến toàn cục – Session và Application

Application và Session là 2 đối tượng khá quan trọng trong ứng dụng web, giúp các trang aspx có thể liên kết và trao đổi dữ liệu cho nhau. Trong phần này, chúng ta sẽ tìm hiểu và sử dụng 2 đối tượng này trong ứng dụng.

3.1 Đối tượng Application

Đối tượng Application được sử dụng để quản lý tất cả các thông tin của một ứng dụng web. Thông tin được lưu trữ trong đối tượng Application có thể được xử lý trong bất kỳ trang aspx nào trong suốt chu kỳ sống của ứng dụng.

Sử dụng biến Application

Tạo biến Application

Application("Tên biến") = <giá trị>

Lấy giá trị từ biến Application

<biến> = Application("Tên biến")

Ví dụ:

```
Application.Lock()
Application("So_lan_truy_cap") = 0
Application("So_nguoi_online") = 0
Application.UnLock()
```

Chú ý:

Do tại một thời điểm có thể có nhiều người cùng lúc truy cập và thay đổi giá trị của các thông tin được lưu trong đối tượng Application, chúng ta nên sử dụng bộ lệnh Lock và UnLock ngay trước và sau khi cập nhật giá trị của biến Application.

Biến Application có thể được sử dụng ở bất kỳ trang nào và được duy trì trong suốt chu kỳ sống của ứng dụng.
Duyệt qua tập hợp biến chứa trong Application

```
Dim i As Integer
Response.Write("<b><u>Danh sách các biến trong đối _
        tượng Application</u></b><br>")
For i = 0 To Application.Count() - 1
 Response.Write(Application.Keys(i) & " : ")
 Response.Write(Application(i) & "<br />")
Next i
```

3.2 Đối tượng Session

Đối tượng Session được dùng để lưu trữ thông tin của người dùng trong ứng dụng. Thông tin được lưu trữ trong Session là của một người dùng trong một phiên làm việc

Chương 8. Hệ thống file và thư mục

1. Xử lý thư mục

1.1. Sử dụng thư viện xuất nhập System.IO

Muốn sử dụng file và thư mục thì trước tiên bạn phải khái báo lệnh sử dụng thư viện IO của .NET:

```
Imports System.IO
```

Thư viện này chứa hầu hết các lớp, phương thức xử lý xuất nhập, đọc/ghi trên hệ thống file. Sau khi khai báo, bạn có thể sử dụng tên các lớp mà không cần sử dụng cách viết khai báo đầy đủ. Ví dụ:

```
'Khai báo xuất nhập
Imports System.IO
Module MakeDir
Sub main()
        'Tạo thư mục
        Directory. CreateDirectory("C:\A")
        ….
```

Một cách khác, bạn có thể không cần khai báo thư viện System.IO nhưng khi muốn sử dụng bất cứ lớp nào trong đó thì phải sử dụng tên đầy đủ như sau:

```
Module MakeDir
Sub main()
        'Tạo thư mục
        System.IO.Directory. CreateDirectory("C:\A")
        ….
```

1.2. Sử dụng lớp Directory

Directory là một lớp chia sẻ (shared) chứa các thư viện sử lý thư mục, có nghĩa rằng bạn không phải tạo một thể hiện của đối tượng lớp Directory khi sử dụng những phương thức lớp. Directory là lớp con nằm trong thư viện System.IO.

Thường, chương trình của bạn tạo ra một thư mục để cất giữ file hoặc thư mục con (sub directory). Để tạo ra một thư mục, bạn có thể sử dụng phương thúc CreateDirectory của lớp Directory. Chương trình MakeDir.vb bên dưới sử dụng phương thức CreateDirectory để tạo ra các thư mục khác nhau. Bằng cách thay đổi tên đường dẫn chuyển cho phương thức CreateDirectory, bạn có thể tạo ra thư mục mới trong thư mục gốc hoặc thư mục chỉ định.

1.3. Kiểm tra lỗi khi xử lý thư mục và file

Bạn có thể dùng phát biểu try … catch để kiểm tra quá trình tạo thư mục không thành công như sau:

```
Try
        Directory.CreateDirectory("Folder3")
        Console.WriteLine("Thu muc tao xong roi !")
Catch E As Exception
        'Đón bắt lỗi
        Console.Writeline("Loi tao thu muc")
        Console.Writeline("E.Massage")
End try
```

1.4. Kiểm tra thư mục tồn tại hay không

Để xác định thư mục có tồn tại hay không, bạn gọi phương thức Exists của lớp Directory. Nếu thư mục chưa tồn tại, chương trình sẽ tạo ra thư mục mới. Nếu có, chương trình sẽ hiển thị thông báo cho biết thư mục đã tồn tại. Chương trình DirectoryExists.vb sẽ minh họa cách xử lý này.

1.5. Xóa thư mục

Cũng như nhu cầu phải tạo ra một thư mục, sẽ có lúc chương trình của bạn phải xóa một thư mục hoặc file mà thư mục chứa đựng. Trong trường hợp như vậy, bạn có thể gọi phương thức Delete của lớp Directory để xóa một thư mục hoặc tất cả các file trong thư mục con (tham số thứ 2 là true)

Directory.Delete("Đường dẫn thư mục", True)

Ví dụ:

Directory.Delete("C:\Myfolder")

Nếu bạn muốn xóa thư mục nhưng nếu có các file trong thư mục thì thao tác xóa không thực hiện được, bạn có thể truyền tham số thứ 2 là false cho phương thức Delete, hoặc không cần truyền đối số thứ hai.

Directory.Delete("Đường dẫn thư mục", false)

Directory.Delete("Đường dẫn thư mục")

Chương trình DeleteDirectory.vb sau sẽ sử dụng phương thức Delete của lớp Directory để xóa thư mục mà bạn tạo ra trước đây bằng chương trình MakeDir.vb.

```
Imports System.IO
Module DeleteDirectory
  Sub Main()
    Console.WriteLine("Xoa thu muc...")
    Try
      Directory.Delete("C:\Folder1", True)
      Console.WriteLine("Da xoa thu muc")
    Catch E As Exception
       Console.WriteLine("Loi xoa thu muc")
       Console.WriteLine(E.Message)
    End Try
  End Sub
End Module
```

1.6. Di chuyển file và thư mục

Ngoài việc cho phép chương trình của bạn xóa một thư mục, lớp Directory cũng cho phép bạn di chuyển thư mục và những file mà thư mục chứa đựng bằng phương thức Move. Phát biểu sau cho phép di chuyển một thư mục có tên C:\VB2005.2\Folder2 sang thư mục gốc với tên Folder 2:

```
Directory.Move("C:\VB2005.2\Folder2", "C:\Folder2")
```

1.7. Xác định và thay đổi thư mục hiện hành

Bạn có thể sử dụng phương thức SetCurrentDirectory và GetCurrentDirectory để thực hiện điều này. Chương trình DirectoryInfo.vb sẽ sử dụng phương thức GetCurrentDirectory để xác định thư mục hiện hành, sau đó sử dụng phương thức SetCurrentDirectory để chỉ định thư mục gốc làm thư mục hiện hành.

```
Imports System.IO
Module DirectoryInfo
  Sub Main()
      'In thông tin về thư mục hiện hành
    Console.WriteLine("Thu muc hien tai la {0}",
Directory.GetCurrentDirectory())
      'Đặt thư mục hiện hành là đường dẫn khác
    Directory.SetCurrentDirectory("C:\")
      'In thông tin về thư mục hiện hành đã thay đổi
    Console.WriteLine("Thu muc hien hanh moi {0}",
Directory.GetCurrentDirectory())
    Console.ReadLine()
  End Sub
End Module
```

1.8. Lấy và xử lý thuộc tính thư mục

Trong quá trình chương trình của chúng ta thao tác xử lý thư mục, sẽ có lúc bạn cần biết thông tin về thư mục (Như ngày tháng và thời gian cập nhật sau

chót…). Để giúp chương trình của bạn xác định những thuộc tính thư mục, lớp DirectoryInfo cung cấp các phương thức mà bạn có thể sử dụng để lấy hoặc đặt thuộc tính cho thư mục.

Chương trình ShowFolderAttribute.vb sau đây sẽ sử dụng phương thức của lớp DirectoryInfo, cho biết thông tin thư mục C:\WINDOWS.

```
Imports System.IO
Module ShowFolderAttribute
   Sub Main()
      If (Directory.Exists("C:\WINDOWS")) Then
         Dim Dir As New DirectoryInfo("C:\Windows")
         Console.WriteLine("Ten day du: {0}", Dir.FullName)
         Console.WriteLine("Ngay tao: {0}", Dir.CreationTime)
         Console.WriteLine("Lan truy cap cuoi: {0}", Dir.LastAccessTime)
         Console.WriteLine("Lan ghi cuoi cung: {0}", Dir.LastWriteTime)
      Else
         Console.WriteLine("C:\WINDOWS khong ton tai")
      End If
   End Sub
End Module
```

Cũng như lớp Directory cung cấp các phương thức sử dụng lấy thông tin về thư mục, lớp DirectoryInfo này cung cấp những phương thức (mà chương trình của bạn có thể sử dụng để đặt thuộc tính cho thư mục). Chương trình SetDirectoryAttributes.vb sau sẽ đặt lại thuộc tính thời gian cho thư mục C:\WINDOWS.

```
Imports System.IO
Module SetDirectoryAttributes

  Sub Main()
    If (Directory.Exists("C:\WINDOWS")) Then
      Dim Dir As New DirectoryInfo("C:\WINDOWS")
      Dim DateTimeNow As DateTime = DateTime.Now()
      Try
        Console.WriteLine("Cap nhat thuoc tinh thu muc")
        ' thay doi thuoc tinh thu muc
        Dir.CreationTime = DateTimeNow
        Dir.LastAccessTime = DateTimeNow
        Dir.LastWriteTime = DateTimeNow
      Catch ex As Exception
        ' Xu ly loi
        Console.WriteLine("Loi khi cap nhat")
        Console.WriteLine(ex.Message)
      End Try
      Console.WriteLine("Thu muc khong ton tai")
    End If
  End Sub
```

End Module

Biên dịch xong thử chạy lại ShowFolderAttribute để nhìn thấy sự thay đổi

2. Xử lý file cùng thư mục

2.1. Lấy danh sách file và thư mục con

Chương trình của bạn có thể lấy về danh sách các file hoặc thư mục con (subdirectory) mà một thư mục chứa đựng. Đây là thao tác thường sử dựng nhất trong quá trình sử lý thư mục. Với những trường hợp như thế, chương trình của bạn có thể sử dụng lớp Directory với phương thức GetFiles, GetDirectory. Sau khi chương trình có được danh sách file hoặc thư mục con, mã có thể sử dụng vòng lặp For Each để duyệt danh sách từng đối tượng file và thư mục như sau:

```
'Lấy về mảng danh sách các tên file trong thư mục hiện hành
Dim files As String() = Directory.GetFiles("*.*")
Dim filename As String
'Duyệt danh sách file đọc được
For Each Filename In Files
        Console.WriteLine(Filename)
Next
```

Chương trình showRootFolder.vb sau sử dụng phương thức GetFiles và GetDirectory để hiển thị danh sách file và thư mục con trong thư mục gốc C:

```
Imports System.IO
```

2.2. Xây dựng lệnh Dir của MS-DOS

Chương trình ShowRootFolder đơn giản vừa hoàn tất trên đây cũng là nội dung lệnh Dir truyền thống của hệ điều hành MS-DOS. Bằng một chút biến tấu linh động hơn, chúng ta có thể xây dựng lại hình ảnh của lệnh Dir mà BillGates cùng Microsoft đã tạo ra hơn 20 năm về trước (Lên nhớ trước đây lệnh Dir đã được viết rất công phu và tốn cả ngàn dòng mã lệnh ngữ assembler), với .NET mọi thứ trở lên cực kỳ đơn giản.

2.3. Duyệt đệ quy thư mục

Để đọc hết nội dung của toàn bộ cấu trúc cây thư mục (gồm tất cả các file và mọi cấp thư mục con), bạn sẽ phải dùng đến kỹ thuật đệ quy. Chương trình ShowFolders.vb sau sẽ hiển thị tất cả các file trong thư mục con do bạn chỉ định bằng phương pháp đệ quy. Đây cũng là bộ khung để bạn có thể mở rộng cho các chức năng duyệt file và sử lý nội dung trên file khác, chẳng hạn như tìm kiếm, đổi tên file hàng loạt, cập nhật, cấp quyền cho file và thư mục…

```
Imports System.IO
```

2.4. Lấp danh sách file và thư mục con bằng lớp DirectoryInfo

Ở ví dụ trên chúng ta thường sử dụng lớp Directory để thao tác xử lý thư mục. Do lớp Directory là một lớp tĩnh, chương trình không cần tạo ra một thể hiện của lớp khi triệu gọi phương thức lớp. tuy nhiên, Visual basic.net cung cấp lớp DirectoryInfo lớp có thể thực hiện những thao tác sử lý trên một thể hiện của đối tượng thư mục cụ thể. Các phương thức mà DirectoryInfo cung cấp hoàn toàn tương tự như những phương thức Directory trong lớp Directory. Ví dụ chương trình sau đây sẽ minh họa cách sử dụng lớp DirectoryInfo liệt kê danh sách thư mục con.

```
Imports System.IO
```

2.5. Sử lý đường dẫn file và thư mục

Khi chúng ta làm việc với file và thư mục, sẽ có lúc bạn cần phân tích thông tin đường dẫn. Ví dụ, nếu chương trình của bạn yêu cầu người dùng nhập vào tên file, bạn có thể trước hết phân tích dữ liệu người dùng vào để xác định xem người dùng để xác định xem người dùng chỉ rõ một tên đường dẫn hay chưa, kiểm tra tên mở rộng, tên file hợp lện hay không.

Để thao tác xử lý thông tin đường dẫn. Chương trình PathInfo.vb, sử dụng lớp Path để hiển thị thông tin của các đường dẫn khác nhau:

```
Imports System.IO
```

Chương 9
Hộp thoại Common Dialog

1. Các hộp thoại chuẩn

1.1. Cho phép người sử dụng chọn mở file

Trong môi trường Windows, hộp thoại chọn mở file được sử dụng rất rộng rãi. Trong hộp thoại này người dùng có thể duyệt danh sách các file qua các ổ đĩa cục bộ hoặc mạng để định vị file mà chương trình cho phép mở. Lợi thế của việc sử dụng hộp thoại này là người sử dụng không cần nhớ tên file hoặc thư mục lưu giữ file.

Chúng ta sử dụng lớp đối tượng .NET trong thư viện System.IO mang tên OpenFileDialog(). Ví dụ:

```
Public Class OpenFileForm
   Private Sub Btn_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Btn.Click
      Dim FileDB As New OpenFileDialog()
      If (FileDB.ShowDialog() = Windows.Forms.DialogResult.OK) Then
         MsgBox("File chọn: " & FileDB.FileName)
      Else
         MsgBox("Người dùng chọn Cancel")
      End If
   End Sub
End Class
```

1.2. Các thao tác sử dụng OpenFileDialog

Khi bạn sử dụng đối OpenFileDialog, bạn có thể xác định bộ lọc cho file.

FileDb.Filter = "All Files | .* | Word files | *.doc | text files | *.txt

Bạn phân tách các bộ lọc bằng ký hiệu |. Để chỉ định bộ lọc mặc định, bạn có thể sử dụng thuộc tính FilterIndex. Ví dụ với bộ lọc trên bạn có thể chỉ định Text File là bộ lọc mặc định như sau:

FileDb.FilterIndex = 3

Thường chương trình của bạn sẽ muốn chỉ rõ những folder khởi đầu xuất hiện khi chương trình hiển thị hộp thoại Open. Để chỉ định hộp thoại ban đầu, chương trình có thể gán thuộc tính thư mục cho thuộc tính InitialDirectory như phát biểu sau:

FileDb. InitialDirectory = "C:\temp"

1.3. Lưu thông tin vào file do người dùng chỉ định

Sau khi mở file sử dụng, chương trình có thể lưu lại file. Để lưu file chúng ta sử dụng hộp thoại SaveFileDialog thay cho OpenFileDialog.

2. Hộp thoại phục vụ in ấn

2.1. Sử dụng hộp thoại PrintDialog để in ấn

Trong Chương trình Visual basic 2005 bạn có thể hiển thị hộp thoại in ấn thông qua lớp PrintDialog. Lớp PrintDialog thừa kế những phương thức từ CommonDialog

2.2. Xác định những máy in có sẵn

Khi chương trình cung cấp khả năng in. Chương trình cần phải biết thông tin về trạng thái máy in sẵn sàng trong hệ thống. Để có một danh sách các máy in trong hệ thống, chương trình của bạn có thể đọc nội dung thuộc tính tập hợp InstalledPrinters của lớp PrinterSettings.

2.3. Thiết lập trang in với hộp thoại PageSetupDialog

Khi người dùng chọn pagesetup từ menu file của các ứng dụng Windows, hệ thống thường hiển thị hộp thoại cho PageSetup cho phép xác định thông số trang in như kích thước, hướng trang nguồn in.
Trong Visual basic 2005, bạn có thể hiển thị hộp thoại PageSetup xử lý xác định thông số thông qua lớp PageSetupDialog. Lớp PageSetupDialog kế thừa những phương thức từ lớp CommonDialog.
2.4 Các thao tác xử lý in ấn và Print Preview
Chúng ta đã có thể chọn máy in, đặt thống số in, trang in, phần còn lại là những thao tác để in dữ liệu dựa trên những thông số được chọn. Để làm được điều này chúng ta sử dụng phương thức PrintPreviewDialog.

CHƯƠNG 10
TUYẾN (THREAD) VÀ TIẾN TRÌNH (PROCESS)

1. Xử lý đa tuyến – Khái niệm

Những hệ điều hành Linux và Windows là những hệ thống đa nhiệm cho phép bạn chạy nhiều hệ điều hành cùng lúc. Như bạn biết, một chương trình đơn giản là một danh sách các chỉ lệnh CPU để thực thi một nhiệm vụ đặc biệt. Trước khi thực hiện một chương trình, chỉ thị lệnh phải cư trú bên trong bộ nhớ truy cập ngẫu nhiên của máy tính (ram).

Nói chung, bạn có thể xem chương trình như một danh sách các chỉ thị lệnh còn tiến trình như một đối tượng chứa các thông tin về chương trình đang chạy. Mỗi khi chạy một chương trình Windows tạo ra một tiến trình mới để giữ những thông tin tài nguyên và trạng thái của chương trình.

Ví dụ, bạn có ba chương trình chạy (tiến trình 1, 2 và 3), Windows sẽ thực thi một số chỉ lệnh của tiến trình 1 trong vòng một khoảng mini-giây. Tiếp đến chuyển đổi đẻ xử lý tiến trình 2 một ít mini-giây nữa và sau cùng là sử lý tiến trình 3. Do hệ điều hành chuyển đổi chỉ thị CPU rất nhanh nên bạn sẽ có cảm giác là các chương trình chạy cùng lúc.

2. Xử lý Thread

2.1. Tạo và chạy nhiều Thread

Thread tương ứng với một chương trình thực thi, để sử dụng thread trong một chương trình Visual basic 2005, bạn trước hết phải tạo ra một đối tượng thread cho mỗi thread mà bạn lập kế hoạch sử dụng. Ví dụ, đoạn mã sau tạo ra 3 đối tượng thread X, Y và Z:

```
Dim X As System.Threading.Thread
Dim Y As System.Threading.Thread
Dim Z As System.Threading.Thread
```

Một thread thực thi một tập các thị lệnh trong chương trình của bạn. Khi bạn tạo ra một thể hiện của thread bạn phải bạn phải chỉ rõ bạn phải chỉ rõ địa chỉ những phát biểu lệnh đầu tiên mà thread sẽ thực thi à thường là địa chỉ của một chương trình con và thủ tục.

Các phát biểu sau sẽ gán các địa chỉ lệnh thực thi khác nhau cho đối tượng thread. Thread A sẽ bắt đầu thực thi thủ tục DisplayA tương tự B là DisplayB và C là DisplayC:

```
A = New Threading.Thread(AddressOff DisplayA)
B = New Threading.Thread(AddressOff DisplayB)
C = New Threading.Thread(AddressOff DisplayC)
```

AddressOff là một chỉ lệnh đặc biệt của ngôn ngữ Visual Basic. Nó cho phép xâm nhập vào vùng địa chỉ nơi hàm địa chỉ nơi hàm tồn tại trong bộ nhớ. Với hàm AddressOff bạn có thể thực hiện những tác vụ cực mạnh tương đương với khái niệm con trỏ trong ngôn ngữ C truyền thống.

Mỗi thủ tục trong trường hợp này, khá đơn giản, thực thi vòng lặp in ra tên thread (như "A", "B" hoặc "C") 100 lần, sau đó kết thúc chương trình con và hủy bỏ thread.

```
Sub DisplayA()
    For I As Integer = 0 To 100
        Console.Write("A")
    Next
End Sub
```

Để khởi động thực thi thread, mã chương trình phải gọi phương thức Start của lớp thread như sau:

```
A.Start()
B.Start()
C.Start()
```

2.2. Đặt thread vào trạng thái chờ ngủ (sleep)

Tùy thuộc vào xử lý của thread trong chương trình mà sẽ có lúc bạn cần đình chỉ sự thực thi của thread trong một khoảng thời gian nào đó. Ví dụ, bạn tạo thread để theo dõi tài nguyên sử dụng của server. Bên trong chương trình bạn có thể sử dụng thời gian định kỳ để kiểm tra ký ức sẵn có, một giây để khảo sát không gian đĩa còn trống và 3 giây kiểm tra kết nối của người dùng từ xa tới server. Tùy vào nhu cầu bạn có thể muốn đánh thức thread và thực thi xử lý mỗi phút hoặc có thể là 10 phút một lần. Để dừng thread trong một khoảng thời gian, bạn có thể sử dụng phương thức Sleep do lớp thread cung cấp. Chẳng hạn, phát biểu sau sẽ đặt dừng (hay ngủ) khoảng 7000 mili-giây (1 giây = 1000 mini-giây).

```
Thread.Sleep(7000)
```

Khi bạn đặt thread vào trạng thái ngủ bằng phương thức Sleep, thread sẽ không thực thi và không tiêu thụ tài nguyên CPU.

Chương trình ThreadSleep.vb sau khởi động 3 thread. Sau đó sử dụng phương thức Sleep để đình chỉ một thread trong vòng 0.3 giây, 0.1 giây và 0.8 giây.

2.3. Dừng, Khởi động lại, hủy bỏ thread

Trong chương trình của bạn sử dugj thread thực thi xử lý những tác vụ đặc biệt. Bạn sử dụng Sleep để dừng thread trong một thời gian biết trước. tuy nhiên, nếu không biết được thời gian dừng là bao nhiêu lâu bạn có thể gọi phương thức Suspend để dừng vô thời hạn một thread.

```
Mythread.Suspend()
```

Tương tự Sleep, Suspend đưa thread vào trạng thái ngủ nhưng không có thời gian tự động thức dậy. để đánh thức thread dậy hoạt động trở lại bạn gọi phương thức Resume như sau:

```
Mythread.Resume()
```

Nếu không muốn sử dụng thread nữa bạn có thể hủy thread để hệ thống giải phóng tài nguyên bằng cách gọi Abort() như sau:

```
Mythread.Abort()
```

2.4. Lấy thông tin về thread

Muốn biết thông tin thread đang hoạt động, lớp thread cung cấp cho bạn rất nhiều phương thức như IsAlive xác định thread còn hoạt động hay không, Priority xác định độ ưu tiên, threadState xác định trạng thái của thread.

2.5. Chờ thread khác hoàn thành các tác vụ sử lý

Khi chương trình của bạn sử dụng thread để xử lý, sẽ có lúc bạn cần chờ xử lý, sẽ có lúc bạn cần chờ thread khác hoạt đông xong thì thread nào đó mới tiếp tục xử lý. Lớp thread cung cấp cho bạn phương thức Join.

Downloading.Join()

2.6. Điều khiển quyền ưu tiên thread

Khi ứng dụng Visual basic 2005 sử dụng nhiều thread thực thi, hệ thống sẽ gán thời gian chiếm giữ CPU cho mỗi thread. Khi thread đang chạy, thường thì hệ thống sẽ dành cho thread 33% thời gian CPU. Nên nhớ rằng, trong một hệ điều hành đa nhiệm như Windows, hệ điều hành chia sẻ thời gian sử dụng CPU cho từng ứng dụng lần lượt sử dụng.

Tùy thuộc vào xứ lý mà thread thực thi trong ứng dụng của bạn, sẽ có lúc thread thực thi này quan trọng hơn những thread khác. Trong những trường hợp như vậy, chương trình của bạn có thể đặt cho thread một quyền ưu tiên cao hơn, khiến cho thread có nhiều thời gian chiếm lĩnh CPU hơn. Để tăng hoặc giảm quyền ưu tiên của thread, bạn gán giá trị ưu tiên cho thuộc tính Priority của lớp thread. Namespace System.Threading.ThreadPriority cung cấp các hằng số giá trị ưu tiên bao gồm: Lowest (thấp nhất), BelowNormal (dưới mức bình thường), Normal (bình thương), AboveNormal (tren mức bình thường), Highest (cao nhất). Ví dụ sau gán quyền ưu tiên cao nhất cho thread Downloader:

Downloader.Priority = ThreadPriority.highest

Chương 11

Lập trình Windows Service

1. Giới thiệu về Windows Services

Trong môi trường Windows, một Service (dịch vụ) là một chương trình đặc biệt chạy phía hậu cảnh để thực hiện một nhiệm vụ nào đó. Ví dụ như hàng đợi máy in cần một dịch vụ (Service) giám sát quá trình xử lý các tác vụ in tài liệu do chương trình ứng dụng gửi đến. Tương tự, dịch vụ Internet Information Services (IIS) chạy phía hậu cảnh để gửi những trang Web và file về trình duyệt (browser) của một máy khách truy xuất Internet. Visual Studio cho phép người lập trình dễ dàng tạo ra những dịch vụ Windows Services. Trong chương này, bạn sẽ học cách tạo những chương trình con mà mỗi Service phải cung cấp để giao tiếp với Windows. Bạn sẽ từng bước thực hiện các thiết lập và chạy Service mỗi khi Windows khởi động. Chúng ta sẽ nghiên cứu về cách xây dựng một Service đơn giản trong Windows, cài đặt và loại bỏ Service. Bạn sẽ học sử dụng lớp ServiceBase cũng như ghi sự kiện của Service ra Event Log, yêu cầu Service thực hiện tác vụ theo định kỳ thời gian. Chúng ta cũng sẽ học sử dụng Thread trong Service, cách thông báo cho người quản trị khi có vấn đề xảy ra với hệ thống và cuối cùng chúng ta sẽ tích hợp chương trình FileSystemWatcher trong chương 2 thành một dịch vụ Windows Service.

Để xem danh sách Windows service bạn thực hiện các bước sau:

1. Start / Settings / Control Panel.

2. Trong cửa sổ Control Panel chọn Administrative Tools.

3. Trong cửa sổ Administrative Tools chọn biểu tượng Services.

Visual Studio cho phép bạn dễ dàng tạo ra một dịch vụ Windows Service. Tạo Service cũng tương tự như tạo một chương trình bình thường. Visual Studio sẽ sinh ra bộ khung để bạn thêm mã vào điều khiển các chức năng của Service.

2. Xây dựng một Windows Service đơn giản.

Trong phần này bạn sẽ tạo ra và sau đó cài đặt dịch vụ Windows Service. Để giúp bạn quen với bước đầu viết Service, Visual Studio cung cấp kiểu dự án Service Windows, bạn chọn kiểu dự án này trong hộp thoại New Project thay cho dự án Windows Application thông thường. Service chúng ta tạo trong chương này sẽ có tên là DemoService.

Dịch vụ Service của chúng ta, trong trường hợp này sẽ không thực hiện bất kỳ xử lý nào. Thay vào đó, dịch vụ sẽ ghi lại một số tác vụ truy xuất trong file sự kiện ở đĩa gốc là file DemoEvents.log. Service sẽ ghi vào file log thông báo thời gian dịch vụ bắt đầu và chấm dứt.

Để tạo ra một dịch vụ Windows sử dụng Visual Studio, bạn thực hiện những bước sau:

1. Trong Visual Studio, chọn File / New Project.

2. Visual Studio sẽ hiển thị hộp thoại New Project.

3. Trong hộp thoại New Project, chọn biểu tượng Windows Services.

Gõ tên cho Service trong ô Name và nhấn OK.

4. Visual Studio sẽ hiển thị một cửa sổ biểu diễn cho Service.

5. Trong màn hình Design View, nhấn phím phải lên mục Project.

6. Visual Studio sẽ hiển thị một menu Popup. Chọn Properties từ menu.

Visual Studio sẽ hiển thị các thuộc tính của dịch vụ.

Trong danh sách thuộc tính liệt kê, gán tên DemoService cho thuộc tính Service (không phải thuộc tính Name). Nhấn vào mục Code View để xem mã cài đặt của DemoService. Như đã nêu, mỗi khi DemoService bắt đầu, dịch vụ sẽ nối vào file log một chuỗi thông báo. Để thực hiện thao tác dẫn nhập file, bạn cần nhập thêm namespace System.IO như sau:

Imports System.IO

Sau đó đặt các phát biểu sau vào phương thức OnStart do Visual Studio sinh mã sẵn.

Như bạn có thể thấy, phát biểu mở file sau đó sử dụng phương thức WriteLine() để ghi dữ liệu. Để file có thể sử dụng lại, chúng ta không đóng file trọng phương thức OnStart(). Trong phương thức OnStop() chúng ta ghi thông báo cho biết Service kết thúc và file mở trước đây trong OnStart() sẽ được đóng lại.

Trước khi bạn có thể cài dịch vụ vào hệ thống Windows, bạn phải thêm mã cài đặt vào dự án DemoService, bạn thực hiện những bước sau:

1. Chọn Design View và nhấn chuột phải trên cửa sổ để hiện thị menu popup.

2. Trong menu Popup bạn chọn Add Installer. Visual Studio sẽ thêm hai thành phần vào dự án Service Process Installer và Service Installer.

Khi Windows cài đặt một dịch vụ, Windows cài đặt theo ngữ cảnh mà đặc quyền của dịch vụ có thể sử dụng. Nếu sử dụng Service Process Installer mà Visual Studio thêm vào dự án của bạn, bạn phải chỉ rõ một tài khoản trên hệ thống (tài khoản đặc biệt bạn tạo ra cho dịch vụ) dùng cho phép truy cập vào ngữ cảnh của dịch vụ. Trước hết, bạn phải chọn kiểu tài khoản như mô tả trong bảng dưới đây:

Kiểu tài khoản	**Mục đích**
Local Service	Định hướng Windows chạy dịch vụ trong ngữ cục bộ có một số đặc quyền ưu tiên.

Network Service	Định hướng Windows chạy dịch vụ trong ngữ cảnh không được ưu tiên sử dụng tài khoản cục bộ.
Local System	Hệ thống cục bộ, tương đương với một tài khoản vô danh đăng nhập sử dụng hệ thống.
User	Định hướng Windows mỗi khi chạy dịch vụ trong ngữ cảnh người dùng đều yêu cầu hỏi nhập trực tiếp username và password.

Tiếp đến là xác định username cho tài khoản. Nếu bạn không chỉ rõ thông tin ngữ cảnh tài khoản, Windows sẽ không thể cài đặt dịch vụ. Để chỉ rõ thông tin cài đặt của dịch vụ, hãy nhấn phím phải lên mục ServiceProcessInstaller trong cửa sổ Properties, bạn chỉ định kiểu tài khoản như hình 9-5 và chọn loại user. Tiếp theo, biên dịch dịch vụ. Ở giai đoạn này, dịch vụ của bạn cần phải chứa các phát biểu xử lý, ở đây đơn giản chúng ta chỉ ghi ra log file trong sự kiện OnStart() và OnStop() của dịch vụ. Chọn Build / Build DemoService từ menu để biên dịch.

3. Cài và tháo bỏ Service trong Windows

Trong thí dụ trên, sau khi biên dịch chúng ta đã được một dịch vụ Windows có tên là DemoService.exe. Bạn phải định hướng về Windows cài đặt dịch vụ (chạy chương trình dịch vụ nơi hậu cảnh) khi hệ thống của bạn khởi động. Một trong những cách dễ nhất để cài đặt dịch vụ là sử dụng chương trình InstallUtil (được cung cấp bởi Visual Studio) mà bạn có thể chạy từ dòng lệnh. Để cài đặt DemoService.exe, mở cửa sổ Command và đánh lệnh InstallUtil sau:

InstallUtil Path \ DemoService.exe

Trong đó, Path là đường dẫn thư mục của file DemoService.exe. Hãy lưu ý những thông báo kết xuất bởi InstallUtil.exe. Nếu cài đặt service không thành công, bạn sẽ nhận được thông báo nguyên nhân gây lỗi. Sau khi lệnh InstallUtil chấm dứt

thành công, dịch vụ của bạn được cài đặt vào hệ thống Windows, nhưng nó sẽ vẫn chưa chạy (bạn sẽ học cách khiến nó tự khởi động sau). Để khởi động thủ công dịch vụ, bạn thực hiện các bước sau:

1. Chọn Select Start / Settings / Control Panel. Windows sẽ hiển thị cửa sổ Control Panel.

2. Trong Control Panel, hãy nhấn đúp biểu tượng Administrative Tools. Trong cửa sổ Administrative Tools, hãy nhấn đúp biểu tượng Services.

3. Trong cửa sổ Services, hãy nhấn phím phải chuột vào dịch vụ mà bạn mong muốn khởi động và sau đó chọn Start.

Ngoài việc sử dụng trình quản lý Services của Windows trên đây, bạn còn có thể khởi động trực tiếp Service từ Visual Studio sử dụng cửa sổ Server Explore. Bạn thực hiện các bước sau:

1. Trong Visual Studio, chọn View / Server Explore, Visual Studio sẽ mở cửa sổ Server Explore.

2. Trong Server Explore, hãy kích chọn server tương ứng với hệ thống của bạn và sau đó lựa chọn dịch vụ liệt kê ra trong phần Services.

3. Tong danh sách các dịch vụ, hãy nhấn phím phải vào dịch vụ mà bạn muốn khởi động và sau đó chọn Start từ menu thanh công cụ.

Khi bạn không còn yêu cầu sử dụng một dịch vụ nữa, bạn có thể sử dụng lại lệnh InstallUtil để loại bỏ dịch vụ. Trong trường hợp này, chúng ta sử dụng thêm khoá chuyển /U với ý nghĩa sử dụng chức năng loại bỏ (Uninstall) như sau:

InstallUtil /U Path\DemoService.exe

4. Sử dụng lớp ServiceBase

Khi bạn tạo ra một dịch vụ Windows, dịch vụ của bạn sử dụng lớp ServiceBase đặt trong namespace System.ServiceProcess.

Lớp ServiceBase tương ứng với tiến trình mà Windows tạo ra để thực thi các yêu cầu xử lý để gửi đến dịch vụ. Bằng cách viết lại mã lệnh đè chồng lên

phương thức lớp cơ sở, chúng ta định nghĩa lại các thao tác dịch vụ thực hiện khi nó bắt đầu (start), dừng (pause) hoặc chấm dứt (stop) và sau đó tiếp tục các thao tác khác của dịch vụ.

Để kiểm tra sự kiện dịch vụ có thể tạm ngừng, tiếp tục, hay Shutdown. Chúng ta thực hiện theo các bước sau:

- Trong chế độ thiết kết, nhấn phím lên dịch vụ và chọn thuộc tính propeties menu
- Trong danh sách dịch vụ liệt kê ra, chúng ta sử dụng drop-down để cho phép hoặc vô hiệu hóa những sự kiện đối với dịch vụ.
- CanHandlePowerEvent = true
- CanPauseAndContinue = true
- CanShutdown = true

Chương 13

ADO.NET - TRUY CẬP DỮ LIỆU

Nếu bạn từng làm việc hay lập trình với cơ sở dữ liệu, thì khi tiếp xúc với mỗi hệ cơ sở dữ liệu để viết ứng dụng sẽ thấy thật không đơn giản chút nào. Một chương trình viết ra trước tiên sẽ được hỏi: sử dụng cơ sở dữ liệu nào? SQL Server, Oracle hay Access... Có quá nhiều loại cơ sở dữ liệu mà mỗi người lập trình đòi hỏi phải thông thạo khi tiếp xúc với chúng.

FoxPro

Access

Microsoft SQL Server

Oracle

DB2

MySQL

Postgress SQL

InterBase

Pocket Database

XML Database

...

Hầu như mỗi hệ cơ sở dữ liệu đều cung cấp cách lưu trữ và lập trình ngay bên trong hệ thống của chính nó. Như chương 1 bạn đã xem qua, ta có thể viết hẳn một chương trình bên trong FoxPro hay Access.

Tuy nhiên, vấn đề sẽ phát sinh khi bạn dùng một ngôn ngữ lập trình không liên quan gì tới cơ sở dữ liệu để đọc và xử lý dữ liệu từ các hệ cơ sở dữ liệu trên. Không có cách nào mà một ngôn ngữ lập trình như Visual Basic, C++ hay sau này là C#, VB.NET truy cập trực tiếp vào từng hệ cơ sở dữ liệu. Mỗi hệ cơ sở dữ liệu đều có cách tổ chức, sắp xếp và cấu trúc dữ liệu đặc thù. Lập trình viên phải dùng những thư viện mà hệ cơ sở dữ liệu cung cấp để tiếp cận việc truy xuất cơ sở dữ liệu. May mắn thay, vấn đề cũng dần dần được giải quyết khá ổn thoả. Bạn sẽ thấy chặng đường để có được công nghệ truy xuất dữ liệu ADO.NET mà chúng ta sắp học là quãng đương của lich sử với rất nhiều kinh nghiệm mà nhà sản xuất phần mềm Microsoft tích luỹ được.

1. ODBC

Từ phiên bản Windows 3.1, cách đây hơn 20 năm, Microsoft đã nhận thấy sự lộn xộn và không nhất quán trong lập trình truy xuất các hệ cơ sở dữ liệu.

ODBC là một giao diện tương tự khi khái niệm Interface trong .NET. Microsoft chỉ đưa ra những đặc tả chung nhất cho ODBC như:

SQLConnect ( )

SQLColumns ( )

SQLDisconnect ( )

SQLEndTran ( )

SQLExecute ( )

SQLFetch ( )

SQLGetData ( )

SQLRowCount ( )

SQLTablets ( )

Đặc tả ODBC đơn giản chỉ là các tên hàm và không có cài đặt cách xử lý gì cả. Microsoft yêu cầu mỗi nhà cung cấp và nhà sản xuất ra hệ cơ sở dữ liệu phải cài đặt cách ứng xử cho những hàm này theo quy định của ODBC. Như vậy, về tên gọi hàm thì giống nhau nhưng mỗi nhà cung cấp sẽ cài đặt cách thức xử lý hàm của riêng mình. Chẳng hạn, SQL Server sẽ cài đặt hàm SQLConnect() khác với cách cài đặt của Oracle, nhưng kết quả cuối cùng vẫn là chuẩn bị mở tài nguyên kết nối để người dùng truy xuất dữ liệu. Bằng sự thống nhất này, giờ đây lập trình viên được giải thoát khỏi vấn nạn phải học lập trình trên các hàm thư viện nguyên gốc của hệ cơ sở dữ liệu. Bạn hình dung nếu không có ODBC, có thể cùng một thao tác kết nối, bạn phải học và nhớ tên 10 hàm khác nhau tương ứng với 10 loại cơ sở dữ liệu.

Như vậy, tương ứng với mỗi loại cơ sở dữ liệu, bạn thường cần đến một trình điều khiển ODBC tương ứng. Muốn biết ODBC được Windows quản lý ở đâu, bạn chọn mục Start / Program Files / Control Panel / Administrative Tools / ODBC Data Source. Hình 2-3 là danh sách các driver của ODBC.

Trong .NET từ phiên bản 1.1 cho đến 2.0 và 3.0, tất cả các lớp truy xuất dữ liệu liên quan đến ODBC đều được chứa trong namespace System.Data.Odbc và có tiếp đầu ngữ Odbc phía trước để bạn phân biệt, ví dụ:

Imports System. Data. Odbc

Module A

Public Sub Main ( )

```
Dim conn As New OdbcConnection ( )
End Sub
End Module
```

Với ODBC, một ưu điểm nổi bật là ứng dụng không cần quan tâm tới đặc thù của cơ sở dữ liệu, ứng dụng nếu thiết kế tốt có thể dễ dàng chuyển đổi chạy được trên nhiều cơ sở dữ liệu khác nhau có trình ODBC hỗ trợ.

2. OLEDB

Một bất lợi của ODBC là trình điều khiển phải viết bằng các ngôn ngữ cấp thấp như C/C++, vì vậy khó bảo trì cũng như mở rộng cho các trình điều khiển ODBC theo đặc tả ban đầu nên cần phải chuyển đổi ODBC sang mô hình truy xuất cơ sở dữ liệu tổng quát dễ mở rộng hơn.

Cùng với kiến trúc *liên kết và nhúng đối tượng* (Object Linking and Embeding hay OLE) được thiết kế cho nền hệ điều hành với Windows 95 ổn định và vững chắc, Microsoft đã quyết định xây dựng một nền tảng mowid cho truy xuất dữ liệu thay thế cho ODBC và thế là đặc tả OLEDB ra đời. Thay vì phải xây dựng trình điều khiển theo đặc tả ODBC, giờ đây các nhà sản xuất và cung cấp hệ cơ sở dữ liệu sẽ viết theo đặc tả OLEDB.

Vì công nghệ OLE được tích hợp với hệ điều hành và cũng rút kinh nghiệm từ cách xây dựng ODBC, Microsoft đã tối ưu và giúp cho các nhà phát triển trình điều khiển OLEDB đạt được tốc độ truy cập nhanh và hiệu quả hơn. Ngay lập tức, chuẩn OLEDB được chấp nhận và lần lượt các driver truy xuất thay thế cho ODBC ra đời như:

SQL Server OLEDB Driver

Oracle OLEDB Driver

Access OLEDB Driver

...

Với một số nhà cung cấp chưa hoặc không viết kịp trình điều khiển truy xuất dữ liệu theo đặc tả OLEDB, Microsoft cung cấp thêm một cách trung gian để lập trình viên có thể tiếp cận với kỹ thuật OLEDB - đó là trình điều khiển OLEDB dành cho ODBC. Bằng cách này coi như tất cả các ứng dụng đều sử dụng được công nghệ OLEDB mới.

Trong .NET tất cả các trình truy xuất dữ liệu liên quan đến OLEDB đều chứa trong namespace System.Data.oledb. Ví dụ:

```
Imports System. Data. Oledb

Module A

Public Sub Main ( )

Dim Conn As New OledbConnection ( )

End Sub

End module
```

3. ADO

Khi các trình điều khiển OLEDB đã trở nên thông dụng, Microsoft hướng tới lập trình viên nhiều hơn, ADO (**A**ctiveX **D**ata **O**bject) là một cải tiến tiếp theo trong chiến lược xây dựng trình điều khiển truy cập dữ liệu thống nhất (Universal Data Access).

ADO sử dụng công nghệ ActiveX (một phần mở rộng của công nghệ OLE, lúc đó vẫn thường được gọi là giải pháp "bình mới rượu cũ" của Microsoft) cho phép truy xuất dữ liệu dạng đối tượng. ADO cung cấp mô hình mở Provider cho phép các nhà sản xuất cơ sở dữ liệu tự viết thêm phần mở rộng cho ADO như: cập nhật dữ liệu theo lô (Batch Update), trả về cùng lúc nhiều kết quả recordset trong câu truy vấn. ADO đã được sử dụng rộng rãi trong rất nhiều ứng dụng cơ sở dữ liệu và nhất là những ứng dụng viết bằng ngôn ngữ Visual Basic 6 (thế hệ ngôn ngữ lập trình có khả năng sử dụng hiệu quả nhất kiến trúc ADO). Một vấn đề phát sinh của ADO đó là yêu cầu về tài nguyên kết nối, lập trình viên phải chủ động

đóng mở kết nối mỗi khi hoàn tất quá trình truy xuất. Không thích hợp cho mô hình truy xuất của Internet với hàng triệu kết nối có thể truy cập đồng thời cơ sở dữ liệu.

4. ADO.NET

4.1. Kiến trúc ADO.NET.

Trong những ngày đầu khi Internet phát triển, ADO được sử dụng rất nhiều trong các ứng dụng Web viết bằng ASP (Active Server Page). Ngay lập tức Microsoft nhận ra mô hình kết nối ADO không thích hợp cho mô hình ứng dụng Internet và những ứng dụng phân tán trong tương lai. Một cuộc cách mạng lớn đã diễn ra để tạo nên công nghệ truy xuất hoàn thiện ADO.NET. Tuy chỉ gắn thêm 4 chữ .NET vào với ADO nhưng thực chất ADO.NET hoàn toàn khác với ADO. Nếu bạn đã từng lập trình với ADO thì có thể hoàn toàn bất ngờ khi chuyển sang ADO.NET vì gần như là không có gì tương tự cả. ADO.NET hỗ trợ rất mạnh cho các ứng dụng phân tán, ứng dụng Internet, các ứng dụng đòi hỏi tốc độ, khối lượng người truy cập đồng thời lớn.

Trong kiến trúc ADO.NET có hai thành phần chính đó là thành phần truy cập dữ liệu và thành phần lưu trữ xử lý dữ liệu.

Thành phần thứ nhất gọi là .NET Framework Data Providers, được thiết kế để thực hiện các thao tác kết nối, gửi các lệnh xử lý tới cơ sở dữ liệu, thành phần này còn được gọi với một tên khác là lớp kết nối (Connectivity Layer).Trong ADO.NET, có 4 đối tượng chính để xử lý phần kết nối và tương tác với dữ liệu mà bạn sẽ phải thường xuyên làm việc là:

Connection: Đối tượng cho phép bạn kết nối đến các nguồn cơ sở dữ liệu như: SQL Server, Oracle, ODBC và OLEDB.

Command: Đối tượng cho phép bạn truy cập cơ sở dữ liệu và thực thi phát biểu SQL hay thủ tục Store Procedure của cơ sở dữ liệu, truyền tham số và trả về dữ liệu.

DataReader: Bô đọc, dùng để đọc nhanh dữ liệu nguồn theo một chiều.

DataAdapter: Bô điều phối hay cầu nối, dùng để chuyển dữ liệu truy vấn được cho các đối tượng lưu trữ và xử lý như DataSet, DataTable. DataAdapter chủ yếu sẽ thực hiện các thao tác truy vấn (SELECT), thêm mới (INSERT), chỉnh sửa (UPDATE) và xoá (DELETE).

Một lần nữa cũng như ODBC và OLEDB, ADO.NET đưa ra những đặc tả yêu cầu nhà cung cấp hỗ trợ. Hầu hết những nhà cung cấp cơ sở dữ liệu phải thiết kế trình điều khiển cài đặt nội dung (hàm, phương thức, thuộc tính) cho những đối tượng nêu trên theo đặc thù cơ sở dữ liệu của mình, những trình điều khiển này còn được gọi với tên là ADO.NET Provider. Khởi đầu, Microsoft cung cấp và hỗ trợ Provider những hệ cơ sở dữ liệu thông dụng như SQL Server, Oracle.

Thành phần thứ hai, DataSet được xem như Container dùng để lưu trữ đối tượng liên quan đến dữ liệu như DataTable, DataRelation, DataView, đôi khi còn gọi là lớp không kết nối (Disconnect Layer). Bạn có thể hình dung DataSet như một cơ sở dữ liệu hay Database thu nhỏ. Trong DataSet bạn có thể chứa các đối tượng như bảng, View, ràng buộc, quan hệ. Sở dĩ gọi là Disconnect Layer là do toàn bô thông tin dữ liệu sẽ được đọc một lần duy nhất và lưu vào DataSet hay các đối tượng dữ liệu như DataTable và quá trình kết nối chấm dứt. Dữ liệu sau đó được xử lý độc lập phía ứng dụng máy khách (client). Disconnected có thể xem là mô hình bắt buộc của ADO.NET. Bằng cách này, các ứng dụng có thể đảm bảo tài nguyên cho hệ thống và phục vụ được số lượng kết nối lớn.

Như đã nêu, ADO.NET cung cấp một giao diện hay đặc tả cho các đối tượng như Connection, Command, DataReader, DataAdapter. Các nhà cung cấp sẽ đưa ra các trình điều khiển gọi là Provider. Mỗi Provider sẽ đặt tên cho các đối tượng theo cách của mình, thường là thêm tiếp đầu ngữ của Provider vào tên những đối tượng nêu trên. Ví dụ:

Provider SQL Server, namespace System.Data.SqlClient

SqlConnection

SqlCommand

SqlDataReader

SqlDataAdapter

Provider Oracle, namespace System.Data.Oracle

OracleConnection

OracleCommand

OracleDataReader

OracleDataAdapter

Provider để truy xuất các trình điều khiển OLEDB, namespace System.Data.OleDB

OledbConnection

OledbCommand

OledbDataReader

OledbDataAdapter

Provider để truy xuất các trình điều khiển ODBC, namespace System.Data.ODBC.

OdbcConnection

OdbcCommad

OdbcDataReader

OdbcDataAdapter

Trên đây là các Provider chuẩn do Microsoft xây dựng sẵn. Bạn vẫn có thể sử dụng những Provider khác. Chẳng hạn, như để truy xuất cơ sở dữ liệu MySQL bạn có thể download Provider viết cho .NET của cơ sở dữ liệu này và sử dụng chúng trong chương trình bằng lệnh Imports.

Các lập trình viên chuyên về cơ sở dữ liệu ở thế hệ lập trình bằng ADO trên ngôn ngữ Visual Basic 6 trước đây có thể khi lần đầu tiên tiếp cận với ADO.NET

sẽ lúng túng vì ADO.NET có quá nhiều thứ và đối tượng để giải quyết cùng một vấn đề. Trong ADO, đơn giản chỉ dùng đối tượng ADO Connection và Recorderset là đủ.

4.2. Truy xuất dữ liệu với SQL Server Provider

Bạn tạo mới ứng dụng Console mang tên SQLDemo. Tiếp đến mã chương trình sử dụng ADO.NET Provider của SQL Server được viết như sau:

```
'Sử dụng Provider dùng cho CSDL SQL Server
Imports System.Data.SqlClient
Module SQLDemo
   Sub Main()
      'Khai báo đối tượng kết nối Connection
      Dim conn As New SqlConnection()
      'Gán thông tin kết nối
      conn.ConnectionString = "Data Source=(local);Initial
Catalog=NorthWind;user=sa;pwd=123123"
      'Mở kết nối
      conn.Open()
      'Khai báo đối tượng điều phối dữ liệu
      Dim da As New SqlDataAdapter("SELECT * FROM Customers", conn)
      'Định nghĩa đối tượng chứa bảng dữ liệu
      Dim table As New DataTable
      'Đọc dữ liệu từ CSDL và đưa vào table
      da.Fill(table)
      'Duyệt và in ra các dòng dữ liệu đưa vào table
      For i As Integer = 0 To table.Rows.Count - 1
         Console.WriteLine("  {0} | {1}   ", table.Rows(i)("CustomerID"),
table.Rows(i)("CompanyName"))
      Next
      Console.ReadLine()
   End Sub

End Module
```

Trong chương trình trên chúng ta sử dụng hai thành phần Connection và DataAdapter. Vì biết trước sẽ truy cập vào cơ sở dữ liệu SQL Server nên mã chương trình quyết định chọn Provider là *System.Data.SqlClient*. Như bạn thấy, đối tượng *SqlConnection* và *SqlDataAdapter* chính là cài đặt cho Connection và DataAdapter. Đối tượng *SqlDataAdapter* tiếp nhận câu lệnh SQL và mở kết nối

đến nguồn dữ liệu thông qua đối tượng conn. Lệnh *da.Fill (table)* sẽ chuyển dữ liệu này là kết quả của câu truy vấn "*SELECT * FROM Customer*" từ cơ sở dữ liệu MyCompany vào bảng *table*. Vòng cặp for sẽ duyệt và in nội dung các dòng dữ liệu trong *table* ra màn hình.

Nếu muốn hiển thị dữ liệu trên ứng dụng Windows Forms, bạn tạo mới dự án Windows Application, kéo điều khiển DataGridView lên Form và viết mã trong sự kiện Load của Form như sau:

```
'Sử dụng Provider dùng cho CSDL SQL Server
Imports System.Data.SqlClient

Public Class SQLDemo

    Private Sub SQLDemo_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Khai báo đối tượng kết nối Connection
        Dim conn As New SqlConnection()
        'Gán thông tin kết nối
        conn.ConnectionString = "Data Source=(local);Initial
Catalog=NorthWind;user=sa;pwd=123123"
        'Mở kết nối
        conn.Open()
        'Khai báo đối tượng điều phối dữ liệu
        Dim da As New SqlDataAdapter("SELECT * FROM Customers", conn)
        'Định nghĩa đối tượng chứa bảng dữ liệu
        Dim table As New DataTable
        'Đọc dữ liệu từ CSDL đổ vào table
        da.Fill(table)
        'Hiển thị dữ liệu trong khung lưới DataGrid
        Me.DataGridView1.DataSource = table
    End Sub
End Class
```

Tương tự ví dụ Console, nhưng thay vì gọi WriteLine để in ra kết quả ở đây ta gán đối tượng Table cho thuộc tính DataSource của DataGridView.

4.3. Truy xuất dữ liệu với Oracle Provider

Nếu hệ thống xác định nhu cầu là cần truy xuất đến cơ sở dữ liệu Oracle thay vì SQL Server bạn có thể dùng Oracle Provider. Mã chương trình là hoàn toàn

không đổi, bạn chỉ cần thay đổi đối tượng Connection và DataAdapter sử dụng Provider dành cho cơ sở dữ liệu Oracle tương ứng.

4.4. Truy xuất dữ liệu với OLEDB Provider

Theo cách tương tự nếu bạn sử dụng OLEDB Provider, mã chương trình sẽ như sau:

```
Imports System.Data.OleDb
Module Module1

    Sub Main()
        'Khai báo đối tượng kết nối Connection
        Dim conn As New OleDbConnection()
        'Gán thông tin kết nối
        conn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=yourdatabasename.mdb;"
        'Mở kết nối
        conn.Open()
        'Khai báo đối tượng điều phối dữ liệu
        Dim da As New OleDbDataAdapter("SELECT * FROM Customers", conn)
        'Định nghĩa đối tượng chứa bảng dữ liệu
        Dim table As New DataTable
        'Đọc dữ liệu từ CSDL và đưa vào table
        da.Fill(table)
        'Duyệt và in ra các dòng dữ liệu đưa vào table
        For i As Integer = 0 To table.Rows.Count - 1
            Console.WriteLine("  {0} | {1}    ", table.Rows(i)("CustomerID"),
table.Rows(i)("CompanyName"))
        Next
        Console.ReadLine()
    End Sub

End Module
```

4.5. Truy xuất dữ liệu với ODBC Provider

Với nguồn dữ liệu ODBC, bạn cũng có thể dễ dàng hình dung mã truy xuất của chúng ta sẽ là:

Trong tất cả các truy xuất thông qua Provider trên đây chỉ có phần nội dung kết nối conn.ConnectionString là khác nhau. Chúng ta học về ý nghĩa chuỗi kết nối ConnectionString trong chương sau.

4.6. Tự làm khó mình

ADO.NET tách biệt các trình Provider ra như vậy nhằm giúp việc truy xuất dữ liệu được nhanh hơn, vì mỗi Provider được viết tối ưu hoá theo thư viện gốc Native mà cơ sở dữ liệu cung cấp. Thế nhưng một lần nữa nếu bạn tinh ý sẽ nhận ra có điều gì đó không ổn ở đây. Tuy phương thức, đối tượng và thuộc tính của một lớp đối tượng như SqlConnection, OracleConnection, OleDbConnection, OdbcConnection đều gọi như nhau nhưng tên lớp lại khác nhau. Điều này sẽ khiến bạn khó có thể viết một ứng dụng mà mã chương trình có thể tuỳ biến sử dụng tất cả các Provider. Một ví dụ dễ hình dung nhất là nếu bạn đã viết được 10.000 dòng mã và tất cả đều sử dụng SqlConnection, SqlDataAdapter đâu đó ở khắp nơi trong mã lệnh, bây giờ ứng dụng có nhu cầu chuyển đổi sang Oracle, bạn phải đi tìm tất cả các lớp Provider của SQL Server đổi lại thành OracleConnection, OracleAdapter.

Giải pháp thường được nhà lập trình ADO.NET 1.1 sử dụng là chọn Provider OLEDB, vì trình điều khiển này hầu như được hỗ trợ bởi rất nhiều Database do quá trình lịch sử lâu dài của nó. Tuy nhiên, truy xuất qua OLEDB sẽ không tối ưu về tốc độ vì với ADO.NET Provider đặc thù như Sql hay Oracle, bạn tận dụng và truy xuất trực tiếp vào được thư viện Native do cơ sở dữ liệu cung cấp.

Có một cách thông minh hơn đó là sử dụng lệnh Select Case để lực chọn trường hợp nào thì sử dụng Provider thích hợp.

5. Giới thiệu ADO.NET 2.0

Mặc dù kiến trúc của ADO.NET phiên bản 1.1 được xem là khá tốt nhưng cũng như bất cứ sản phẩm phần mềm nào khác đều cần phải cải tiến không ngừng sau quá trình sử dụng. Nhận được rất nhiều phản hồi từ phía các nhà phát triển, Microsoft đã cải tạo và thêm rất nhiều tính năng mới cho kiến trúc ADO.NET trong phiên bản .NET Framework 2.0. Một điểm mốc quan trọng là giờ đây ngôn ngữ và môi trường thực thi CLR của .NET đã được đưa chung vào với SQL Server

2005. Nhiều namespace mới được thêm vào ADO.NET cho phép tương tác từ bên ngoài và bên trong thủ tục Store Procudure của SQL Server 2005. Có thể nói những thay đổi trong ADO.NET rất đáng giá và khẳng định chiến lược tiếp theo của Microsoft về mô hình truy cập dữ liệu là rất nhất quán. Nếu đã quen với ADO.NET 1.1 thì bạn sẽ thấy ADO.NET 2.0 là sản phẩm hoàn thiện, khắc phục tất cả những thiếu sót của phiên bản 1.1.

5.1. Lớp DbProviderFactory - Viết các ứng dụng độc lập Provider

Như đã nêu, việc đưa ra rất nhiều các Provider truy xuất dữ liệu như SqlConnection, OracleConnection, OledbConnection, OdbcConnection đã khiến cho các ứng dụng .NET khó chuyển đổi chạy độc lập trên nhiều nền Database khác nhau. Với ADO.NET 2.0, bạn vẫn có thể sử dụng các Provider độc lập, nhưng nếu muốn có một giải pháp tổng quát linh động hơn, hãy sử dụng lớp DbProviderFactory.

Ví dụ sau sẽ sử dụng lớp DbProviderFactory tạo đối tượng kết nối Connection và điều phối dữ liệu DataAdapter.

Khi muốn thay đổi Provider, bạn chỉ cần thay đổi tên của ProviderName là đủ. Ví dụ chuyển toàn bộ hệ thống sang cơ sở dữ liệu Oracle, bạn chỉ cần định nghĩa lại:

Dim ProviderName As String = “System.Data.Oracle”

5.2. Xử lý bất đồng bộ (Asynchronous)

Với những ứng dụng truy xuất dữ liệu phức tạp, khi bạn phát đi một câu lệnh SQL, quá trình xử lý là đồng bộ. Có nghĩa là ứng dụng sẽ chờ cho đến khi nhận được kết quả trả về từ cơ sở dữ liệu thì mới thực thi lệnh tiếp theo. Sẽ có những trường hợp vì lí do nào đó, quá trình xử lý diễn ra rất lâu trên Server trên máy chủ cơ sở dữ liệu đang hoạt động, khi đó hoặc ứng dụng của bạn sẽ gần như “treo” đứng im không hoạt động, hoặc sau một thời gian chờ ADO.NET sẽ phát sinh lỗi Exception timeout. ADO.NET 2.0 cho phép bạn thực hiện những thao tác không đồng bộ (Asynchronous) trên cơ sở dữ liệu. Bạn gửi đi lệnh SQL đến Server

và chương trình sẽ chuyển ngay sang thực thi lệnh kế tiếp. Khi Server xử lý xong và trả về kết quả, ADO.NET sẽ thông báo để bạn biết và xử lý tiếp. Các phương thức như BeginExecuteReader (bắt đầu thực thi lệnh) hay EndExecuteReader (thông báo lệnh kết thúc) sẽ thực hiện công việc này.

5.3. Sao chép tập dữ liệu lớn (Bulk Copy)

Phiên bản mới của ADO.NET 2.0 cho phép bạn gửi một lệnh đến Server thực hiện thao tác sao chép tập dữ liệu lớn nhe backup hay chuyển dữ liệu giữa các bảng. Tính năng này chỉ áp dụng cho trình Provider của cơ sở dữ liệu SQL Server, nó tương đương với lệnh bcp của SQL Server. Bạn sử dụng đối tượng SqlBulkCopy để thực hiện công việc này.

5.4. Multiple Active Result (MARS)

Trước đây, mỗi lần bạn chỉ có thể thực thi và lấy về một tập kết quả (tương đương với một bảng dữ liệu). Với ADO.NET 2.0, bạn có thể gửi đi cùng lúc nhiều lệnh SQL và mở nhiều đối tượng DataReader trên một kết nối với nhiều đối tượng Command khác nhau.

Như vây, thay vì trước đây phải gửi đi 2 lệnh SELECT khác nhau lần lượt đến Server để lấy về danh sách của Shippers và Employees bây giờ bạn chỉ cần gửi đi một câu lệnh. Hãy hình dung nếu gửi đi 10 câu lệnh 1 lần và lấy về kết quả cùng lúc sẽ hiệu quả hơn cơ chế ADO.NET trước đây gửi 10 câu lệnh tuần tự.

5.5. Liệt kê danh sách Server

Nếu bạn chọn cơ sở dữ liệu SQL Server 2005 để phát triển ứng dụng thì ADO.NET 2.0 là một lựa chọn hoàn hảo. Một trong những điểm mới trong phiên bản ADO.NET 2.0 là khả năng liệt kê danh sách các đối tượng (Instance) của Microsoft SQL Server 2000 và Microsoft SQL Server 2005. Nếu viết những ứng dụng của hệ thống, bạn sẽ thấy những tính năng này rất cần thiết vì nó cho phép người dùng được quyền thay đổi và lựa chọn những instance có sẵn của SQL Server. Bạn sử dụng lớp SqlDataSourceEnumerator để lấy về danh sách các instance như sau:

Lấy về bảng danh sách các instance của SQL Server

Dim dtServers As DataTable=

SqlDataSourceEnumerator.Instance.GetDataSource

5.6. Đọc và thống kê thông tin kết nối

Bạn sử dụng lớp SqlConnection.RetrieveStatistics để đọc thông tin kết nối hiện hành của cơ sở dữ liệu SQL Server. Thông tin kết nối rất có ích khi bạn muốn biết Server đang làm việc trong trạng thái nào, có bao nhiêu user đang kết nối, hệ thống đang bận xử lý hay rảnh rỗi. Dưới đây là ví dụ về cách lấy thông tin thống kê.

5.7. Xử lý theo lô (Batch Update)

ADO.NET 1.1 sử dụng phương thức .Update() của SqlDataAdapter để cập nhật dữ liệu trong DataSet. Mỗi thao tác như cập nhật, tạo mới hay xoá bản tin đều được SqlDataAdapter sinh ra một câu lệnh SQL và gửi đến máy chủ cơ sở dữ liệu. Hiệu suất thực thi sẽ giảm nếu cùng lúc bạn yêu cầu cập nhật DataSet với số lượng mẩu tin lớn. Ví dụ có 100 mẩu tin cần cập nhật thì sẽ có 100 lệnh SQL gửi đến Server 100 lần. ADO.NET 2.0 cung cấp thêm thuộc tính .UpdateBatchSize cho SqlDataAdapter. Bạn có thể xác định kích thước . UpdateBatchSize là số lượng câu lệnh cập nhật SQL sẽ gửi đi 1 lần. Mặc định . UpdateBatchSize là 1. Để tăng tốc độ xử lý theo lô, bạn có thể chỉ định con số này là 10, 20, hay 50. Ví dụ, bảng dữ liệu đơn hàng Orders với 800 dòng cần cập nhật nếu . UpdateBatchSize = 1, thời gian thực thi để cập nhật toàn bộ dữ liệu khoảng 8.4 giây, nếu UpdateBatchSize = 50, thời gian này là 3.6 giây; nếu UpdateBatchSize = 11, thời gian cập nhật là 4.8 giây. Điều chirnh con số này tuỳ theo tính biến động thường xuyên của dữ liệu bạn cần cập nhật.

5.8. Nâng cấp thao tác dữ liệu từ đối tượng DataSet

Đối tượng DataTableReader trình bày tập dữ liệu chỉ đọc một chiều tạo được ra từ đối tượng DataSet hay DataTable. Đối tượng DataTable giờ đây cho

phép bạn đánh chỉ mục để tăng tốc độ khi thực hiện các thao tác: insert, delete, update của dữ liệu trong đối tượng DataRow.

5.9. Serialize đối tượng DataSet

Đặc điểm mới này sẽ cho phép đối tượng DataSet và DataTable mã hoá (serialize) với định dạng nhị phân khi truyền hai đối tượng này qua mạng bằng hình thức truy cập từ xa (Remoting).

Chương 14
LẬP TRÌNH SQL SERVER VÀ XÂY DỰNG ỨNG DỤNG

MỤC TIÊU CỦA CHƯƠNG

- Về kiến thức

+ Củng cố lại các lệnh SQL thao tác với cơ sở dữ liệu

+ Giúp sinh viên nắm được cấu trúc và mô hình hoạt động của ADO.NET, các đặc trưng cơ bản của ADO.NET

+ Trang bị cho sinh viên khái niệm về các thuộc tính, phương thức hoạt động của một số đối tượng chính tương tác với cơ sở dữ liệu như Connection, Command, DataReader, DataAdapter, DataSet và DataTable.

+ Giúp sinh viên làm quen với các form kết nối, hiển thị, cập nhật, lọc dữ liệu với các cơ sở dữ liệu Acess và SQL.

- Về kỹ năng

+ Sinh viên biết cách sử dụng các đối tượng của ADO.NET để thiết kế các form liên kết project với cơ sở dữ liệu.

+ Biết cách vận dụng các câu lệnh SQL trong các tình huống xử lý cụ thể.

+ Có thể tự thiết lập các form đăng nhập, tìm kiếm, hiển thị, cập nhật, lọc dữ liệu bằng nhiều công cụ khác nhau.

- Về thái độ:

Giúp sinh viên có hứng thú với các phần mềm cơ sở dữ liệu; có kiến thức thực tế; có tự tin về khả năng xây dựng hoặc tham gia trong nhóm xây dựng một hệ thống quản trị cơ sở dữ liệu.

NỘI DUNG BÀI GIẢNG LÝ THUYẾT

Trong ví dụ 5.1 và 5.2, chúng ta đã kết nối với cơ sở dữ liệu SQL có tên De1 trên server HTC-VAIO\Huyen. Cơ sở dữ liệu này gồm 2 bảng:

SanPham

MaSP (mã sản phẩm	TenSP (tên sản phẩm)	NhaCC (nhà cung cấp)	DonGia (đơn giá)
1	LCD Toshiba 32 inch	Toshiba	6690000
2	LCD Toshiba 40 inch	Toshiba	10900000
3	LCD LG 32 inch	LG	5290000
4	LCD Sony 40 inch	Sony	14900000
5	LED Samsung 22 inch	Samsung	5790000

BanHang

SoHD (số hóa đơn)	NgayBan (ngày bán)	MaSP (mã sản phẩm)	Ngươiban (người bán)	SoLuong (số lượng)	ThanhTien (thành tiền)
1	'2/1/2011'	1	An	2	
2	'2/1/2011'	2	Bình	3	
3	'2/1/2011'	3	Vân	2	
4	'2/3/2011'	4	Bình	3	
5	'3/1/2011'	5	An	1	
6	'3/1/2011'	1	An	1	
7	'3/2/2011'	1	Vân	2	
8	'4/1/2011'	2	Vân	2	
9	'4/1/2011'	3	An	2	
10	'4/1/2011'	3	An	1	
11	'4/1/2011'	4	Bình	5	
12	'4/1/2011'	4	Vân	3	

- Đếm số sản phẩm của Toshiba: (sử dụng thủ tục Tao_ket_noi() đã định nghĩa trong ví dụ 5.2)

```
  Tao_ket_noi()
  Ket_noi.Open()
  Dim lenhsql1 As String = "select count(masp) from sanpham " &
_
  where NhaCC='TOSHIBA'"
  Dim cmd1 As New SqlClient.SqlCommand(lenhsql1, Ket_noi)
  MsgBox(cmd1.ExecuteScalar)
  Ket_noi.Close()
```

- Cập nhật thành tiền bằng đơn giá nhân số lượng

```
  Tao_ket_noi()
  Ket_noi.Open()
  Dim lenhsql2 As String
  Lenhsql2 = "update banhang set thanhtien = soluong * " &_
  "(select dongia from sanpham where banhang.masp=sanpham.masp)"
  Dim cmd2 As New SqlClient.SqlCommand(lenhsql2, Ket_noi)
  Cmd2.ExecuteNonQuery()
  Ket_noi.Close()
```

- Bổ sung vào bảng Sanpham một bản ghi mới với mã sản phẩm là 6, tên sản phẩm là LCD Sony 32inch, hãng sản xuất SONY, đơn giá 10900000.

```
  Tao_ket_noi()
  Ket_noi.Open()
  Dim lenhsql3 As String = "insert sanpham values " & _
  "(6,'LCD Sony 32 inch', 'SONY',10900000)"
  Dim cmd3 As New SqlClient.SqlCommand(lenhsql3,
Ket_noi)
  Cmd3.ExecuteNonQuery()
  Ket_noi.Close()
```

5.3.4 Parameter

a. Khai báo và sử dụng tham số

Trong CommandText có thể sử dụng các tham số để thay thế cho các giá trị chưa xác định và khi thực hiện sẽ dùng đối tượng Parameter để truyền giá trị vào chỗ các dấu hỏi. Tùy theo Command, Parameter sẽ khai báo từ lớp OledbParameter hay SqlParameter. Tại vị trí cần thay bằng tham số sẽ để dấu ? nếu là lớp OledbParameter và để @tên_biến_thamsố nếu là SqlParameter

Các cú pháp khai báo tham số:

```
Dim <tên parameter> As New loại_tham_số()
```

Trong đó loại_tham_số là `OledbParameter` hoặc `SqlParameter`

Hoặc

```
Dim <tên parameter> As New loại_tham_số(tên_ts)
```

Hoặc

```
Dim <tên parameter> As New loại_tham_số (tên_ts, giá trị)
```

Hoặc

```
Dim <tên parameter> As New loại_tham_số =
biến_command.CreateParameter
```

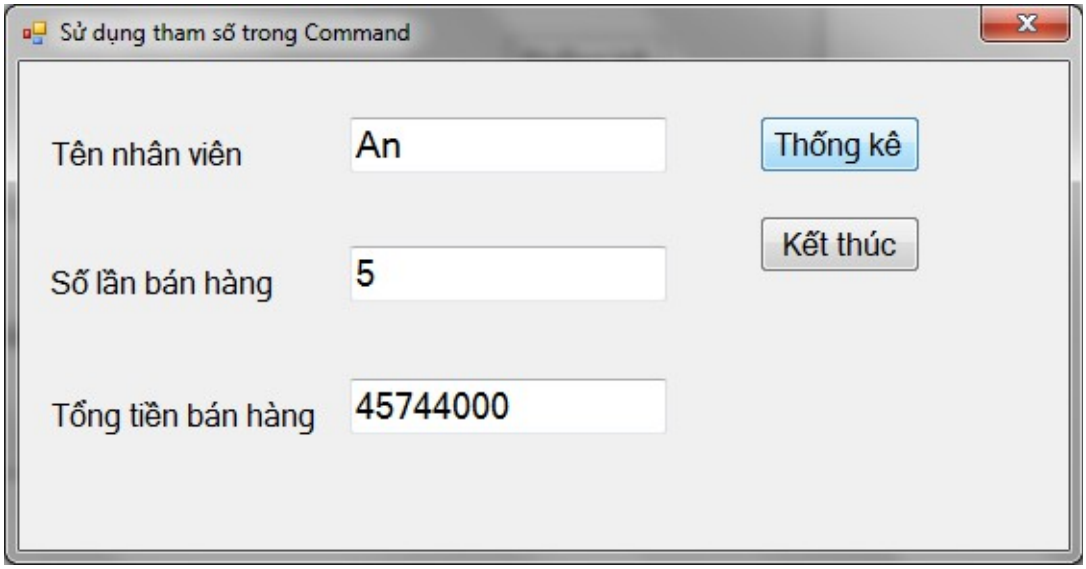
Các thuộc tính của tham số

Tên	**Mô tả**
Direction	Giá trị cho biết loại tham số với các giá trị sau: (đọc ghi) ? Input (mặc định): loại tham số đầu vào ? InputOutput: loại tham số vào và ra ? Output: loại tham số đầu ra ? ReturnValue: loại tham số nhận giá trị trả về của một thủ tục nội, một hàm, hay một hàm do người dùng định nghĩa.
OleDbType SqlDbType	Kiểu dữ liệu OleDb hoặc SqlDb của tham số (đọc ghi)
ParameterName	Tên tham số (đọc ghi)

Value	Giá trị của tham số (đọc ghi)

Ví dụ 5.4: Sử dụng tham số trong command

Thiết kế form như hình dưới đây. Nút Thống kê kết nối đến cơ sở dữ liệu SQL De1 trên server HTC-VAIO\Huyen và hiển thị số lần bán hàng trong textbox `txtcount` và tổng số tiền bán hàng của nhân viên trong textbox `txtsum`. Tên nhân viên (`textbox1`) được định nghĩa là tham số.



Tạo tham số với Command để hiển thị số lần bán hàng:

```
' tạo và mở connection
  Tao_ket_noi()
  Ket_noi.Open()
' hiển thị số lần bán hàng
  Dim lenhsql as string = "select count(sohd) from banhang " & _
  "where nguoiban=@nb"
  Dim cmdcount As New SqlClient.SqlCommand(lenhsql,Ket_noi)
  Dim thamsonb As SqlClient.SqlParameter =
cmdcount.CreateParameter
  thamsonb.ParameterName = "@nb"
  thamsonb.Value = TextBox1.Text
  cmdcount.Parameters.Add(thamsonb)
  txtcount.Text = cmdcount.ExecuteScalar()
' đóng kết nối
  Ket_noi.Close()
```

Ta có thể viết đoạn mã tương tự để hiển thị tổng số tiền bán hàng với tên nhân viên là tham số. Hoặc sử dụng cách viết như sau: (dùng chung tham số @nb cho đối tượng command có command text thay đổi theo câu lệnh SQL tương ứng).

```
 ' tạo và mở connection
  Tao_ket_noi()
  Ket_noi.Open()
' hiển thị số lần bán hàng
  Dim cmd As New SqlClient.SqlCommand
  cmd.CommandText = "select count(sohd) from banhang where " & _
  "nguoiban=@nb"
  cmd.Connection = Ket_noi
  Dim thamsonb As SqlClient.SqlParameter = cmd.CreateParameter
  thamsonb.ParameterName = "@nb"
  thamsonb.Value = TextBox1.Text
  cmd.Parameters.Add(thamsonb)
  txtcount.Text = cmd.ExecuteScalar()
' hiển thị tổng số tiền bán hàng
  cmd.CommandText = "select sum(thanhtien) from banhang where " &
_
  "nguoiban=@nb"
  txtsum.Text = cmd.ExecuteScalar()
' đóng kết nối
  Ket_noi.Close()
```

Khi sử dụng OLEDBCommand thì câu lệnh SQL thay đổi như sau:

```
lenh.CommandText = "Select sum(thanhtien) from banhang " & _
where nguoiban = ?"
Dim thamso As OleDbParameter = lenh.CreateParameter()
thamso.Value = "An"
lenh.Parameters.Add(thamso)
```

b. Đưa tham số trực tiếp vào tập hợp Parameters

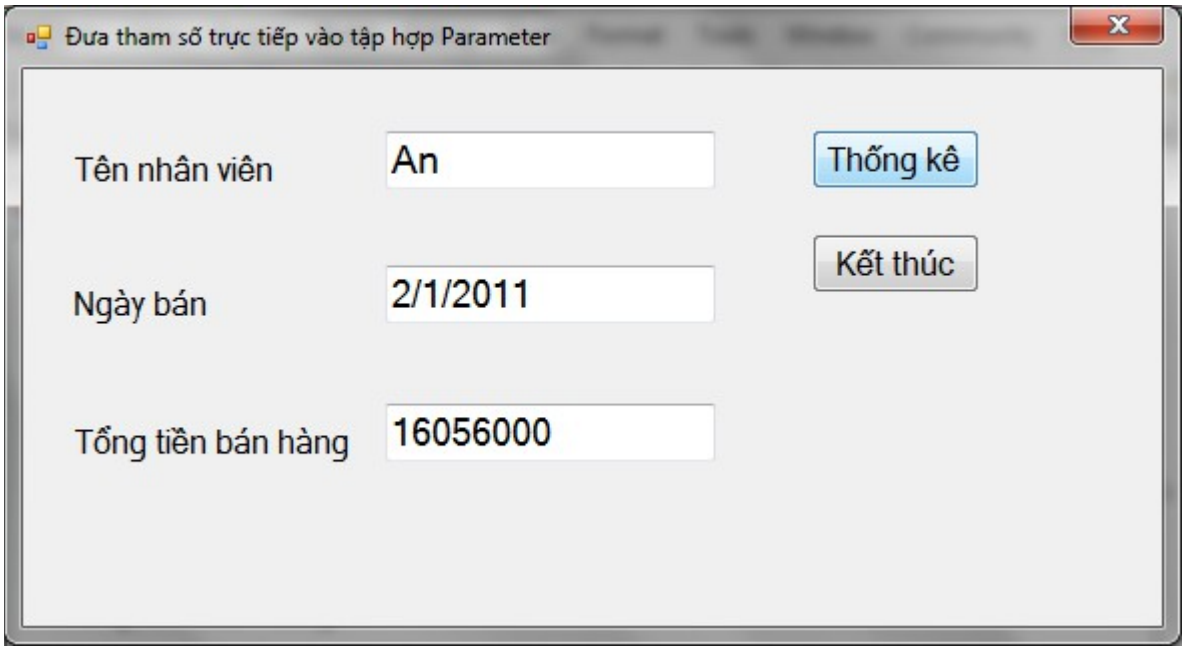
Có thể đưa nhiều tham số trực tiếp vào tập hợp Parameter. Mỗi tham số được đưa vào theo cú pháp như sau:

```
Dim tên_thamsố as Loại_thamsố = _
tên_command.CreateParameter(tên_thamsố, kiểu_dữ_liệu, kích_cỡ)
Tên_thamsố.Value = giá_trị
```

Nếu là OLEDBParameter thì danh sách tham số đưa vào phải đúng theo thứ tự của danh sách tham số trong câu lệnh. Nếu là SQL thì không cần theo đúng thứ tự.

Ví dụ 5.5: Hiển thị số tiền bán hàng của một nhân viên trong một ngày nào đó:

Form được thiết kế như mẫu sau. Tên nhân viên được nhập vào hộp txtten; ngày bán nhập vào hộp txtngay, tổng tiền bán hàng được hiển thị trong txtsum.



Biến cố Click của nút Thống kê

```
Tao_ket_noi()
Ket_noi.Open()
Dim cmd As New SqlClient.SqlCommand

cmd.CommandText = "select sum(thanhtien) from banhang " & _
"where nguoiban=@nb and ngayban=@ngay"

cmd.Connection = Ket_noi

Dim thamsonb As SqlClient.SqlParameter = _
cmd.Parameters.Add("@nb", SqlDbType.NVarChar, 50)
thamsonb.Value = txtten.Text

Dim thamsongay As SqlClient.SqlParameter = _
cmd.Parameters.Add("@ngay", SqlDbType.DateTime)
thamsongay.Value = CDate(txtngay.Text)

txtsum.Text = cmd.ExecuteScalar()

Ket_noi.Close()
```

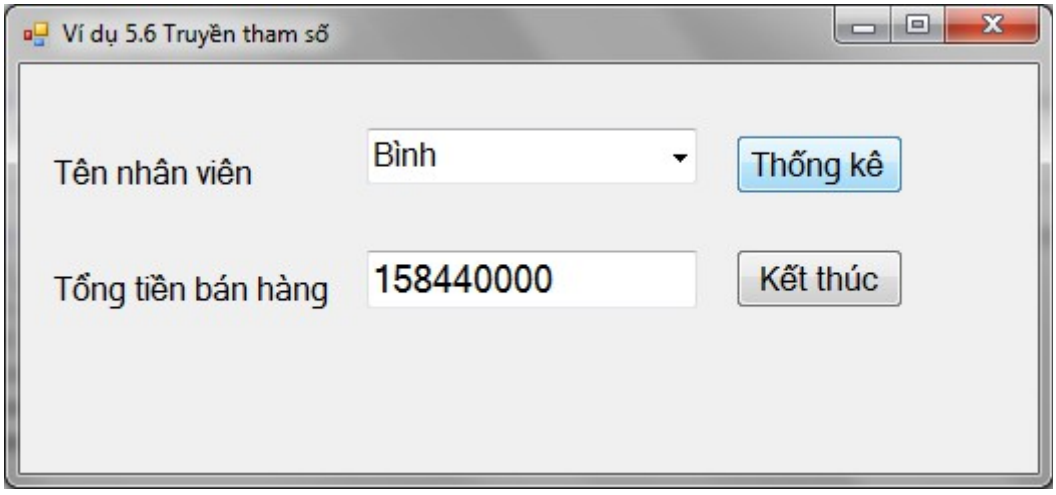
c. Truyền tham số

Ví dụ 5.6: Giả sử trong cơ sở dữ liệu De1, ta đã tạo một thủ tục nội tại trả về tổng số tiền bán được của một nhân viên. Thủ tục được tạo như sau:

```
create function ThongKeNV (@tnb nvarchar(50)) returns decimal
as
begin
  declare @tt decimal
  select @tt= sum(thanhtien) from banhang where nguoiban=@tnb
  return (@tt)
end
```

Thủ tục có một tham số đầu vào là @tnb thuộc kiểu nvarchar(50) tương ứng với tên một nhân viên bán hàng và trả về giá trị thuộc kiểu decimal tương ứng với tổng số tiền bán được của nhân viên đó. Ta sẽ xây dựng chương trình sử dụng đối tượng command với 2 tham số được truyền vào (1 tham số trả về và một tham số đầu vào). Tham số trả về phải được truyền cho command trước tiên.

Form được thiết kế như hình dưới đây. Tên nhân viên được chọn trong combo box (`cboten`). Nút Thống kê dùng để gọi hàm `thongkenv` đã tạo trong SQL và hiển thị kết quả trong textbox `txtsum`.



Biến cố Click của nút Thống kê:

```
' hiển thị số lần bán hàng
  Tao_ket_noi()
  Ket_noi.Open()

' tạo command có tên cmd tương ứng với thủ tục thongkenv
  Dim cmd As New SqlClient.SqlCommand
  cmd.Connection = Ket_noi
  cmd.CommandType = CommandType.StoredProcedure
```

```
  cmd.CommandText = "thongkenv"
' tạo tham số sotien là tham số trả về
  Dim sotien As New SqlClient.SqlParameter
  sotien.Direction = ParameterDirection.ReturnValue
  sotien.SqlDbType = SqlDbType.Decimal
  cmd.Parameters.Add(sotien)
' tạo tham số thamsonb là tham số đầu vào ứng với người bán
  Dim thamsonb As SqlClient.SqlParameter = _
  cmd.Parameters.Add("@tnb", SqlDbType.NVarChar, 50)
' thiết lập giá trị cho thamsonb
  thamsonb.Value = cboten.Text
' thực thi command
  cmd.ExecuteScalar()
' hiển thị giá trị tham số
  txtsum.Text = sotien.Value
' đóng kết nối
  Ket_noi.Close()
```

5.4 DataReader

Là đối tượng truy cập dữ liệu trực tiếp, sử dụng con trỏ phía Server và duy trì kết nối với Server trong suốt quá trình đọc dữ liệu, DataReader thuộc không gian tên System.Data.OledbDataReader hoặc System.Data.SqlDataReader

5.4.1 Các thuộc tính của DataReader

Tên	**Mô tả**
FieldCount	Trả về số cột trên dòng hiện hành của DataReader.
IsClosed	Cho biết DataReader đã đóng chưa.
Item	Trị của cột truyền vào. Tham số truyền có thể là tên cột hoặc số thứ tự (từ 0)

5.4.2 Các phương thức của DataReader

Tên	**Mô tả**
Close	Đóng DataReader.
GetFieldType	Trả về kiểu dữ liệu của cột truyền vào.
GetName	Trả về tên của cột truyền vào.
GetOrdinal	Trả về số thứ tự của cột truyền vào (bắt đầu từ 0).
GetSchemaTable	Trả về bảng chứa thông tin mô tả cột của DataReader.

GetValue	Trả về giá trị trên cột truyền vào.
Read	Di chuyển đến dòng kế tiếp và trả về True nếu còn dòng để di chuyển, ngược lại trả về False

Trong khi DataReader đang mở, các thao tác dữ liệu khác trên nguồn dữ liệu đều không thể thực hiện cho đến khi DataReader đóng lại bằng lệnh Close.

5.4.3 Tạo và sử dụng DataReader

DataReader được tạo với cách duy nhất là gọi phương thức ExcuteReader của đối tượng Command.

Cú pháp:

```
Tên_command.ExecuteReader()
```

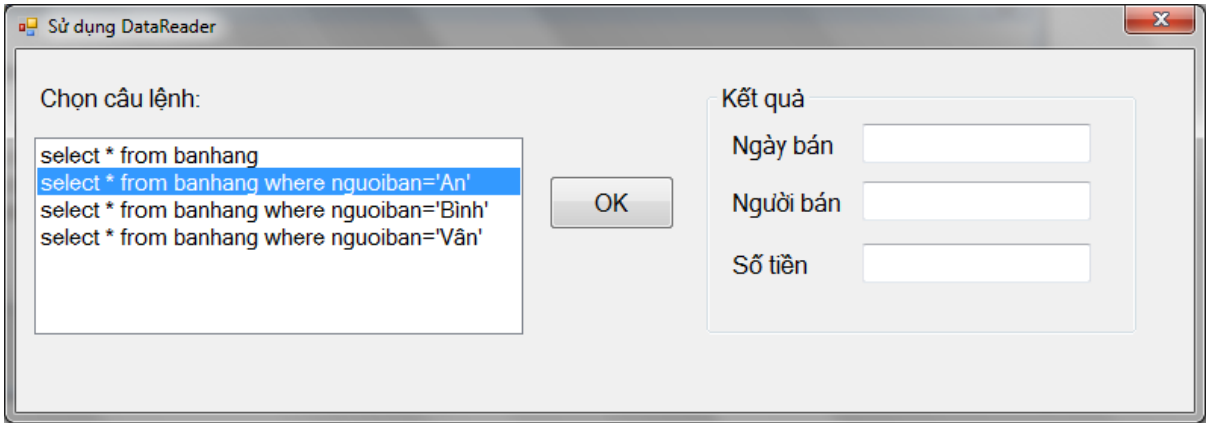
Hoặc

```
Tên_command.ExecuteReader(<behavior>)
```

Các giá trị của behavior:

Giá trị	**Mô tả**
CloseConnection	Khi đối tượng DataReader đóng lại, Connection tự động đóng theo.
Default	Tương tự như khi gọi ExecuteReader()
SchemaOnly	Truy vấn chỉ trả về cấu trúc các cột và không làm ảnh hưởng đến các thao tác khác.

Ví dụ 5.7 Thiết kế form như hình sau:



Form sử dụng DataReader để lưu trữ kết quả truy vấn được chọn trong Listbox1. Nút OK để thực hiện truy vấn. Nếu còn dữ liệu chưa được đọc trong datareader thì nút OK chuyển thành nút Tiếp để xem các bản ghi còn lại. Nếu dữ liệu đã được đọc hết thì các hộp textbox sẽ được xóa trắng và nút Tiếp lại trở thành nút OK.

- Khai báo các biến chung của class: (đặt ngay sau dòng Public Class, bên ngoài tất cả các Sub).

```
Dim cmd As New SqlClient.SqlCommand
Dim reader1 As SqlClient.SqlDataReader
```

- Gọi thủ tục Tao_Ket_Noi() trong sự kiện Load của Form

```
Tao_ket_noi()
```

- Đóng kết nối và đóng datareader trong sự kiện FormClosed

```
If Ket_noi.State = ConnectionState.Open Then
    reader1.Close()
End If
```

- Sự kiện Click của cmdOK

```
If ListBox1.SelectedItem Is Nothing Then
   MsgBox("Hãy chọn một câu lệnh")
Else
   ' tạo command cmd
   cmd.CommandText = ListBox1.SelectedItem
   cmd.Connection = Ket_noi

   ' mở kết nối và tạo reader nếu kết nối đang đóng
   If Ket_noi.State = ConnectionState.Closed Then
      Ket_noi.Open()
      reader1 = _
      cmd.ExecuteReader(CommandBehavior.CloseConnection)
   End If

   ' đọc dữ liệu trong reader1 vào textbox
   If reader1.Read() Then
      cmdOK.Text = "Tiếp"
      txtngay.Text = reader1.Item("ngayban")
      txtten.Text = reader1.Item("nguoiban")
      txtst.Text = reader1.Item("thanhtien")
   Else

      ' đọc hết reader1
      cmdOK.Text = "OK"
      txtten.Clear()
      txtst.Clear()
      reader1.Close()
   End If

   My.Application.DoEvents()
End If
```

5.5 DataAdapter

Để lấy dữ liệu từ nguồn dữ liệu về cho ứng dụng, chúng ta sử dụng một đối tượng gọi là DataAdapter. Đối tượng này cho phép lấy cấu trúc và dữ liệu của các bảng trong nguồn dữ liệu.

DataAdapter là một bộ gồm bốn đối tượng Command:

SelectCommand: cho ph ép lấy thông tin từ nguồn dữ liệu về

InsertCommand: cho ph ép thêm dữ liệu vào bảng trong nguồn dữ liệu.

UpdateCommand: cho ph ép sửa đổi dữ liệu trên bảng trong nguồn dữ liệu.

DeleteCommand: cho ph ép hủy bỏ dữ liệu trên bảng trong nguồn dữ liệu.

Thông thường, chúng ta chỉ cần khai báo nội dung lệnh cho SelectCommand, các nội dung của các command còn lại có thể được phát sinh nhờ đối tượng CommandBuilder dựa vào nội dung của SelectCommand.

5.5.1 Tạo DataAdapter

Cũng như Command, chúng ta cần khai báo rõ DataAdapter sử dụng theo Data Provider nào: SqlDataAdapter hoặc OledbDataAdapter. Hai lớp này thuộc không gian tên:

System.Data.OleDb.OleDbDataAdapter

System.Data.SqlClient.SqlDataAdapter

Để tạo DataAdapter ta có thể dùng các cú pháp sau:

```
New <loại>DataAdapter()
```

Hoặc

```
New <loại>DataAdapter(đối_tượng_SelectCommand)
```

với đối_tượng_SelectCommand đã định nghĩa sẵn.

Hoặc

```
New <loại>DataAdapter(lệnhSQL, đối_tượng_Connection)
```

Hoặc

```
New <loại> DataAdapter
(lệnhSQL,đối_tượng_ConnectionString)
```

Ví dụ: Để tạo DataAdapter lấy dữ liệu trong bảng VatTu của cơ sở dữ liệu Access có tên C:\QlHanghoa.mdb ta có thể sử dụng các lệnh dưới đây:

Mẫu 1:

```
Dim bo_doc_ghi As New OledbDataAdapter()
bo_doc_ghi.SelectCommand.CommandText = _
"Select * from VATTU"
```

```
bo_doc_ghi.SelectCommand.Connection.ConnectionString = _
"Provider = Microsoft.Jet.OLEDB.4.0; Data Source =
C:\QlHanghoa.mdb"
```

Mẫu 2:

```
Dim lenh As New OledbCommand("Select * from VATTU", _
"Provider = Microsoft.Jet.OLEDB.4.0; Data Source = " & _
C:\QlHanghoa.mdb")
Dim bo_doc_ghi As New OledbDataAdapter(lenh)
```

Mẫu 3:

```
Dim ketnoi As New OledbConnection("Provider=Microsoft.Jet.OLEDB.4.0;" &
_
"Data Source = C:\QlHanghoa.mdb")
Dim bo_doc_ghi As New OledbDataAdapter("Select * from VATTU", ketnoi)
```

Hoặc mẫu 4:

```
Dim chuoi As String = "Provider = Microsoft.Jet.OLEDB.4.0; " &
_
"Data Source= C:\QlHanghoa.mdb"
Dim bo_doc_ghi As New OledbDataAdapter("Select * from VATTU", _
chuoi)
```

Để tạo DataAdapter lấy dữ liệu từ bảng Banhang của cơ sở dữ liệu SQL có tên De1 trên server HTC-VAIO\Huyen ta có thể sử dụng đoạn mã dưới đây:

```
Dim chuoi As String = "server = HTC-VAIO\Huyen; " & _
"database=De1;integrated security=SSPI"
Dim lenhsql = "select * from banhang"
Dim adapter1 As New SqlClient.SqlDataAdapter(lenhsql, chuoi)
```

Nếu trong Module1 đã định nghĩa chuoi_ket_noi là biến tầm vực module (ví dụ 5.2b) thì ta có thể viết lại đoạn lệnh trên như sau:

```
Dim lenhsql = "select * from banhang"
Dim adapter1 As New SqlClient.SqlDataAdapter(lenhsql, _
chuoi_ket_noi)
```

Hoặc tạo DataAdapter thông qua kết nối Connection đã được tạo với thủ tục Tao_ket_noi() trong ví dụ 5.3:

```
Tao_ket_noi()
If Ket_noi.State = ConnectionState.Closed Then Ket_noi.Open()
```

```
Dim lenhsql As String = "select * from banhang"
Dim adapter1 As New SqlClient.SqlDataAdapter(lenhsql, Ket_noi)
```

DataAdapter chỉ thao tác được với nguồn dữ liệu qua một đối tượng Connection đang kết nối, nhưng điểm đặc biệt của DataAdapter là khi Connection chưa mở, DataAdapter sẽ tự động mở kết nối khi cần và tự động đóng lại, ngược lại, với một Connection đã mở sẵn, chúng ta phải tự đóng kết nối, DataAdapter không thực hiện đóng tự động.

5.5.2 Thuộc tính của DataAdapter

Tên	Mô tả
ContinueUpdateOnError	Chỉ định cách thức xử lý khi DataAdapter cập nhật dữ liệu về nguồn và bị lỗi. Nếu là True, DataAdapter bỏ qua dòng bị lỗi và cập nhật các dòng kế tiếp. Lỗi phát sinh sẽ được đưa vào thuộc tính RowError của dòng bị lỗi, ngược lại False, sẽ phát sinh Exception khi dòng bị lỗi.
DeleteCommand	Đối tượng Command chứa nội dung lệnh hủy các mẩu tin trên nguồn dữ liệu.
InsertCommand	Đối tượng Command chứa nội dung lệnh chèn các mẩu tin mới vào nguồn dữ liệu.
SelectCommand	Đối tượng Command chứa nội dung lệnh truy xuất các mẩu tin từ nguồn dữ liệu.
TableMappings	Tập hợp các ánh xạ tên bảng khi DataAdapter đổ dữ liệu hay cấu trúc vào đối tượng chứa.
UpdateCommand	Đối tượng Command chứa nội dung lệnh cập nhật các mẩu tin vào nguồn dữ liệu.

5.5.3 Chức năng của DataAdapter

a. Đổ dữ liệu vào Dataset hoặc dataTable

Sau khi có đối tượng DataAdapter với nội dung SelectCommand và thông tin về kết nối, chúng ta có thể sử dụng DataAdapter để lấy dữ liệu về cho các đối tượng chứa dữ liệu như DataTable, DataSet qua phương thức Fill. Phương thức trả về số mẩu tin lấy về được.

- Đổ dữ liệu vào DataTable có sẵn:

```
Fill(tên_datatable)
```

- Đổ dữ liệu vào DataSet có sẵn. Dữ liệu được lấy về Dataset dưới dạng các DataTable, với tên mặc định là Table,Table1,Table2,…

```
Fill(tên_dataset)
```

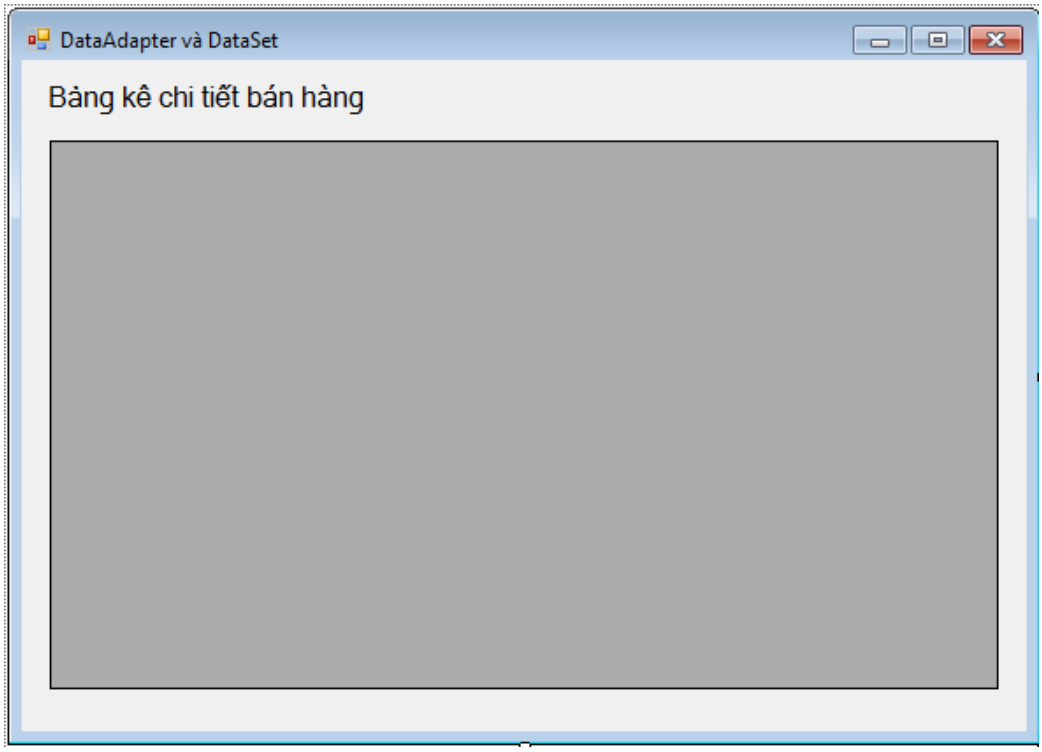
▪ Đổ dữ liệu vào DataSet cho bảng <tên_ DataTable>; nếu chưa có, bảng sẽ được tạo.

```
Fill(tên_dataset, tên_DataTable)
```

Trong đó tên_dataTable không nhất thiết phải trùng với tên của bảng trong cơ sở dữ liệu.

Ví dụ 5.8a: Sử dụng DataAdapter và DataSet để hiển thị thông tin về các hóa đơn bán hàng.

Trong ví dụ này ta thiết kế form chứa hai đối tượng. Một label với dòng text là Bảng kê chi tiết bán hàng và một đối tượng DataGridView. Ban đầu khi thêm DataGridView vào ta được form như hình thu nhỏ dưới đây:



Phần code của chương trình được viết như sau:

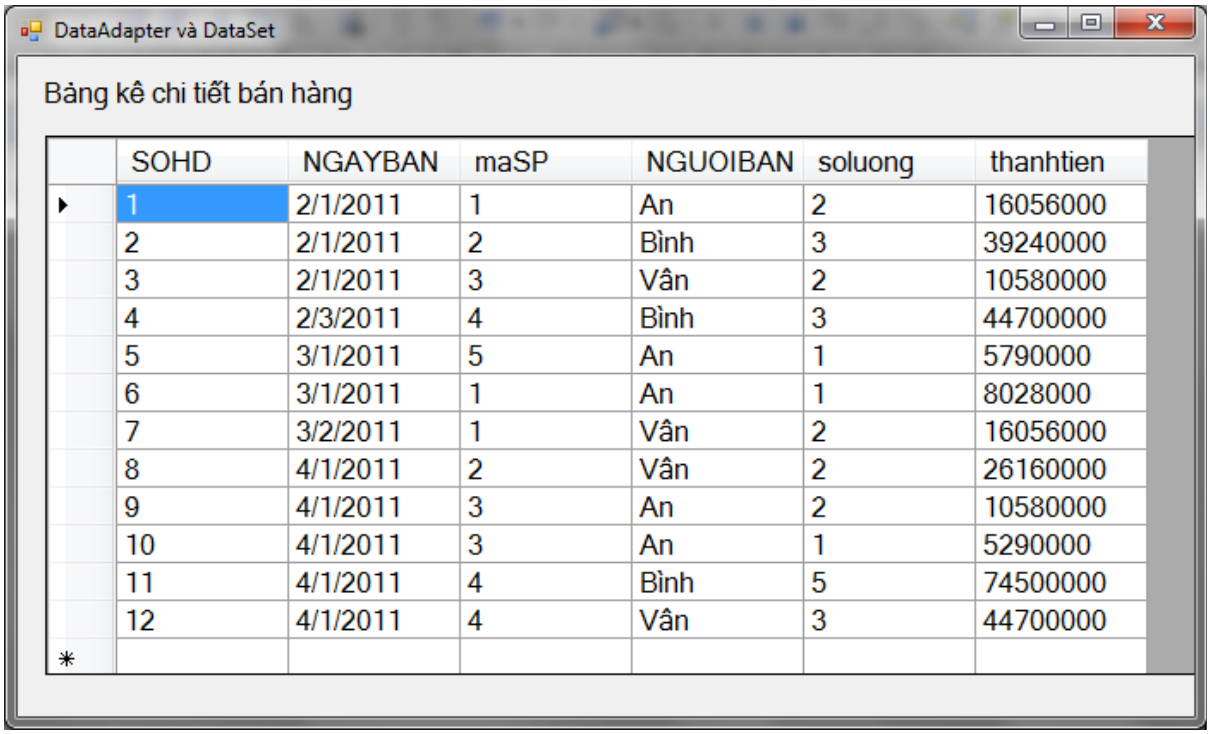
```
' khai báo namespace của dataset
Imports System.Data
```

Thủ tục Load của form:

```
Tao_ket_noi()
if Ket_noi.State = ConnectionState.Closed Then
   Ket_noi.Open()
End if
' tạo dataAdapter
Dim lenhSql = "select * from banhang"
```

```
Dim adapter1 As New SqlClient.SqlDataAdapter(lenhSql, _
Ket_noi)
' tạo và đổ dữ liệu vào dataset, đặt tên bảng mới
Dim dataset1 As New DataSet
adapter1.Fill(dataset1, "chitiethd")
' đặt thuộc tính cho lưới
DataGridView1.DataSource = dataset1
DataGridView1.DataMember = "chitiethd"
```

Kết quả chạy chương trình:



SOHD	NGAYBAN	maSP	NGUOIBAN	soluong	thanhtien
1	2/1/2011	1	An	2	16056000
2	2/1/2011	2	Bình	3	39240000
3	2/1/2011	3	Vân	2	10580000
4	2/3/2011	4	Bình	3	44700000
5	3/1/2011	5	An	1	5790000
6	3/1/2011	1	An	1	8028000
7	3/2/2011	1	Vân	2	16056000
8	4/1/2011	2	Vân	2	26160000
9	4/1/2011	3	An	2	10580000
10	4/1/2011	3	An	1	5290000
11	4/1/2011	4	Bình	5	74500000
12	4/1/2011	4	Vân	3	44700000

Ví dụ 5.8 b: Thiết kế form trong ví dụ 5.8a bằng cách sử dụng DataAdapter và DataTable:

```
Imports System.Data
Public Class Form2
   Private Sub Form2_Load(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MyBase.Load
      ' tạo dataAdapter lấy tên bo_doc_ghi
      Dim lenhSQL As String = "select * from banhang"
      Dim bo_doc_ghi As New SqlClient.SqlDataAdapter(lenhSQL,
_
      Chuoi_ket_noi)
```

```
        ' khai báo đối tượng dataTable
        Dim dataTable1 As New DataTable

        ' đổ dữ liệu từ dataAdapter vào dataTable
          bo_doc_ghi.Fill(dataTable1)

        ' gắn lưới với dataTable
          DataGridView1.DataSource = dataTable1
    End Sub
End Class
```

5.5.3 Sử dụng DataAdapter lấy cấu trúc từ nguồn dữ liệu

▪ Tạo cấu trúc dữ liệu cho DataSet có sẵn, phương thức trả về một tập hợp các cấu trúc bảng được thêm vào DataSet

```
FillSchema(tên_dataset, kiểu_cấu_trúc)
```

Trong đó kiểu_cấu_trúc qui định việc ánh xạ tên có được chấp nhận hay không. Các giá trị của kiểu cấu trúc được qui định như sau:

Giá trị	**Mô tả**
SchemaType.Mapped	Sử dụng các TableMappings cho các cấu trúc đưa vào DataSet nếu trùng hợp. Tên bảng đặt theo tên tương ứng trong cơ sở dữ liệu.
SchemaType.Source	Không sử dụng các TableMappings. DataAdapter sẽ lấy cấu trúc về cho các bảng trong DataSet và đặt tên các bảng theo cách mặc nhiên Table, Table1

Thông thường SchemaType.Mapped được sử dụng

▪ Tạo cấu trúc dữ liệu cho DataTable có sẵn, phương thức trả về DataTable

```
FillSchema(tên_datatable, kiểu_cấu_trúc)
```

▪ Tạo cấu trúc dữ liệu cho bảng <tên Datatable> trong DataSet; nếu chưa có, bảng sẽ được tạo ra

```
FillSchema(tên_dataset, kiểu_cấu_trúc, chuỗi_tên_bảng)
```

Nếu bảng đã có cấu trúc sẵn, các cột có tên khác với cột có sẵn trên bảng sẽ được thêm vào. Tùy theo cấu trúc trên nguồn dữ liệu, những thuộc tính sau sẽ được đưa vào cấu trúc của cột:

- AllowDBNull: cho phép nhận giá trị Null hay không
- AutoIncrement: tự động tăng
- MaxLength: kích thước tối đa của cột (tính theo Byte)
- ReadOnly: chỉ cho phép đọc

◆ Unique: cột có trị duy nhất

Phương thức FillSchema cũng định luôn khóa chính và các ràng buộc cho bảng theo nguyên tắc sau:

◆ Các cột khóa chính trong lệnh truy vấn của SelectCommand được dùng làm các cột khóa chính cho bảng.

◆ Nếu không có cột khóa chính trong lệnh truy vấn, nhưng có những cột có trị duy nhất và nếu những cột này không cho phép nhận giá trị Null, thì được dùng làm khóa chính. Ngược lại, mỗi một cột tuy có trị duy nhất nhưng chấp nhập giá trị Null, thì một ràng buộc duy nhất được thêm vào tập hợp Constraints của bảng nhưng không có khóa chính.

Chỉ có các ràng buộc khóa chính và khóa duy nhất mới thêm vào tập hợp Constraints, các loại ràng buộc khác không được đưa vào.

Những đặc điểm của phương thức FillSchema cũng giống như của Fill

5.5.4 Tạo bộ lệnh cập nhật cho DataAdapter:

Để tự động phát sinh lệnh cập nhật cho các command còn lại của DataAdapter, chúng ta sử dụng CommandBuilder thuộc không gian tên:

System.Data.Oledb.OledbCommandBuilder

System.Data.SqlClient.SqlCommandBuilder

Cú pháp sử dụng:

```
New <loại>CommandBuilder(tên_DataAdapter)
```

Mỗi DataAdapter chỉ có thể kết hợp với một CommandBuilder. Đối tượng này dùng nội dung lệnh trong SelectCommand của DataAdapter để tạo nội dung lệnh cho các Command còn lại với các đặc điểm sau:

▪ CommandBuilder chỉ phát sinh nội dung lệnh cập nhật cho các DataAdapter có nội dung

▪ SelectCommand truy xuất đến chỉ một bảng nguồn

▪ Nội dung trong SelectCommand phải có ít nhất một khóa chính hoặc một khóa duy nhất (Unique key) để DataAdapter phân biệt các dòng khi cập nhật. Nếu không, CommandBuilder sẽ không phát sinh được nội dung lệnh cho các Command Insert, Update, Delete.

▪ Trường hợp nội dung của SelectCommand là truy vấn từ hơn một bảng hoặc từ một StoredProcedure, các Command cập nhật không tự động phát sinh, nhưng chúng ta phải khai báo các Command cập nhật cách tường minh.

▪ Sau khi lệnh được phát sinh, nếu nội dung lệnh của SelectCommand thay đổi, lệnh của các Command khác không tự động thay đổi theo cho đến khi phương thức RefreshSchema của CommandBuilder được gọi.

- Trường hợp nội dung của SelectCommand là nhiều câu Select SQL, CommandBuilder chỉ tạo được lệnh cập nhật cho lệnh truy vấn đầu tiên.

5.5.5 Dùng DataAdapter cập nhật các thay đổi về nguồn dữ liệu

Sử dụng các cú pháp lệnh sau:

```
Update(<mảng dòng>)
```

Phương thức này cập nhật các dòng <mảng dòng> vào nguồn dữ liệu. Trong đó <mảng dòng> là mảng các đối tượng lớp DataRow, thường mảng này là kết quả trả về của phương thức GetChanges của DataSet

```
Update(<dataset>)
```

Phương thức này cập nhật các thay đổi trên tất cả các bảng của <dataset> vào nguồn dữ liệu với dataset là Đối tượng DataSet mà DataAdapter sẽ cập nhật vào nguồn

```
Update(<datatable>)
```

Phương thức này cập nhật các thay đổi trên DataTable vào nguồn dữ liệu với <datatable> là đối tượng DataTable mà DataAdapter sẽ cập nhật vào nguồn

```
Update(<dataset>, <tên bảng>)
```

Phương thức này cập nhật các thay đổi trên bảng có tên <tên bảng> trong DataSet vào nguồn dữ liệu

Khi phương thức Update được gọi, DataAdapter sẽ kiểm tra tình trạng các dòng là thêm mới, sửa đổi, xóa và gọi thực hiện tự động các Command tương ứng cho mỗi dòng. Nếu các Command chưa được tạo sẽ phát sinh lỗi. Chúng ta có thể tạo và gán cho mỗi loại Command trên DataAdapter hoặc tự động phát sinh thông qua CommandBuilder như đề cập ở mục trên.

Thông thường, mỗi Command trên DataAdapter có một tập hợp tham số kết hợp với nó. Các tham số này được ánh xạ với dòng đang cập nhật thông qua thuộc tính:

- SourceColumn: Cho biết tên cột trên dòng được cập nhật liên kết với tham số
- SourceVersion: Cho biết giá trị của phiên bản nào trên dòng được cập nhật truyền vào vị trí tham số.

Giá trị	**Mô tả**
Current	Tham số dùng giá trị hiện hành của cột. Đây là giá trị mặc định.
Default	Tham số dùng giá trị mặc định đã qui định cho cột.
Original	Tham số dùng giá trị gốc: giá trị từ khi bảng được lấy về từ nguồn dữ liệu hay từ lần gọi cập nhật lần trước.
Proposed	Tham số sử dụng một giá trị đề nghị.

5.6 DataSet

Đối tượng DataSet thuộc namespace System.Data. Đây được coi như một kho chứa các bảng (Table). Người sử dụng có thể thay đổi dữ liệu trong các bảng này và khi muốn cập nhật vào cơ sở dữ liệu thì thi hành phương thức Update của đối tượng DataAdapter.

Các bảng trong DataSet có thể do DataAdapter Fill vào hoặc cũng có thể là các bảng được tạo thành từ đối tượng DataTable.

Các bảng này được quản lý bởi tập hợp Tables của lớp DataSet

5.6.1 Tạo DataSet

a. Cú pháp khai báo

```
Dim tên_dataset as New DataSet()
```

b. Các thuộc tính của DataSet

Tên	Mô tả
DataSetName	Tên của DataSet. (đọc ghi)
HasErrors	Giá trị cho biết có lỗi xảy ra trên một trong các bảng của DataSet: True/False (chỉ đọc)
Relations	Tập hợp các quan hệ (DataRelation) một nhiều của DataSet. (chỉ đọc)
Tables	Tập hợp các bảng (DataTable) của DataSet. (chỉ đọc)

5.6.2. Làm việc với DataSet

a. Thêm một bảng vào DataSet:

Muốn đưa một DataTable (bảng) vào DataSet, chúng ta dùng phương thức Add của tập hợp Tables với các mẫu lệnh dưới đây:

```
1. <Dataset>.Tables.Add()
```

Một bảng mới tự động tạo ra với tên mặc nhiên (Table1, Table2,…) và đưa vào tập hợp Tables của DataSet

```
2. <Dataset>.Tables.Add(<tên bảng>)
```

Một bảng mới tự động tạo ra với tên là <tên bảng> và đưa vào tập hợp Tables

```
3. <Dataset>.Tables.Add(<bảng>)
```

Chú ý: Tên bảng trong DataSet có phân biệt chữ HOA chữ thường. Nghĩa là có thể có 2 bảng tên "table1" và "Table1"

b. Thêm nhiều bảng vào DataSet:

Muốn đưa nhiều bảng vào DataSet, chúng ta dùng phương thức AddRange của tập hợp Tables.

```
<DataSet>.Tables.AddRange(<mảng DataTable>)
```

Trong đó <mảng DataTable>: Là một mảng các DataTable đã tạo ra muốn đưa vào DataSet.

c. Xóa bảng khỏi DataSet:

Để xóa một bảng khỏi DataSet, chúng ta dùng các phương thức sau của tập hợp Tables.

1. Xóa <DataTable> khỏi tập hợp Tables của DataSet.

```
<DataSet>.Tables.Remove(<DataTable>)
```

2. Xóa <DataTable> có tên là <tên bảng> khỏi tập hợp Tables của DataSet.

```
<DataSet>.Tables.Remove(<tên bảng>)
```

3. Xóa <DataTable> có chỉ số là <chỉ số> khỏi tập hợp Tables của DataSet.

```
<DataSet>.Tables.RemoveAt(<chỉ số>)
```

Tuy nhiên, bảng có thể đang hiển thị dữ liệu trên Form nên chúng ta cần kiểm tra xem có thể xóa được không với cú pháp:

```
<DataSet>.Tables.CanRemove(<DataTable>)
```

Phương thức trả về True có thể xóa, ngược lại là False.

Ví dụ:

```
Dim bang As DataTable = dst.Tables(0)
If dst.Tables.CanRemove(bang) then
   dst.Tables.Remove(bang)
End If
```

d. Xóa tất cả các bảng khỏi DataSet

Để xóa tất cả các bảng khỏi DataSet, chúng ta dùng phương thức Clear của tập hợp Tables.

```
<DataSet>.Tables.Clear()
```

e. Kiểm tra bảng có thuộc về DataSet

```
<DataSet>.Tables.Contains(<tên DataTable>)
```

Phương thức trả về True nếu trong Tables có DataTable có tên <tên DataTable>, ngược lại là False.

f. Lấy chỉ số của bảng

```
1.  <DataSet>.Tables.IndexOf(<tên DataTable>)
```

Phương thức trả về chỉ số của DataTable có tên <tên DataTable>.

```
2.  <DataSet>.Tables.IndexOf(<DataTable>)
```

Phương thức trả về chỉ số của <DataTable>.

g. Lấy số bảng chứa trong DataSet

```
<DataSet>.Tables.Count
```

h. Để kiểm tra dữ liệu của DataSet có thay đổi

1. `<DataSet>.HasChanges()`

Phương thức kiểm tra sự thay đổi của tất cả các dòng dữ liệu trên các bảng (thêm mới, xóa bỏ, sửa đổi) và trả về True nếu có, ngược lại False.

2. `<DataSet>.HasChanges(<trạng thái dòng>)`

Phương thức kiểm tra sự thay đổi của tất cả các dòng dữ liệu trên các bảng có trạng thái như <trạng thái dòng> và trả về True nếu có, ngược lại False..

i. Lấy ra dòng dữ liệu đã thay đổi trong DataSet

1. `<DataSet>.GetChanges()`

Phương thức trả về bản sao của DataSet gồm những dòng dữ liệu đã bị thay đổi trên các bảng (do thêm mới, xóa bỏ, sửa đổi).

2. `<DataSet>.GetChanges(<trạng thái dòng>)`

Phương thức trả về bản sao của DataSet gồm những dòng dữ liệu đã bị thay đổi có trạng thái như <trạng thái dòng>.

Ví dụ:

```
Dim ds As DataSet
If dst.HasChanges() then ds = dst.GetChanges()
…
If dst.HasChanges(DataRowState.Modified) then
   ds = dst.GetChanges(DataRowState.Modified)
End If
```

j. Để cập nhật các thay đổi trên DataSet

```
<DataSet>.AcceptChanges()
```

Phương thức cập nhật các thay đổi kể từ lúc lấy dữ liệu về hoặc từ lần gọi AcceptChanges trước.

Trên DataTable, DataRow cũng có phương thức tương ứng. Khi gọi AcceptChanges của DataSet sẽ kéo theo gọi AcceptChanges của DataTable, đến lượt kéo theo gọi AcceptChanges của DataRow.

k. Để hủy bỏ các thay đổi trên DataSet:

```
<DataSet>.RejectChanges()
```

Phương thức này của DataSet phục hồi tất cả các thay đổi kể từ lúc lấy dữ liệu về hoặc từ lần gọi AcceptChanges trước.

Khi gọi RejectChanges của DataSet sẽ kéo theo gọi RejectChanges của DataTable, DataRow.

l. Để trộn dữ liệu bên ngoài vào DataSet

Dữ liệu bên ngoài có thể là các dòng (mảng dòng), DataTable hoặc một DataSet khác.

Nguyên tắc của phương thức này như sau: khi được gọi, cấu trúc của nguồn và đích được so sánh với nhau. Nếu cấu trúc dữ liệu bên ngoài khác với cấu trúc của DataSet do thêm cột, các cột mới sẽ được thêm vào DataSet (tùy cú pháp) cùng với dữ liệu. Trên dữ liệu bên ngoài, những dòng có tình trạng Unchanged, Modified và Deleted phải có cột khóa chính như những dòng đã có trên DataSet. Những dòng có tình trạng Added (thêm mới) phải thỏa ràng buộc về khóa chính của DataSet.

Trong quá trình trộn lẫn dữ liệu, các ràng buộc được bỏ qua: thuộc tính EnforceConstraints có giá trị False. Cuối quá trình, nếu ràng buộc nào không thể chuyển lại True sẽ làm phát sinh một Exception. Vì vậy, chúng ta cần giải quyết các vi phạm ràng buộc trước khi gán EnforceConstraints với giá trị True.

Phương thức này có rất nhiều cách sử dụng.

```
1.  <DataSet>.Merge(<mảng dòng>)
```

Trộn một mảng các dòng vào DataSet.

```
2.  <DataSet>.Merge(<DataTable>)
```

Trộn một DataTable bên ngoài vào DataSet.

```
3.  <DataSet>.Merge(<DataSet>)
```

Trộn một Dataset bên ngoài vào DataSet.

```
4.  <DataSet>.Merge(<DataSet>,<giữ thay đổi>)
```

Trộn một Dataset bên ngoài và cấu trúc của nó vào DataSet.

<giữ thay đổi>: nếu là True, các thay đổi từ bên ngoài được giữ nguyên trên DataSet, là False không thay đổi theo bên ngoài.

```
5.  <DataSet>.Merge(<DataSet>,<giữ thay đổi>, _
<xử lý khi cấu trúc khác nhau>)
```

Trộn một Dataset bên ngoài và cấu trúc của nó vào DataSet.

<xử lý khi cấu trúc khác nhau>: tùy theo giá trị truyền vào.

```
6.  <DataSet>.Merge(<mảng dòng>,<giữ thay đổi>,_
<xử lý khi cấu trúc khác nhau>)
```

Trộn một mảng dòng bên ngoài và cấu trúc của nó vào DataSet.

```
7.  <DataSet>.Merge(<DataTable>,<giữ thay đổi>, _
<xử lý khi cấu trúc khác nhau>)
```

Trộn một DataTable bên ngoài và cấu trúc của nó vào DataSet.

Các giá trị của <xử lý khi cấu trúc khác nhau> được mô tả như sau:

Tên	Mô tả
Add	Nếu khác nhau, thêm các cột mới vào cấu trúc của DataSet.
Error	Nếu khác nhau phát sinh lỗi.
Ignore	Nếu khác nhau, bỏ qua các cột mới.

m. Để hủy bỏ DataSet

```
<DataSet>.Dispose()
```

Khi được gọi, mọi tài nguyên trên vùng nhớ mà DataSet đang sử dụng sẽ được giải phóng.

n. Để tạo một quan hệ giữa hai bảng trong DataSet:

Để thiết lập quan hệ cha-con giữa hai bảng (DataTable) trong một DataSet, phải thỏa yêu cầu sau:

- Field hoặc các Field của bảng cha trong quan hệ phải thỏa yêu cầu tính duy nhất.
- Chỉ có thể thiết lập quan hệ giữa hai bảng trong cùng DataSet

Chúng ta sử dụng phương thức Add của tập hợp Relations trong DataSet với các cú pháp sau.

```
1. <DataSet>.Relations.Add(<đối tượng DataRelation>)
```

<đối tượng DataRelation>: đối tượng này phải được tạo sẵn sẽ đề cập ở phần sau

```
2. <DataSet>.Relations.Add(<DataColumn trên bảng
cha>, <DataColumn trên bảng con>)
```

<DataColumn trên bảng cha>, <DataColumn trên bảng con>: Các đối tượng này thuộc lớp DataColumn sẽ đề cập ở phần sau là các cột tham gia quan hệ trên bảng cha và bảng con tương ứng

```
3. <DataSet>.Relations.Add(<mảng DataColumn trên bảng
cha>, <mảng DataColumn trên bảng con>)
```

<mảng DataColumn trên bảng cha>, <mảng DataColumn trên bảng con>: Các mảng này là mảng các cột tham gia quan hệ trên bảng cha và bảng con tương ứng.

```
4. DataSet>.Relations.Add(<tên quan hệ>, _
<DataColumn trên bảng cha>, _
<DataColumn trên bảng con>)

5. <DataSet>.Relations.Add(<tên quan hệ>, _
<mảng DataColumn trên bảng cha>, _
```

```
<mảng DataColumn trên bảng con>)
```

```
6.  <DataSet>.Relations.Add(<tên quan hệ>, _
<DataColumn trên bảng cha>, _
<DataColumn trên bảng con>, <tạo ràng buộc>)
```

```
7.  <DataSet>.Relations.Add(<tên quan hệ>, _
<mảng DataColumn trên bảng cha>, _
<mảng DataColumn trên bảng con>, <tạo ràng buộc>)
```

Mặc định, khi tạo quan hệ giữa hai bảng, các ràng buộc đồng thời được tạo ra cho trên mỗi bảng như sau:

◆ Ràng buộc duy nhất trên bảng cha dựa vào các cột của bảng cha tham gia vào quan hệ (nếu chưa có)

◆ Ràng buộc khóa ngoại trên bảng con dựa vào các cột trên bảng cha và bảng con trong quan hệ. Tham số <tạo ràng buộc> cho phép chúng ta quy định có tạo ràng buộc hay không: True có tạo, False không tạo.

o. Thêm nhiều quan hệ vào DataSet:

Muốn đưa nhiều quan hệ có sẵn vào DataSet, chúng ta dùng phương thức AddRange của tập hợp Relations.

```
<DataSet>.Relations.AddRange(<mảng quan hệ>)
```

<mảng quan hệ>: là một mảng các quan hệ đã tạo ra muốn đưa vào DataSet.

p. Xóa quan hệ khỏi DataSet

Để xóa một quan hệ khỏi DataSet, chúng ta dùng các phương thức sau của tập hợp Relations.

```
1.    <DataSet>.Relations.Remove(<quan hệ>)
```

Xóa <quan hệ> khỏi tập hợp Relations của Dataset.

```
2.    <DataSet>.Relations.Remove(<tên quan hệ>)
```

Xóa quan hệ có tên là <tên quan hệ> khỏi tập hợp Relations.

```
3.    <DataSet>.Relations.RemoveAt(<chỉ số>)
```

Xóa quan hệ có chỉ số là <chỉ số> khỏi tập hợp Relations.

Tuy nhiên, quan hệ có thể đang được sử dụng nên chúng ta cần kiểm tra xem có thể xóa được không với cú pháp:

```
4.  DataSet>.Relations.CanRemove(<quan hệ>)
```

Phương thức trả về True có thể xóa, ngược lại là False.

Để hủy tất cả các quan hệ trong DataSet, chúng ta dùng các phương thức sau của tập hợp Relations:

```
<DataSet>.Relations.Clear()
```

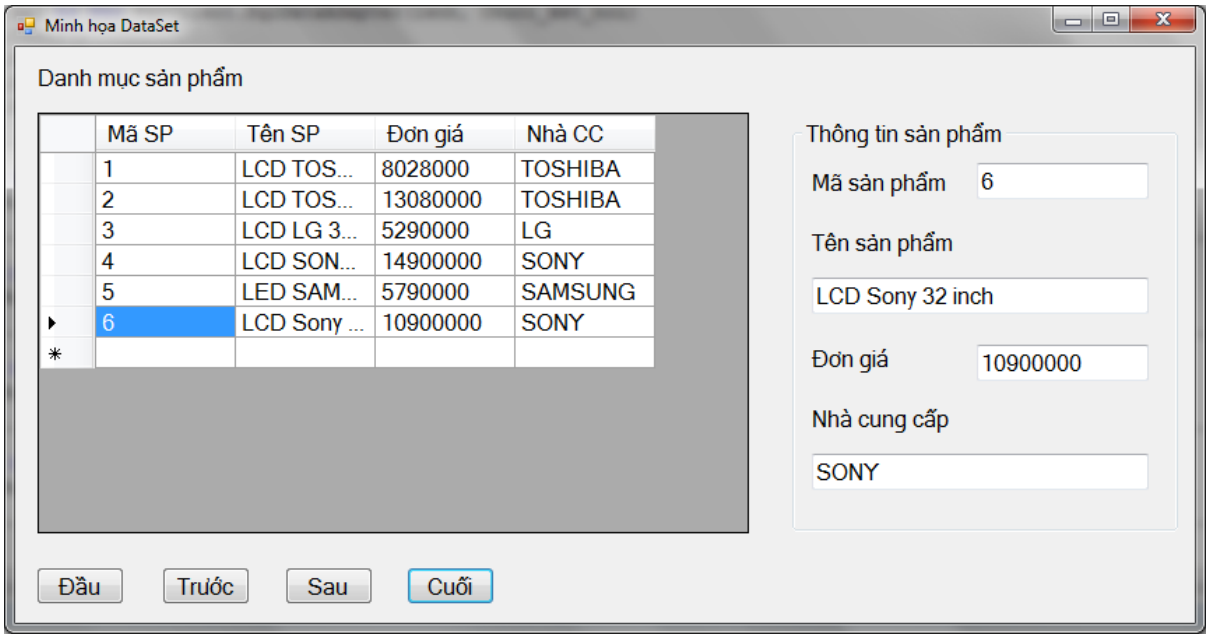
q. Kiểm tra quan hệ có thuộc về DataSet:

```
<DataSet>.Relations.Contains(<tên quan hệ>)
```

Phương thức trả về True nếu trong Relations có quan hệ tên <tên quan hệ>, ngược lại là False.

5.6.3 Ví dụ minh họa làm việc với DataSet

Ví dụ 5.9: Thiết kế form dùng để xem thông tin sản phẩm. Khi chạy chương trình form sẽ thể hiện các thông tin như hình dưới đây. Người sử dụng có thể sử dụng các nút đầu, cuối, trước, sau hoặc bấm trực tiếp trên lưới để xem thông tin sản phẩm.



Hướng dẫn tạo form: Sử dụng công cụ datagridview để thêm lưới dg1 vào form. Các button Đầu, Trước, Sau, Cuối lần lượt có tên là btdau, btt, bta, btc. Các hộp textbox tương ứng với mã sản phẩm là txtmsp; tên sản phẩm là txttensp, đơn giá là txtdg; nhà cung cấp là txtncc.

Trong phần code chúng ta sẽ định nghĩa một đối tượng dataset có tên dst, tầm vực đơn thể class và một đối tượng dataAdapter có tên da1 lấy dữ liệu từ bảng sanpham của cơ sở dữ liệu De1 trên server HTC-VAIO\Huyen. Module khai báo kết nối và chuỗi kết nối được tạo như trong ví dụ 5.2b.

Để di chuyển con trỏ bản ghi ta sử dụng lớp có tên là BindingContext của form. BindingContext là đối tượng quản lý các BindingManagerBase. Các đối tượng kế thừa từ lớp Control đều có thể có BindingContext. Tuy nhiên, chỉ có Form và các điều khiển chứa các điều khiển khác như Groupbox, TabControl, Panel mới có thể

tạo một BindingContext để quản lý các BindingManagerBase hiển thị dữ liệu của các điều khiển chứa trong nó.

Vị trí của con trỏ bản ghi được xác định hoặc thiết lập thông qua đối tượng Position của BindingContext bằng cách sử dụng một trong hai cú pháp sau:

```
Me.BindingContext()(datasource).Position
```

Hoặc

```
Me.BindingContext()(datasource,datamember).Position
```

Phần code của form:

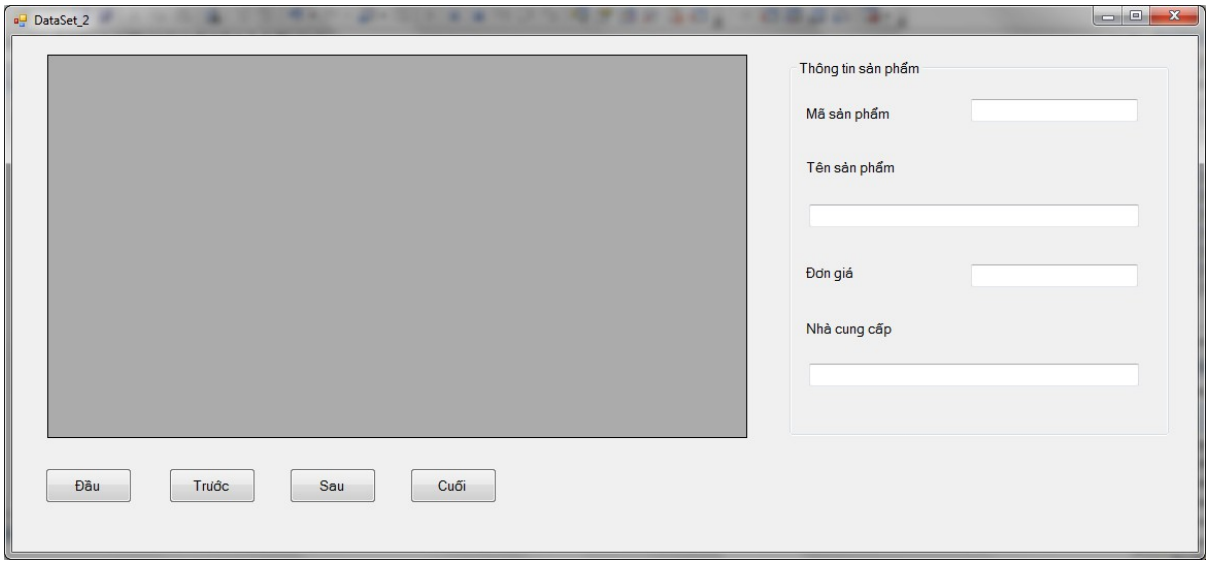
```
Imports System.Data
Public Class FrmDataSet_1
    Dim dst As New DataSet

    Private Sub FrmDataSet_1_Load...
        Dim lenh As String = "select masp as 'Mã SP', _
        tensp as 'Tên SP',dongia as 'Đơn giá', _
        nhacc as 'Nhà CC' from sanpham"
        Dim da1 As New SqlClient.SqlDataAdapter(lenh, Chuoi_ket_noi)
        dst.Clear()   'xóa nội dung dataset
        da1.Fill(dst, "sanpham")   'đổ DL vào bảng sản phẩm của dst
        'khai báo thuộc tính cho lưới dg1
        dg1.DataSource = dst
        dg1.DataMember = "sanpham"
    End Sub

    Private Sub btdau_Click...
        'chuyển con trỏ về bản ghi đầu tiên
        Me.BindingContext()(dst, "sanpham").Position = 0
    End Sub

    Private Sub btt_Click ...
        'chuyển con trỏ về bản ghi trước
        Me.BindingContext()(dst, "sanpham").Position -= 1
    End Sub

    Private Sub bts_Click ...
        'chuyển con trỏ về bản ghi tiếp theo
        Me.BindingContext()(dst, "sanpham").Position += 1
    End Sub

    Private Sub btc_Click...
        'chuyển con trỏ về bản ghi cuối cùng
```

```
        Me.BindingContext()(dst, "sanpham").Position = _
        Me.BindingContext()(dst, "sanpham").Count - 1
    End Sub

    'thủ tục in thông tin của sản phẩm khi thay đổi bản ghi hiện thời
   Private Sub dg1_SelectionChanged...
        Dim dong%
        'lấy giá trị dòng hiện thời của lưới dg1
        dong = dg1.CurrentCellAddress.Y

        'lấy giá trị các ô trên dòng hiện thời của lưới dg1
        txtmsp.Text = dg1.Item(0, dong).Value
        txttensp.Text = dg1.Item(1, dong).Value
        txtdg.Text = dg1.Item(2, dong).Value
        txtncc.Text = dg1.Item(3, dong).Value

    End Sub
End Class
```

Visual Basic.NET cung cấp một tính năng có thể tự động chèn các DataAdapter và DataSet vào project.

Chẳng hạn trong ví dụ 5.9 trên chúng ta có thể thêm các điều khiển DataAdapter và DataSet như sau:

- Bước 1: Thiết kế form như mẫu dưới đây: (thuộc tính và tên của các điều khiển giống như trong ví dụ 5.9)



- Bước 2: Kết nối đến cơ sở dữ liệu De1

Chọn View/Server 

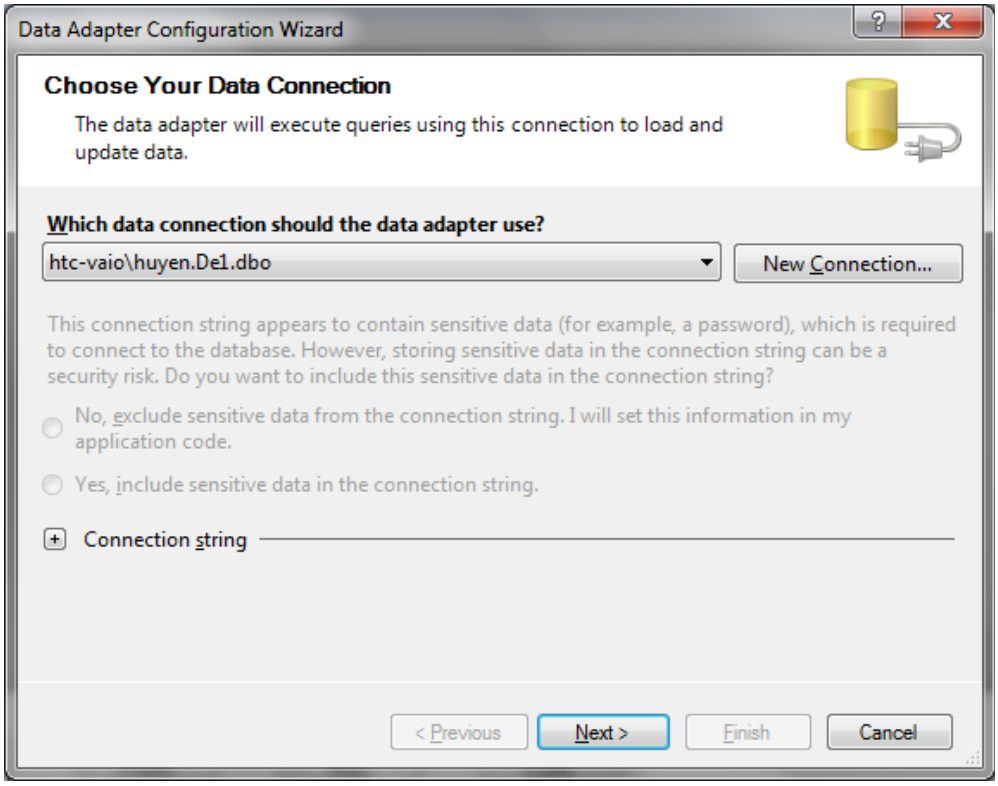
Có thể chọn Attach để kết nối với cơ sở dữ liệu ở một vị trí bất kỳ.

+ Nhấn OK. Có thể kiểm tra kết nối bằng cách nhấn nút Test Connection. Khi xuất hiện thông báo Test Connection Succeed có nghĩa là bạn đã kết nối thành công.

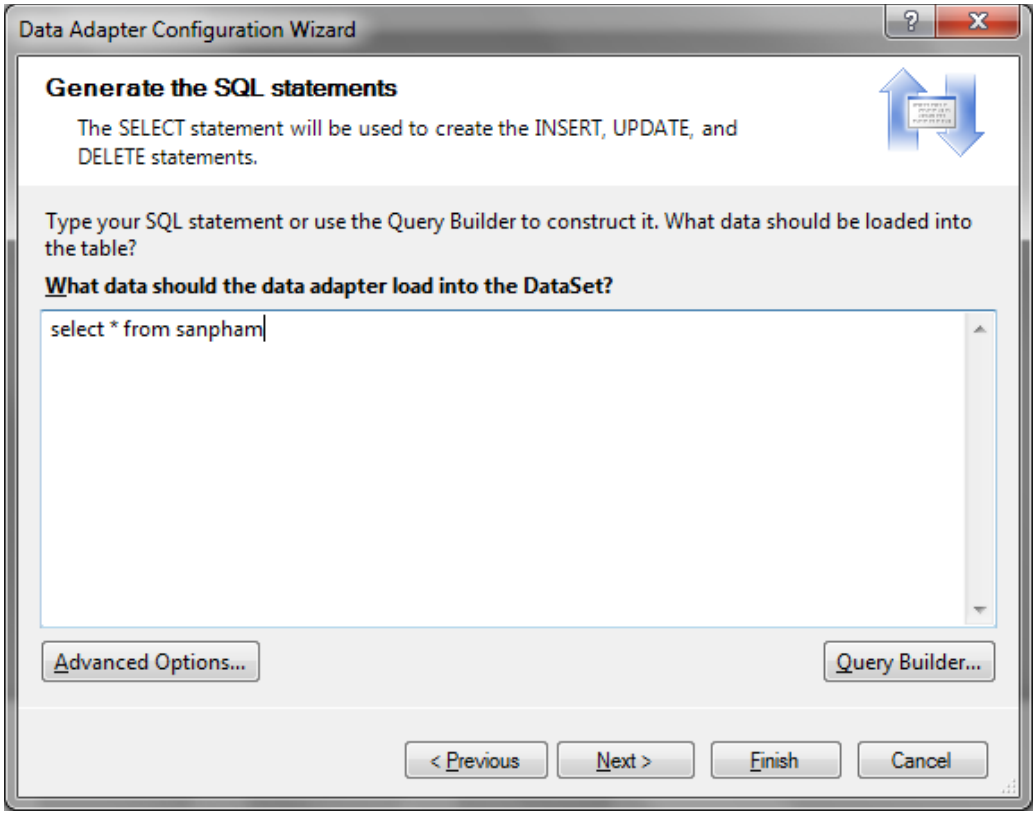
- Bước 3: Thêm điều khiển DataAdapter

+ Chọn biểu tượng SQLDataAdater (hoặc OLEDBDataAdater nếu là cơ sở dữ liệu Access) và kéo vào form. Hộp thoại Data Adapter Configuration sẽ xuất hiện để bạn khai báo cấu hình của Data Adapter. Nếu bạn đã hoàn thành bước hai thì đường dẫn đến cơ sở dữ liệu cần kết nối sẽ xuất hiện trong mục Which data connection. Ta cũng có thể nhấn nút New Connection để tạo một kết nối mới.

+ Nhấn Next để chuyển sang hộp thoại khai báo giá trị Command Type. Có 3 kiểu đó là sử dụng câu lệnh SQL hoặc sử dụng một thủ tục nội tại của SQL hoặc tạo một thủ tục mới. Trong trường hợp này ta khai báo giá trị Use SQL Statement.



+ Nhấn Next sang màn hình khai báo câu lệnh SQL cho Select Command (bắt buộc). Các câu lệnh Update, Delete, Insert có thể tự sinh ra. Có thể dùng nút Query Builder để xây dựng query.



Nhập câu lệnh Select vào cửa sổ lệnh (trong trường hợp này để tham chiếu đến bảng sanpham ta nhập lệnh `select * from sanpham`).

- Bước 4: Tạo dataset

Gọi lệnh Data/Generate Dataset hoặc nhấn phải chuột vào Data Adapter chọn Generate Dataset. Đặt tên cho dataset hoặc chọn một dataset đã có sẵn.

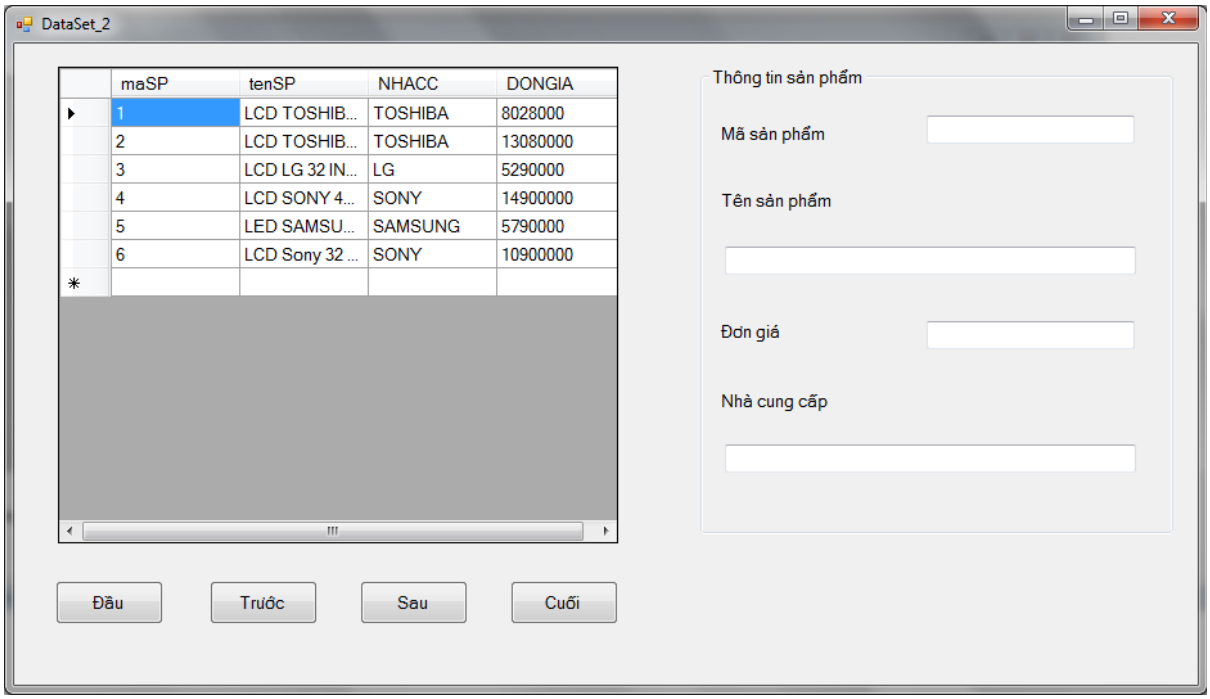
- Bước 5: Viết code

Sau khi hoàn tất bước 4, bạn đã tạo được một đối tượng DataAdapter1 và một đối tượng DataSet1. Trên cửa sổ Solution Explorer sẽ xuất hiện đối tượng DataSet1.xsd.

Để đổ dữ liệu vào cho dataset ta bổ sung đoạn mã sau vào sự kiện load của form:

```
SqlDataAdapter1.Fill(DataSet11, "sanpham")
' đặt thuộc tính của lưới dg1
dg1.DataSource = DataSet11
dg1.DataMember = "sanpham"
```

Khi chạy chương trình, ta sẽ thấy dữ liệu được hiển thị như hình dưới đây:



Để di chuyển con trỏ bản ghi và hiển thị dữ liệu bản ghi hiện thời, ta viết code cho sự kiện Click của nút đầu, cuối, trước, sau và sự kiện SelectionChanged của lưới dg1.

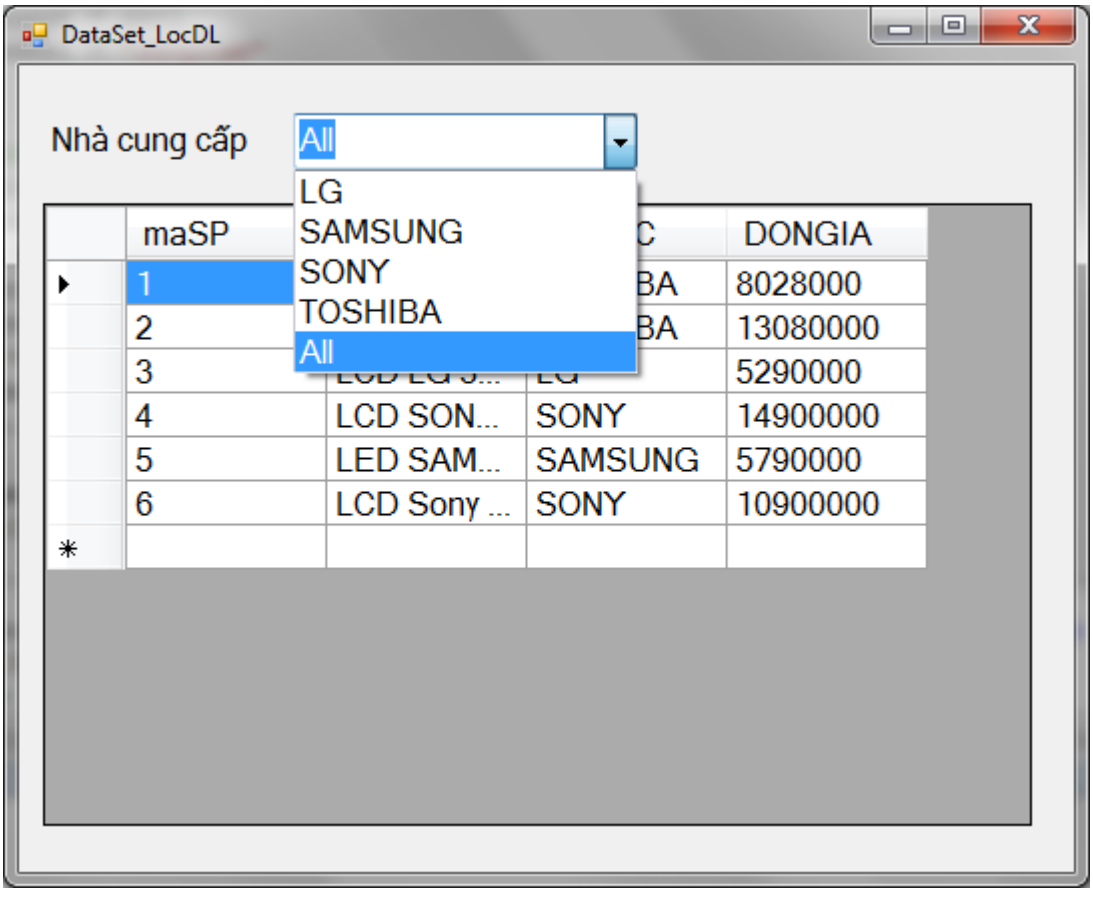
```
Private Sub btdau_Click...
    'chuyển con trỏ về bản ghi đầu tiên
    Me.BindingContext()(dataset11, "sanpham").Position = 0
End Sub

Private Sub btt_Click ...
    'chuyển con trỏ về bản ghi trước
    Me.BindingContext()(dataset11, "sanpham").Position -= 1
End Sub

Private Sub bts_Click ...
    'chuyển con trỏ về bản ghi tiếp theo
    Me.BindingContext()(dataset11, "sanpham").Position += 1
End Sub

Private Sub btc_Click...
    'chuyển con trỏ về bản ghi cuối cùng
    Me.BindingContext()(dataset11, "sanpham").Position = _
    Me.BindingContext()(dataset11, "sanpham").Count - 1
End Sub

 'thủ tục in thông tin của sản phẩm khi thay đổi bản ghi hiện thời
Private Sub dg1_SelectionChanged...
    Dim dong%
```

```
        'lấy giá trị dòng hiện thời của lưới dg1
        dong = dg1.CurrentCellAddress.Y

        'lấy giá trị các ô trên dòng hiện thời của lưới dg1
        txtmsp.Text = dg1.Item(0, dong).Value
        txttensp.Text = dg1.Item(1, dong).Value
        txtdg.Text = dg1.Item(2, dong).Value
        txtncc.Text = dg1.Item(3, dong).Value

    End Sub
```

Ví dụ 5.10 - Lọc dữ liệu. Thiết kế form cho phép xem danh sách sản phẩm của từng nhà cung cấp theo mẫu dưới đây.

- Thiết kế: Form gồm hai đối tượng: một combo box có tên cboncc hiển thị các nhà cung cấp trong bảng sản phẩm và một lưới có tên dg1. Khi chọn một nhà cung cấp thì các sản phẩm của nhà cung cấp đó sẽ được hiển thị trên lưới. Trong combo box chứa giá trị All, khi chọn giá trị này thì thông tin của tất cả các sản phẩm sẽ được hiển thị.

Để nạp tên các nhà cung cấp từ bảng sanpham vào cboncc; ta khai báo một đối tượng datareader có tên readerncc nhận dữ liệu từ phương thức ExecuteReader của Command.



- Phần khai báo chung:

```
Dim lenh As String = "select * from sanpham"
Dim da1 As New SqlClient.SqlDataAdapter(lenh, Chuoi_ket_noi)
Dim dst1 As New DataSet
Dim cmd As New SqlClient.SqlCommand
```

- Sự kiện Load của form:

```
dst1.Clear()
da1.Fill(dst1, "sanpham")
dg1.DataSource = dst1
dg1.DataMember = "sanpham"

'nạp tên nhà cung cấp vào cboncc
Tao_ket_noi()
Ket_noi.Open()
cmd.CommandText = "select distinct nhacc from sanpham"
cmd.Connection = Ket_noi
Dim readerncc As SqlClient.SqlDataReader
readerncc =
cmd.ExecuteReader(CommandBehavior.CloseConnection)
While readerncc.Read
   cboncc.Items.Add(readerncc.Item(0))
End While
cboncc.Items.Add("All")
readerncc.Close()
```

- Thay đổi thông tin trên lưới khi chọn một giá trị của cboncc

```
Private Sub cboncc_SelectedIndexChanged ...
If cboncc.Text = "All" Then
   lenh = "select * from sanpham"
Else
   lenh = "select * from sanpham where nhacc='" & _
   cboncc.Text & "'"
End If

da1.SelectCommand.CommandText = lenh
dst1.Clear()
da1.Fill(dst1, "sanpham")

End Sub
```

5.7 DataTable

Dữ liệu các bảng trong nguồn dữ liệu được lấy về và đưa vào các DataTable. DataTable thuộc không gian tên System.Data.DataTable.

Có các cách khai báo như sau:

```
New DataTable()
New DataTable(<tên bảng>)
```

5.7.1 Các thuộc tính của DataTable

Tên	Mô tả
CaseSensitive	Qui định việc so sánh chuỗi trong bảng có phân biệt chữ Hoa
ChildRelations	Trả về tập hợp những quan hệ trong đó DataTable đóng vai
Columns	Trả về tập hợp các cột trong DataTable (thuộc lớp
Constraints	Trả về tập hợp các ràng buộc trong DataTable.
DataSet	Trả về DataSet chứa DataTable.
DefaultView	Trả về DataView phát sinh từ DataTable.
HasErrors	Cho biết có dòng nào trên DataTable bị lỗi : True có dòng
ParentRelations	Trả về tập hợp những quan hệ trong đó DataTable đóng
PrimaryKey	Mảng các cột có chức năng làm khóa chính của DataTable.
Rows	Trả về tập hợp các dòng dữ liệu của DataTable.
TableName	Tên của DataTable.

5.7.2 Một số phương thức của DataTable

a. Phát sinh một dòng mới theo cấu trúc của DataTable

```
<DataTable>.NewRow()
```

Phương thức này trả về một DataRow mới có cấu trúc như của DataTable với các giá trị mặc định nhưng chưa đưa vào tập hợp Rows (nghĩa là chưa thuộc về DataTable, có trạng thái là Detached). Và một khi được đưa vào tập hợp sẽ có trạng thái là Added.

b. Hủy tất cả các dòng dữ liệu trên DataTable

```
<DataTable>.Clear()
```

c. Sao chép cấu trúc, ràng buộc của DataTable thành DataTable khác

```
<DataTable>.Clone()
```

Phương thức trả về một DataTable đã có sẵn cấu trúc và ràng buộc của DataTable nhưng không có dữ liệu

d. Sao chép cấu trúc, ràng buộc và dữ liệu của DataTable thành DataTable khác

```
<DataTable>.Copy()
```

Phương thức trả về một DataTable đã có sẵn cấu trúc, ràng buộc và dữ liệu của DataTable. DataTable trả về có cùng cấp với DataTable - nếu DataTable là đối tượng của một lớp kế thừa, DataTable trả về cũng thuộc lớp đó.

e. Để lấy ra một bản sao những thay đổi trên DataTable

```
<DataTable>.GetChanges()
```

Phương thức trả về một DataTable gồm những dòng dữ liệu đã thay đổi kể từ lần lấy dữ liệu từ nguồn về hoặc từ lần cập nhật trước vào DataTable bằng Phương thức AcceptChanges.

```
<DataTable>.GetChanges(<trạng thái DataRow>)
```

Như trên nhưng chỉ trả về những dòng có tình trạng đúng với <trạng thái DataRow>.

f. Lấy ra một mảng các dòng bị lỗi trên DataTable

```
<DataTable>.GetErrors()
```

Phương thức trả về một mảng các DataRow bị lỗi. Phương thức này được gọi sau khi gọi GetChanges nhằm phát hiện các dòng bị lỗi để xử lý.

g. Cập nhật các thay đổi vào DataTable

```
<DataTable>.AcceptChanges()
```

Phương thức cập nhật các thay đổi kể từ lần cập nhật trước hoặc khi DataTable được mở vào chính nó. Sau khi thực hiện tất cả các DataRow đều có trạng thái Unchanged. Những DataRow có trạng thái Deleted bị loại bỏ khỏi DataTable.

h. Hủy bỏ các thay đổi của DataTable

```
<DataTable>.RejeptChanges()
```

Phương thức phục hồi lại các giá trị kể từ lần cập nhật trước hoặc khi DataTable được mở vào chính DataTable. Sau khi thực hiện, tất cả các dòng mới thêm vào đều bị loại bỏ. Những DataRow có trạng thái Deleted, Modified được phục hồi lại tình trạng gốc.

i. Tính toán trên các DataRow của DataTable

```
<DataTable>.Compute(<biểu thức tính toán>,<biểu thức lọc>)
```

Phương thức thực hiện tính toán theo <biểu thức tính toán> trên những dòng thỏa điều kiện của <biểu thức lọc> và trả về giá trị tính toán được kiểu Object.

Chú ý rằng Trong <biểu thức tính toán>, phải có hàm gộp như Count, Sum,...

<biểu thức lọc> : phải có dạng <tên cột> <toán tử> <giá trị>

j. Chọn các DataRow của DataTable

```
<DataTable>.Select()
```

Phương thức trả về mảng các DataRow trên DataTable theo thứ tự của khóa chính nếu có.

`Select(<biểu thức lọc>)`

Phương thức trả về mảng các DataRow trên DataTable thỏa điều kiện của <biểu thức lọc> theo thứ tự của khóa chính nếu có.

`Select(<biểu thức lọc>, <biểu thức sắp xếp>)`

Phương thức trả về mảng các DataRow trên DataTable thỏa điều kiện của <biểu thức lọc> theo thứ tự của <biểu thức sắp xếp>.

`Select(<biểu thức lọc>, <biểu thức sắp xếp>,<trạng thái dòng>)`

Phương thức trả về mảng các DataRow trên DataTable thỏa điều kiện của <biểu thức lọc> theo thứ tự của <biểu thức sắp xếp> và có RowState ứng với tham số <trạng thái dòng>.

5.7.3 Một số sự kiện của DataTable

Tên	**Mô tả**
ColumnChanged	Sự kiện xảy ra sau khi giá trị trên cột của một dòng đã thay đổi. Tham số EventArgs chứa thông tin: ▪ Column: DataColumn thay đổi ▪ ProposedValue: giá trị thay đổi của DataColumn ▪ Row: DataRow có cột thay đổi
ColumnChanging	Sự kiện xảy ra khi giá trị trên cột của một DataRow đang thay đổi. Tham số EventArgs chứa thông tin như ColumnChanged
RowChanged	Sự kiện xảy ra sau khi một DataRow đã thay đổi thành công (không phát sinh Exception) Tham số EventArgs chứa thông tin: ▪ Action: hành động đã xảy ra trên dòng ▪ Row: dòng có hành động xảy ra
RowChanging	Sự kiện xảy ra khi một dòng đang thay đổi. Tham số EventArgs chứa thông tin như RowChanged
RowDeleted	Sự kiện xảy ra sau khi một dòng trên DataTable đã bị đánh dấu hủy. Tham số EventArgs chứa thông tin như RowChanged
RowDeleting	Sự kiện xảy ra trước khi một dòng trên DataTable bị đánh

	dấu hủy. Tham số EventArgs chứa thông tin như RowChanged

5.7.4. Cấu trúc của bảng - DataColumn

Cấu trúc của bảng là tập hợp các DataColumn với các thuộc tính của chúng. Chúng ta có thể tạo cấu trúc của bảng từ cấu trúc bảng trong nguồn dữ liệu hoặc tạo lập từ các DataColumn.

DataColumn thuộc không gian tên System.Data.DataColumn

a. Các thuộc tính của DataColumn

Tên	**Mô tả**
AllowDBNull	Thuộc tính cho biết cột có chấp nhận giá trị Null không (đọc ghi).
AutoIncrement	Thuộc tính cho biết giá trị cột có tự động tăng khi thêm dòng mới không (đọc ghi)..
AutoIncrementSeed	Giá trị bắt đầu cho cột khi thêm dòng đầu tiên, nếu AutoIncrement là True.
AutoIncrementStep	Bước tăng cho dòng thêm mới kế tiếp, nếu AutoIncrement là True.
Caption	Tiêu đề của cột.
ColumnName	Tên cột.
DataType	Kiểu dữ liệu của cột.
DefaultValue	Giá trị mặc định cho cột khi thêm một dòng mới.
Expression	Biểu thức dùng để tính giá trị cho cột
MaxLength	Độ rộng tối đa cho cột kiểu chuỗi.
Ordinal	Trả về số thứ tự của cột trong tập hợp DataColumn
ReadOnly	Giá trị cho biết cột có được phép sửa đổi sau khi dòng mới được thêm vào.
Table	Trả về DataTable chứa cột.
Unique	Giá trị cho biết giá trị cột của mỗi dòng có phải là duy nhất.

b. Tạo mới DataColumn

Dùng một trong các cú pháp sau:

```
1. New DataColumn()
2. New DataColumn(<tên cột>)
3. New DataColumn(<tên cột>,<kiểu dữ liệu>)
```

4. New DataColumn(<tên cột>,<kiểu dữ liệu>,<biểu thức>)

Trong đó:

- <tên cột>: Tên muốn đặt cho cột

- <kiểu dữ liệu>: Kiểu dữ liệu của cột. Kiểu dữ liệu của DataColumn được khai báo thông qua cú pháp System.Type.GetType("<kiểu dữ liệu>") và chuỗi kiểu dữ liệu phải đúng như không gian tên, tên lớp có phân biệt chữ Hoa chữ thường

- <biểu thức>: Biểu thức tính giá trị cho cột

c. Sử dụng thuộc tính Expression

Để tạo thêm cột tính toán dựa trên các cột đã có, chúng ta có thể tạo thêm cột mới và sử dụng thuộc tính Expression để tính toán số liệu cho cột mới tạo này.

Thuộc tính Expression cho phép

- Diễn dịch, tính toán dữ liệu từ các cột có sẵn trên DataTable
- Hiển thị dữ liệu của các bảng khác trong cùng Dataset thông qua quan hệ

5.7.5 DataRow

Dữ liệu lưu trữ trong DataTable qua các DataRow. DataRow thuộc không gian tên System.Data.DataRow.

a. Các thuộc tính của DataRow

Tên	**Mô tả**
HasErrors	Thuộc tính cho biết dòng có đang bị lỗi hay không.
Item	Nội dung dữ liệu lưu giữ của cột trên dòng có tên, cột datacolumn hay số thứ tự cột truyền vào: <datarow>.Item(<tên>) <datarow>.Item(<cột>) <datarow>.Item(<chỉ số>) Nội dung dữ liệu lưu giữ của cột trên dòng có tên, datacolumn hay số thứ tự cột truyền vào, giá trị trả về theo phiên bản truyền vào: <datarow>.Item(<tên>,<phiên bản>) <datarow>.Item(<cột>,<phiên bản>) <datarow>.Item(<chỉ số>,<phiên bản>) Phiên bản có các trị sau : + Current: Dòng chứa trị hiện hành. + Default: Dòng chứa phiên bản mặc định. Đối với dòng có DataRowState là Added, Modified hoặc

Tên	Mô tả
	Current, phiên bản mặc định là Current. Đối với dòng có DataRowState là Deleted, phiên bản mặc định là phiên bản Original. Đối với dòng có DataRowState là Detached, phiên bản mặc định là phiên bản Proposed. + Original: Dòng chứa trị gốc. + Proposed: Dòng chứa trị đã chỉnh sửa
ItemArray	Tất cả nội dung của DataRow dưới dạng một mảng.
RowError	Mô tả nội dung lỗi của dòng, nếu đang bị lỗi.
RowState	Trả về tình trạng của DataRow và có các trị sau : - Added: Dòng đã được thêm vào tập hợp Rows nhưng chưa được cập nhật. - Deleted: Dòng đã được đánh dấu hủy bằng phương thức Delete. - Detached: Dòng không thuộc về tập hợp Rows gồm những dòng sau: + Dòng đã được tạo mới nhưng chưa đưa vào tập hợp Rows + Dòng đã bị loại bỏ khỏi tập hợp bằng phương thức Remove, RemoveAt của bảng + Dòng đã bị đánh dấu hủy bằng Delete và sau đó đã gọi AcceptChanges của Datarow. + Modified: Dòng đã được sửa đổi nhưng chưa được cập nhật. + Unchanged: Dòng không thay đổi kể từ lúc đọc từ nguồn dữ liệu hoặc từ lần cập nhật trước.
Table	Trả về tên bảng chứa DataRow.

b. Các phương thức của DataRow

Tên	Mô tả
BeginEdit	Bắt đầu chỉnh sửa dòng. Trong chế độ này, mọi sự kiện tạm thời bị ngưng lại, các kiểm tra tạm thời bỏ qua. Khi người dùng bắt đầu thay đổi nội dung trên các điều khiển liên kết, phương thức này được ngầm gọi. Ở chế độ này, và khi EndEdit chưa được gọi, dòng sẽ mang trị các phiên bản Original (giá trị gốc) và Proposed (giá trị mới đề nghị). Có thể truy xuất các trị này thông qua thuộc tính Item(<cột>,<phiên bản>)

Tên	Mô tả
CancelEdit	Hủy bỏ việc chỉnh sửa trên dòng.
Delete	Đánh dấu hủy dòng. Khi phương thức được gọi, dòng có RowState là Deleted
EndEdit	Chấm dứt việc chỉnh sửa.
GetChildRows	Trả về các dòng có quan hệ nhánh con với dòng đang tham chiếu
GetParentRow	Trả về dòng có quan hệ nhánh cha với đối tượng dòng đang tham chiếu
IsNull	Cho biết trị của cột truyền vào có mang trị Null.
RejectChanges	Hủy bỏ các thay đổi từ lần cập nhật trước.

5.7.6 Tập hợp Rows

Rows là tập hợp các dòng dữ liệu của DataTable. Mọi tham chiếu đến DataRow đều thông qua tập hợp này. Sau đây là một số chức năng của tập hợp Rows:

a. Số DataRow trong tập hợp

```
<DataTable>.Rows.Count
```

b. Tham chiếu DataRow trong tập hợp

```
<DataTable>.Rows.Item(<chỉ số DataRow muốn tham chiếu>)
<DataTable>.Rows(<chỉ số DataRow muốn tham chiếu>)
```

c. Truy xuất trị của ô trên một DataRow

```
<DataTable>.Rows.Item(<chỉ số dòng>)(<chỉ số cột>)
<DataTable>.Rows(<chỉ số dòng>)(<chỉ số cột>)
```

Các cách sau áp dụng cho cả hai cách thông qua Item và không có Item

```
<DataTable>.Rows(<chỉ số dòng>)(<tên cột>)
<DataTable>.Rows(<chỉ số dòng>)(<cột>)
<DataTable>.Rows(<chỉ số dòng>)(<chỉ số cột>,<phiên bản>)
<DataTable>.Rows(<chỉ số dòng>)(<tên cột>,<phiên bản>)
<DataTable>.Rows(<chỉ số dòng>)(<cột>,<phiên bản>)
```

d. Thêm DataRow vào DataTable

```
<DataTable>.Rows.Add(<DataRow>)
```

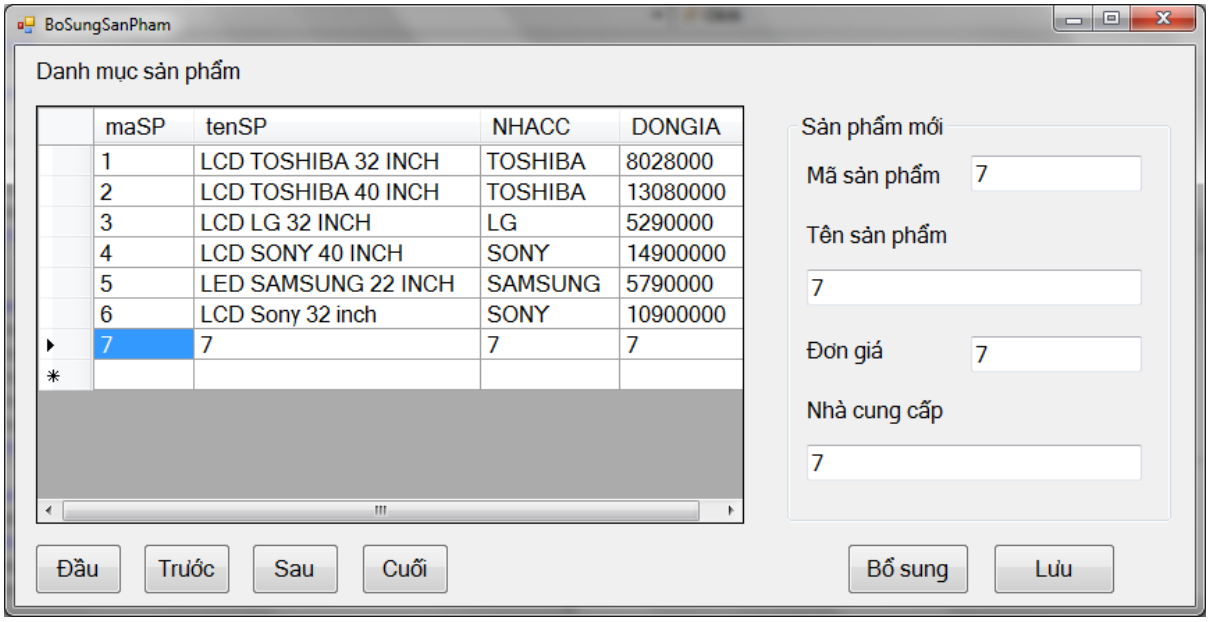
Thêm một dòng có sẵn vào DataTable.

```
<DataTable>.Rows.Add(<mảng trị>)
```

Tạo một dòng mới với các trị trong mảng trị ứng với thứ tự các DataColumn và đưa vào tập hợp Rows của DataTable.

Ví dụ 5.11: Thêm bản ghi mới vào bảng sanpham

Form BosungSanPham được thiết kế như trong ví dụ 5.9; ngoài ra bổ sung thêm hai button là nút Bổ sung và nút Lưu.



Phần khai báo chung của form:

```
Imports System.Data
Public Class BoSungSanPham
    Dim dt As New DataTable
    Dim dst As New DataSet
    Dim lenh As String = "select * from sanpham"
    Dim da1 As New SqlClient.SqlDataAdapter(lenh, Chuoi_ket_noi)
    'tự động sinh các lệnh Insert, Update, Delete cho DataAdapter
    Dim bo_lenh_da1 As New SqlClient.SqlCommandBuilder(da1)
```

Thủ tục Load của Form:

```
    Private Sub BoSungSanPham_Load......
        dst.Clear()
        da1.Fill(dst, "sanpham")
        dg1.DataSource = dst
        dg1.DataMember = "sanpham"
    End Sub
```

Các biến cố Click của button Đầu, Trước, Sau, Cuối và biến cố SelectionChanged của lưới được giữ nguyên như trong ví dụ 5.9

Button Bổ sung dùng để xóa nội dung trong các hộp Textbox và đợi nhập dữ liệu mới:

```
    txtmsp.Clear()
```

```
    txttensp.Clear()
    txtncc.Clear()
    txtdg.Clear()
    txtmsp.Focus()
```

Button Lưu dùng để tạo một dòng mới bằng phương thức NewRow, gán dữ liệu cho dòng mới và bổ sung vào bảng bằng phương thức Add sau đó cập nhật DataAdapter và DataSet.

```
    'định nghĩa bảng dt
    dt = dst.Tables("sanpham")
    'thêm một dòng theo cấu trúc của dt
    Dim dong As DataRow = dt.NewRow
    'gán giá trị cho các cột
    dong.Item("masp") = txtmsp.Text
    dong.Item("nhacc") = txtncc.Text
    dong.Item("tensp") = txttensp.Text
    dong.Item("dongia") = txtdg.Text
    'bổ sung dòng vào bảng
    dt.Rows.Add(dong)
    'cập nhật DataAdapter
    da1.Update(dst, "sanpham")
    'cập nhật dataset
    dst.AcceptChanges()
```

e. Để chèn dòng vào DataTable tại một vị trí

`<DataTable>.Rows.InsertAt(<DataRow>,<vị trí>)`

Chèn một DataRow có sẵn vào DataTable tại <vị trí>.

Nếu <vị trí> lớn hơn số dòng của DataTable, dòng mới được thêm vào vị trí cuối DataTable.

f. Hủy DataRow trên DataTable

`<DataTable>.Rows.Remove(<DataRow>)`

Hủy <DataRow> khỏi tập hợp Rows của DataTable.

`<DataTable>.Rows.RemoveAt(<chỉ số>)`

Hủy DataRow tại vị trí <chỉ số> khỏi DataTable

Phương thức trên là tương tự gọi phương thức Delete và AcceptChanges của Datarow

`<DataTable>.Rows.Clear()`

Hủy toàn bộ các dòng dữ liệu của DataTable

Ví dụ 5.12: Xóa bản ghi hiện thời khỏi bảng sanpham:

Form được thiết kế như hình dưới đây. Nút Xóa dùng để xóa bản ghi hiện thời.

Biến cố Click của nút Xóa có thể dùng hai cách: cách một với phương thức RemoveAt của Rows và cách hai với phương thức Delete của DataRow.

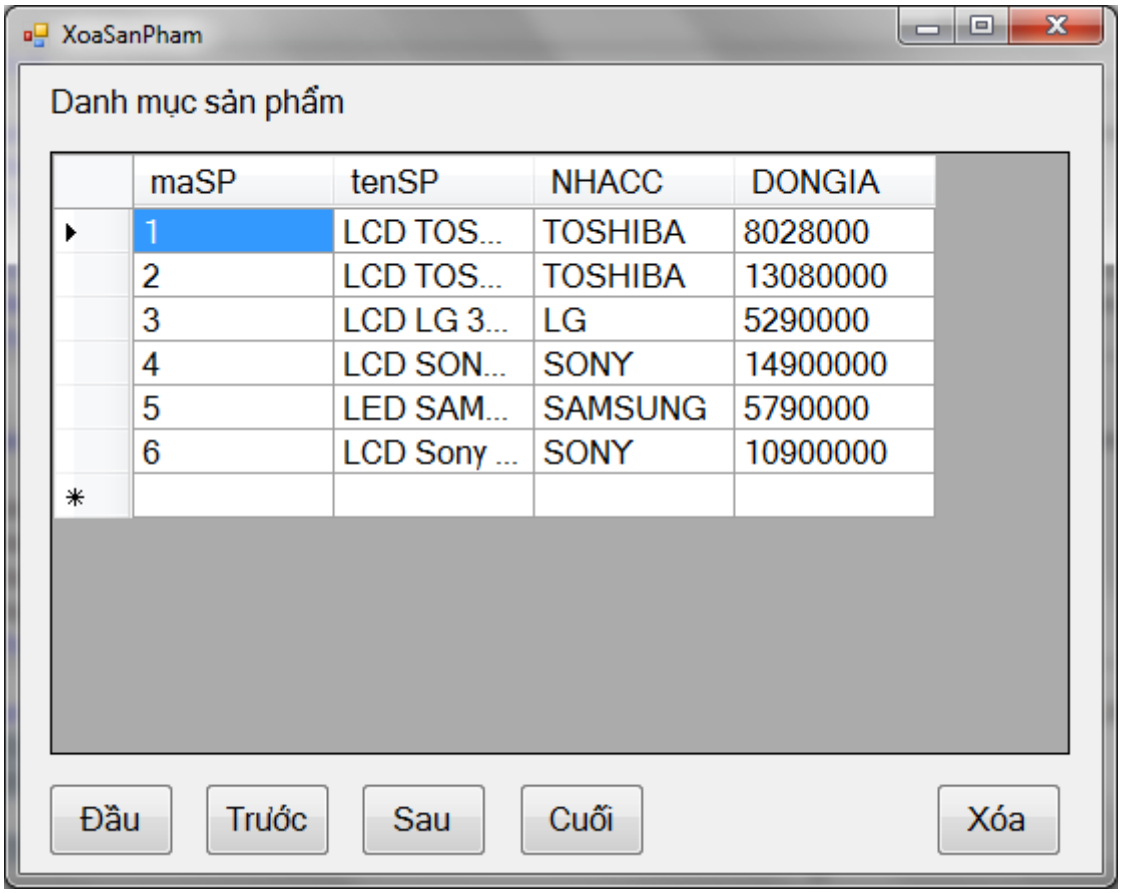
Cách 1: Sử dụng RemoveAt:

```
Dim chiso As Int32 = Me.BindingContext(dst, "sanpham").Position
dt = dst.Tables("sanpham")
Dim traloi As String
traloi = MsgBox("Xóa bản ghi hiện thời ?", MsgBoxStyle.OkCancel)
If traloi = 1 Then
   dt.Rows.RemoveAt(chiso)
End If
da1.Update(dst, "sanpham")
dst.AcceptChanges()
```

Cách 2: Sử dụng Delete để xóa bản ghi hiện thời:

```
Dim chiso As Int32 = Me.BindingContext(dst, "sanpham").Position
dt = dst.Tables("sanpham")

Dim row As DataRow = dt.Rows(chiso)
Dim traloi As String
traloi = MsgBox("Xóa bản ghi hiện thời ?", MsgBoxStyle.OkCancel)
If traloi = 1 Then
  row.Delete()
End If
da1.Update(dst, "sanpham")
dst.AcceptChanges()
```

Khi test thử chương trình, bạn chỉ nên xóa các bản ghi mới thêm vào để tránh việc xóa hết các bản ghi trong cơ sở dữ liệu và vi phạm quan hệ ràng buộc giữa bảng sanpham và bảng banhang. Hoặc có thể tắt tạm thời hai lệnh cập nhật để không xóa trên cơ sở dữ liệu.



g. Để kiểm tra trên khóa chính có chứa trị truyền vào

`<DataTable>.Rows.Contains(<trị>)`

Phương thức kiểm tra trên cột khóa chính xem có dòng nào chứa <trị> và trả về True nếu tìm thấy, ngược lại trả về False

`<DataTable>.Rows.Contains(<mảng trị>)`

Phương thức kiểm tra trên các cột khóa chính có dòng nào chứa bộ giá trị <mảng trị> và trả về True nếu tìm thấy, ngược lại trả về False. Thứ tự trị truyền vào phải theo thứ tự các cột khóa trên DataTable.

h. Tìm DataRow có khóa chính chứa trị truyền vào

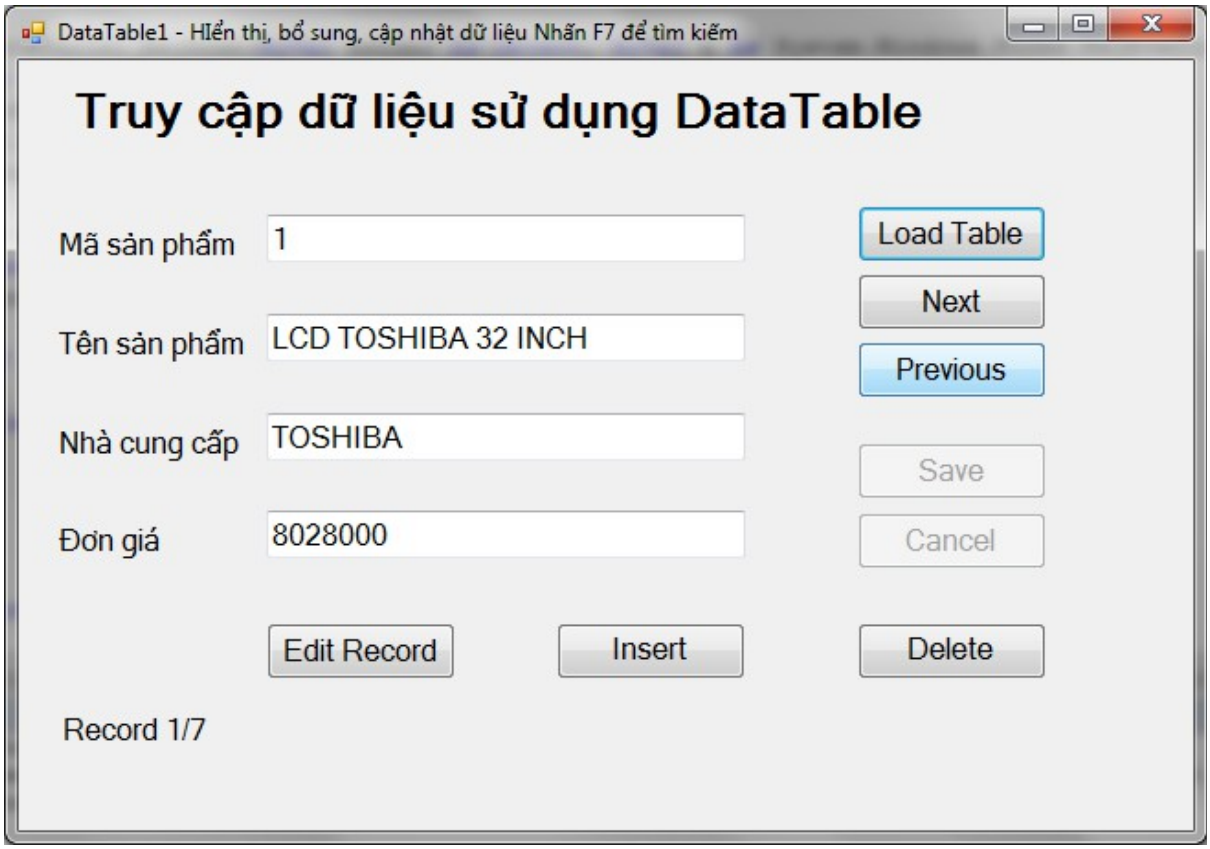
`<DataTable>.Rows.Find(<trị>)`

Phương thức trả về dòng chứa <trị> trên cột khóa chính. Nếu không tìm thấy trả về Nothing

`<DataTable>.Rows.Find(<mảng trị>)`

Phương thức trả về DataRow chứa bộ giá trị <mảng trị> trên các cột khóa chính. Thứ tự trị truyền vào phải theo thứ tự các cột khóa trên DataTable.

Ví dụ 5.13 - minh họa tạo thao tác với tập hợp Rows:



Nút Load Table sẽ nạp các bản ghi của bảng sanpham vào một dataTable. Nút Next và Previous sẽ chuyển đến các bản ghi tiếp theo hoặc trước đó. Khi nhấn nút Insert các hộp textbox sẽ được xóa trắng để nhập dữ liệu mới vào. Nhấn Save để ghi lại thông tin được nhập.

Nút Delete cho phép xóa bản ghi hiện thời (có hiển thị hộp thoại yêu cầu người dùng xác nhận xóa). Nút Edit cho phép sửa chữa thông tin của bản ghi hiện thời thông qua các hộp textbox tương ứng.

Sau khi đã nạp dữ liệu bằng nút Load Table, nếu nhấn F7 người sử dụng có thể tìm kiếm thông tin về một sản phẩm nào đó bằng cách nhập vào mã sản phẩm cần tìm.

Phần khai báo các biến chung:

```
Dim da1 As New SqlClient.SqlDataAdapter("select * from
sanpham", Chuoi_ket_noi)
Dim dst As New DataSet
Dim dt1 As New DataTable
'khai báo chỉ số bản ghi hiện thời và tổng số bản ghi
Dim chiso, tongso As Int32
'khai báo chế độ Insert hoặc Edit hoặc Delete
Dim chedo As String
```

Để hiển thị dữ liệu của bản ghi hiện thời chương trình xây dựng một thủ tục có tên Filltextbox như sau:

```
Private Sub FillTextBox()
   txtmasp.Text = dt1.Rows(chiso)("masp").ToString
   txttensp.Text = dt1.Rows(chiso)("tensp").ToString
   txtnhacc.Text = dt1.Rows(chiso)("nhacc").ToString
   txtdongia.Text = dt1.Rows(chiso)("dongia").ToString
   Label6.Text = "Record " & chiso + 1 & "/" & tongso
End Sub
```

Đoạn mã cho biến cố Click của nút Load Table:

```
dst.Clear()
'tạo cấu trúc cho bảng sản phẩm của dst để lấy khóa chính
dùng cho việc tìm kiếm theo mã sản phẩm
da1.FillSchema (dst, SchemaType.Mapped, "sanpham"
da1.Fill(dst, "sanpham")
dt1 = dst.Tables("sanpham")
tongso = dt1.Rows.Count
FillTextBox()
btedit.Enabled = True
btinsert.Enabled = True
btdelete.Enabled = True
Dim bo_lenh As New SqlClient.SqlCommandBuilder(da1)
```

Biến cố Click của nút Next và nút Previous

```
Private Sub btnext_Click...
    If chiso < tongso - 1 Then chiso += 1 Else chiso = 0
    FillTextBox()
End Sub
```

```
Private Sub btpr_Click...
   If chiso > 0 Then chiso -= 1
   FillTextBox()
End Sub
```

Đoạn mã cho biến cố Click của các nút Insert, Delete, Save:

```
Private Sub btinsert_Click....
   txtmasp.Clear()
   txttensp.Clear()
   txtnhacc.Clear()
```

```
    txtdongia.Clear()

    txtmasp.Focus()

    chedo = "Insert"
    btsave.Enabled = True
    btcancel.Enabled = True
End Sub
Private Sub btdelete_Click...
    chedo = "Delete"
    btsave.Enabled = True
    Dim row As DataRow = dt1.Rows(chiso)
    Dim traloi As String
    traloi = MsgBox("Xóa bản ghi hiện thời ?", _
    MsgBoxStyle.OkCancel)
    If traloi = 1 Then
        If chiso > 0 Then chiso -= 1 Else chiso = tongso - 1
        tongso -= 1
        FillTextBox()
        row.Delete()
    End If
End Sub
```

```
Private Sub btedit_Click...
    btsave.Enabled = True
    chedo = "Edit"
    Dim row As DataRow = dt1.Rows(chiso)
    row.BeginEdit()
    row("masp") = txtmasp.Text
    row("tensp") = txttensp.Text
    row("nhacc") = txtnhacc.Text
    row("dongia") = CSng(txtdongia.Text)
    row.EndEdit()
End Sub
```

```
Private Sub btsave_Click...
    If chedo = "Insert" Then
        If txtmasp.Text = "" And txttensp.Text = "" And _
        txtnhacc.Text = "" And txtdongia.Text = "" Then
            MsgBox("Phải nhập đủ thông tin")
        Else
            Label6.Text = "Saving ..."
            Dim row As DataRow = dt1.NewRow
            row("masp") = txtmasp.Text
```

```
            row("tensp") = txttensp.Text
            row("nhacc") = txtnhacc.Text
            row("dongia") = CSng(txtdongia.Text)
            dt1.Rows.Add(row)
         End If
         tongso += 1
         chiso = tongso - 1
         da1.Update(dst, "sanpham")
         dst.AcceptChanges()
         Label6.Text = "Record " & tongso & "/" & tongso
      End If
      If chedo = "Delete" Then
         tongso -= 1
         da1.Update(dst, "sanpham")
         dst.AcceptChanges()
      End If
      If chedo = "Edit" Then
         da1.Update(dst, "sanpham")
         dst.AcceptChanges()
      End If
End Sub
```

Xử lý việc tìm kiếm. Để các sự kiện bàn phím có tác dụng trên form, ta phải đổi thuộc tính KeyPreview của form thành True.

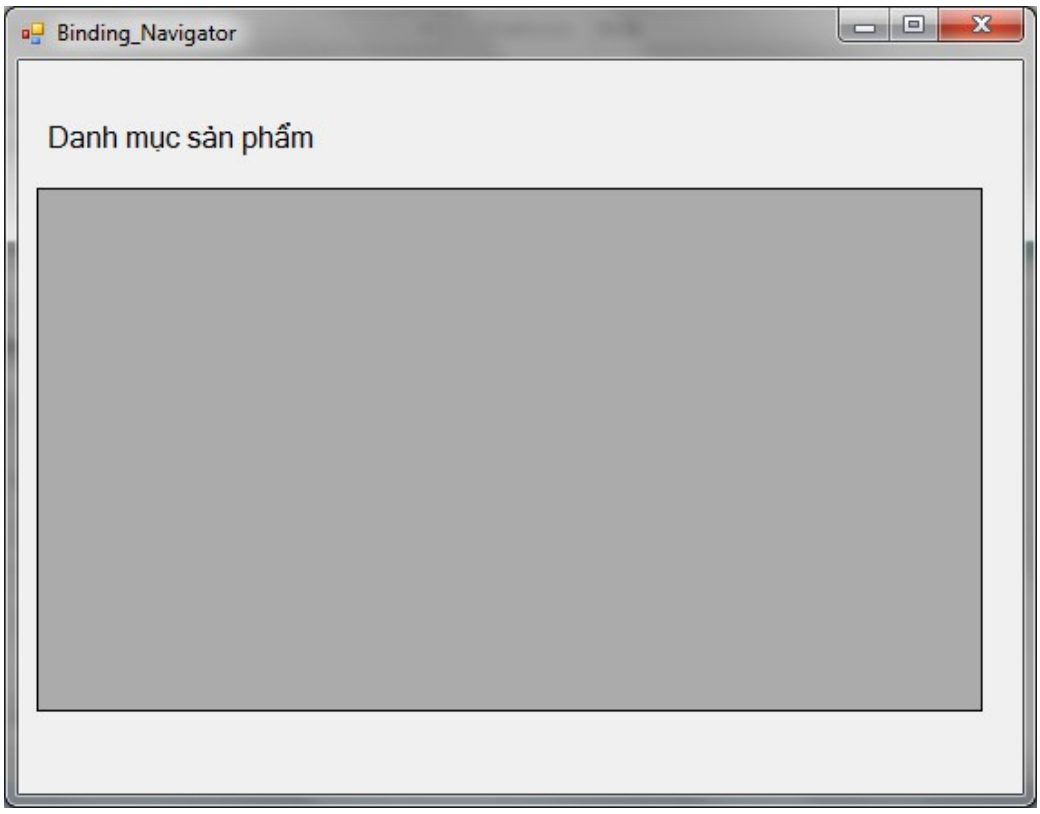
Bổ sung đoạn lệnh sau vào thủ tục KeyDown của form:

```
Dim msp As Int16
If e.KeyCode = Keys.F7 Then
   If dt1.DataSet Is Nothing Then
      MsgBox("Chưa nạp dữ liệu")
   Else
      msp = InputBox("Nhập mã sản phẩm cần tìm")
      Dim dongsp As DataRow
      dongsp = dt1.Rows.Find(msp)
      If dongsp Is Nothing Then
         MsgBox("Không tìm thấy")
      Else
         txtmasp.Text = dongsp("masp")
         txttensp.Text = dongsp("tensp")
         txtnhacc.Text = dongsp("nhacc")
         txtdongia.Text = dongsp("dongia")
      End If
   End If
```

```
End If
```

Ví dụ 5.14 - Sử dụng Binding Navigator để cập nhật dữ liệu bảng sanpham

- Bước 1: Thiết kế form gồm một label và một datagridview như hình sau:



- Bước 2: Kết nối đến cơ sở dữ liệu De1 bằng View/Server Explorer

- Bước 3: Bổ sung điều khiển SQLDataAdapter vào form và nhập lệnh sau cho màn hình Select Command:

```
Select * from sanpham
```

- Bước 4: Tạo dataset mới bằng chức năng Generated DataSet. Đặt tên cho Dataset là dst11.

- Bước 5: Bổ sung điều khiển Binding Source vào form. Khai báo thuộc tính DataSource của nó là dst11 và thuộc tính DataMember là sanpham.

- Bước 6: Kéo điều khiển Binding Navigator vào form (bên trên label danh mục sản phẩm). Khai báo thuộc tính Binding Source của nó là BindingSource1 (tên của BindingSource).

- Bước 7: Khai báo thuộc tính DataSource của lưới là BindingSource1.

- Bước 7: Bổ sung đoạn lệnh dưới đây vào thủ tục Load của Form (đoạn lệnh này đổ dữ liệu từ SQLDataAdapter1 vào dst11) và tự động sinh các lệnh khác cho SQLDataAdapter1

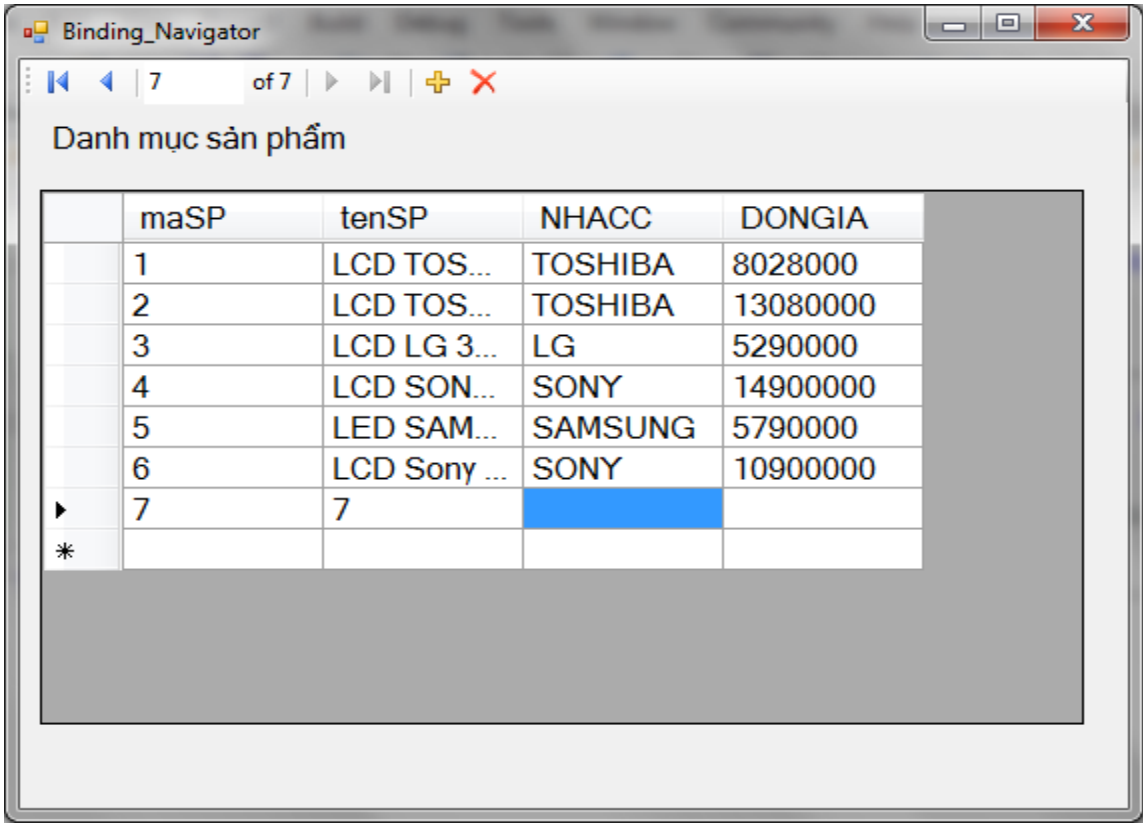
```
Dst111.Clear()
```

```
SqlDataAdapter1.Fill(Dst111, "sanpham")
Dim bo_lenh As New SqlClient.SqlCommandBuilder(SqlDataAdapter1)
```

Các lệnh cập nhật cho dataset và dataAdapter sẽ được đưa vào biến cố FormClosed:

```
    SqlDataAdapter1.Update(Dst111, "sanpham")
    Dst111.AcceptChanges()
```

Khi chạy, form sẽ được thể hiện như hình dưới đây. Ta có thể dùng các công cụ Add New và Delete của BindongNavigator để thêm hoặc xóa bản ghi; dùng các nút Move First, Move Previous, Move Next và Move Last để di chuyển con trỏ bản ghi.



5.8 Tạo báo biểu

5.8.1 Crystal Report

Crystal Report là phần mềm thiết kế báo biểu chuyên nghiệp được tích hợp trong các phiên bản của Visual Studio. Phiên bản Studio .NET của Microsoft được tích hợp Crystal Report 8.5.

Bản thân Crystal Report là một phần mềm tạo báo biểu độc lập với rất nhiều chức năng thiết kế báo biểu và dịch vụ. Người dùng có thể kết nối với nhiều nguồn dữ liệu khác nhau bằng các ODBC Driver.

Báo biểu khi tạo ra cũng có thể được lưu trữ thành những file .rpt độc lập, ở dạng có dữ liệu hay không có dữ liệu. Sau đó, file .rpt có thể được chuyển tới người dùng khác và mở bằng Crystal Report hay có thể kết hợp với các ứng dụng viết bằng Visual Basic, Visual C++.

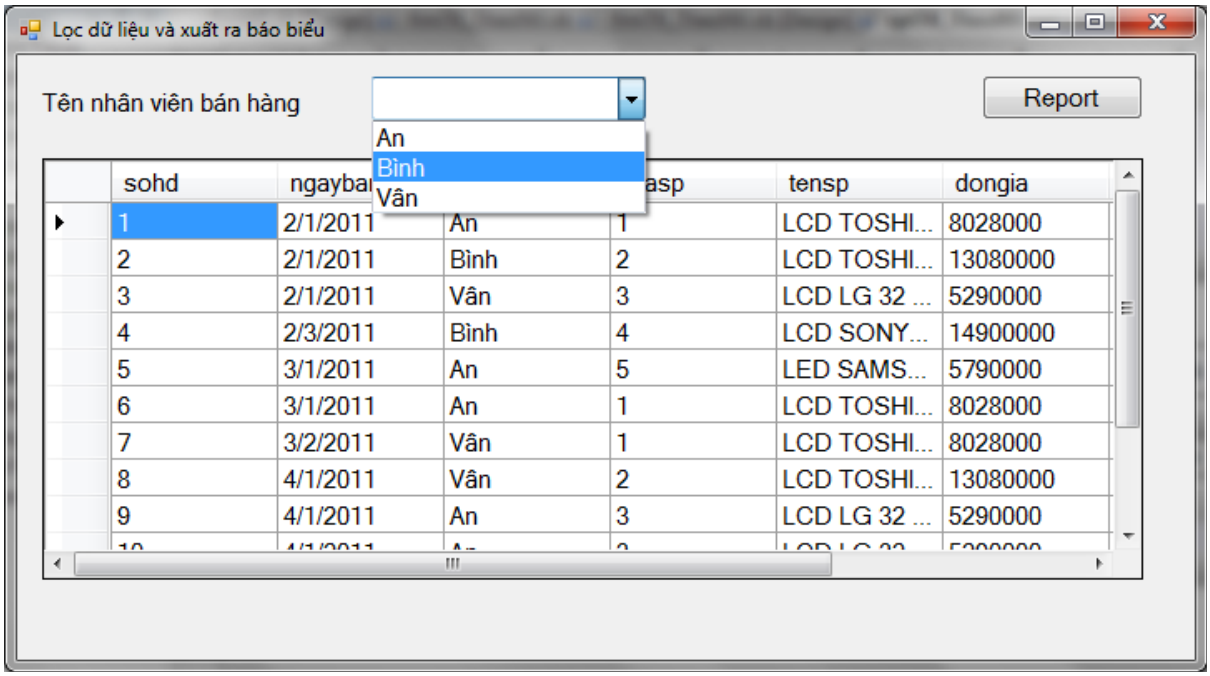
Xét về mặt thiết kế báo biểu, Crystal Report cung cấp đầy đủ các chức năng đinh dạng dữ liệu và các chức năng phân nhóm, tính toán, sub – report và kể cả khả năng lập trình bằng formula dựa trên các fomula field. Người dùng ngoài việc sử dụng formula field còn có thể tự xây dựng bộ thư viện hàm của riêng mình và đưa vào Crystal Report thông qua các DLL. Bên cạnh khả năng thiết kế báo biểu thông thường, Crystal Report còn cung cấp chức năng thiết kế biểu đồ dựa trên nguồn dữ liệu lấy từ CSDL.

Xét về mặt sử dụng báo biểu, công cụ hiển thị của Crystal Report cho phép người dùng tương tác rất linh hoạt. Báo biểu hiển thị có thể được lọc lại các dữ liệu cần thiết hay xem một phần báo biểu bằng cách sử dụng cấu trúc hiển thị dữ liệu dạng cây. Các Section trong báo biểu cũng có thể mở rộng hay thu hẹp để hiển thị hay che bớt những dữ liệu không cần thiết. Một khi báo biểu đã được xây dựng, người dùng còn có thể Export sang các dạng file khác như Word, Excel, HTML,…

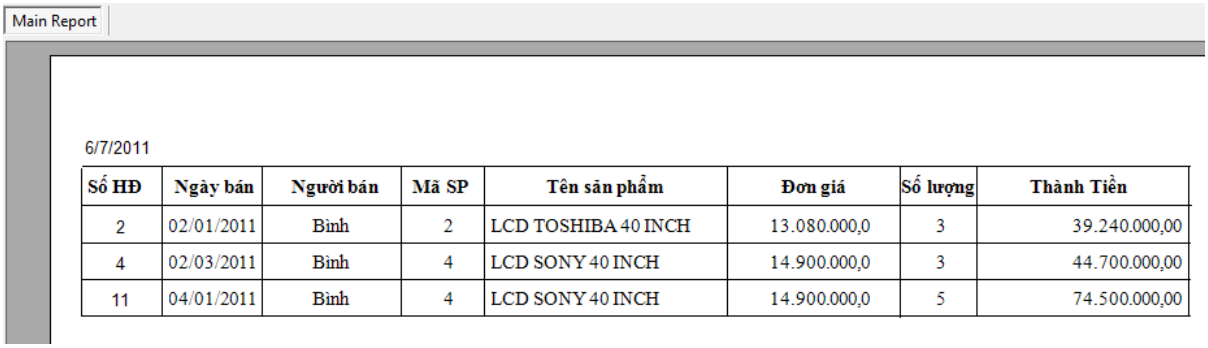
Bằng cách tích hợp Crystal Report 8.5, Visual Studio .NET đem lại cho người dùng một công cụ xây dựng báo biểu hiệu quả, tiết kiệm nhiều thời gian so với việc phải sử dụng các đối tượng in ấn để tự phát sinh báo biểu. Chúng ta có thể sử dụng Report Expert để tạo ra báo biểu dựa vào wizard và template định sẵn hay thiết kế chi tiết báo biểu bằng tay. Trong giáo trình này chỉ hướng dẫn cách tạo báo biểu bằng wizard.

5.8.2 Hướng dẫn sử dụng report wizart.

Thiết kế form lọc dữ liệu theo mẫu sau.



Form cho phép người sử dụng chọn tên một nhân viên bán hàng trong cboten và hiển thị thông tin chi tiết về những hóa đơn bán hàng của nhân viên đó. Khi nhấn nút Report ta sẽ xuất dữ liệu trên form sang một report được thể hiện như hình sau:



Phần hiển thị và lọc dữ liệu được thực hiện thông qua đối tượng dataAdapter và đối tượng DataSet như đã hướng dẫn trong các ví dụ trước. Bạn có thể tham khảo phần code dưới đây với da là tên của DataAdapter và dst là tên của DataSet. Form có tên ngầm định là form1

```
'khai báo biến tầm vực đơn thể - biến chung của class
Dim lenh1 As String = "select banhang.sohd, banhang.ngayban," & _
"banhang.nguoiban, banhang.masp,sanpham.tensp,sanpham.dongia," & _
"banhang.soluong, banhang.thanhtien from banhang,sanpham " & _
" where banhang.masp=sanpham.masp"
Dim lenh2 As String
Dim da As New SqlClient.SqlDataAdapter(lenh1, Chuoi_ket_noi)
Dim dst As New DataSet
```

```
' Thủ tục Load của form
Private Sub Form1_Load...
   dst.Clear()
   da.Fill(dst, "bangthongke")
   dg.DataSource = dst
   dg.DataMember = "bangthongke"

   Tao_ket_noi()
   If Ket_noi.State = ConnectionState.Closed Then Ket_noi.Open()
   Dim comm1 As New SqlClient.SqlCommand _
   ("select distinct nguoiban from banhang", Ket_noi)
   Dim reader1 As SqlClient.SqlDataReader = _
   comm1.ExecuteReader(CommandBehavior.CloseConnection)
   While reader1.Read
      cboten.Items.Add(reader1.Item(0))
   End While
   reader1.Close()
End Sub
```

```
'Thủ tục chọn một mục trong cbo_ten
Private Sub cboten_SelectedIndexChanged ...
   lenh2 = " and banhang.nguoiban='" & cboten.Text & "'"
   da.SelectCommand.CommandText = lenh1 & lenh2
   dst.Clear()
   da.Fill(dst, "bangthongke")
End Sub
```

Vấn đề chỉ còn lại cách xây dựng báo biểu thống kê chi tiết các lần bán hàng theo tên nhân viên và viết code cho nút Report. Hãy thực hiện theo các bước sau:

- **Bước 1**: Trong SQL Server tạo thủ tục nhận tên người bán làm tham số và trả về thông tin bán hàng của nhân viên đó. Thủ tục có tên thongkeBHtheoNV và được tạo như sau:

```
Create proc ThongkeBHtheoNV(@tennb nvarchar(50))
as
begin
   select banhang.sohd,
banhang.ngayban,banhang.nguoiban,
   banhang.masp,sanpham.tensp,sanpham.dongia,
   banhang.soluong,banhang.thanhtien
   from sanpham,banhang
```

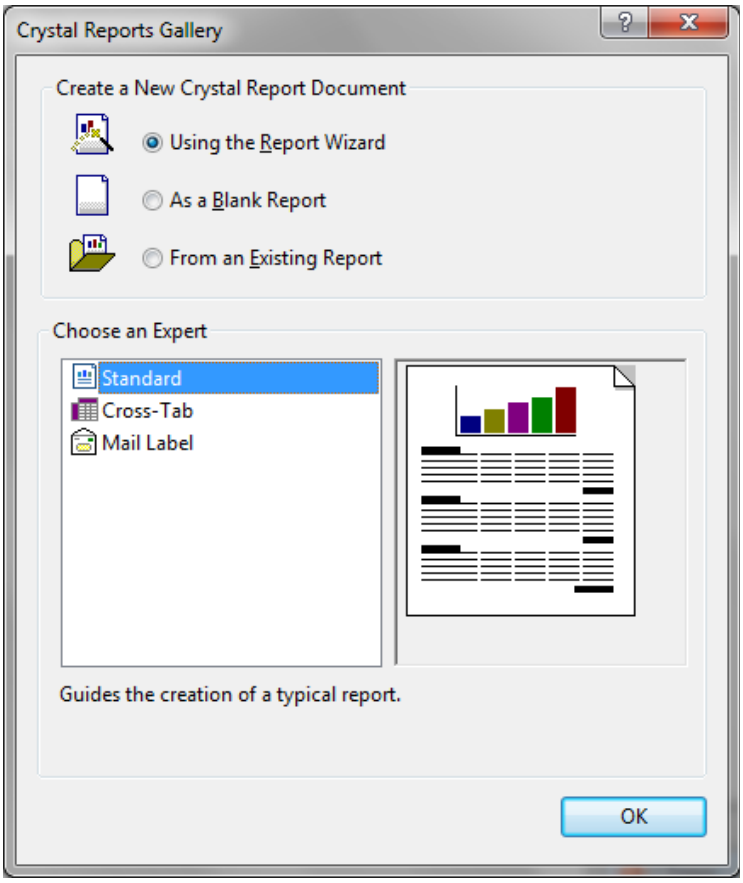
```
    where banhang.masp=sanpham.masp and
    banhang.nguoiban=@tennb
end
```

Biên dịch thủ tục trên (nhấn F5)

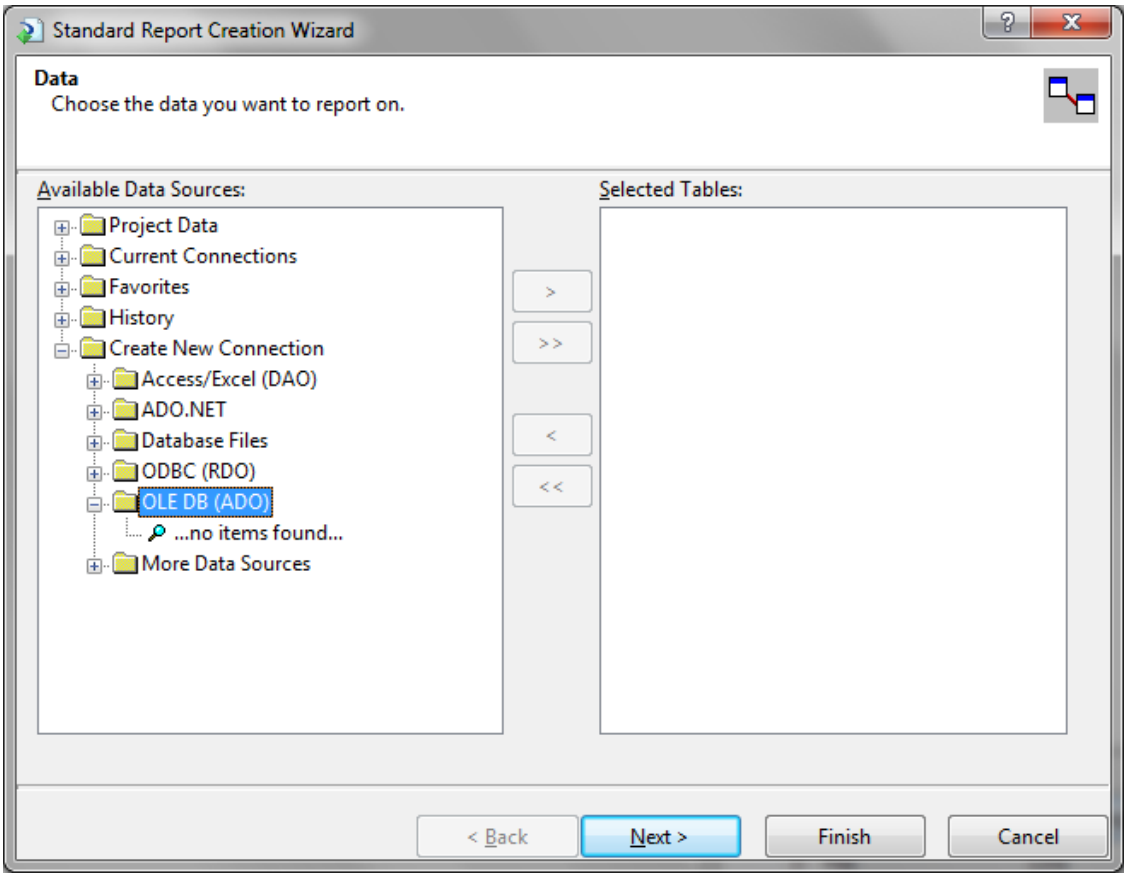
- Bước 2:

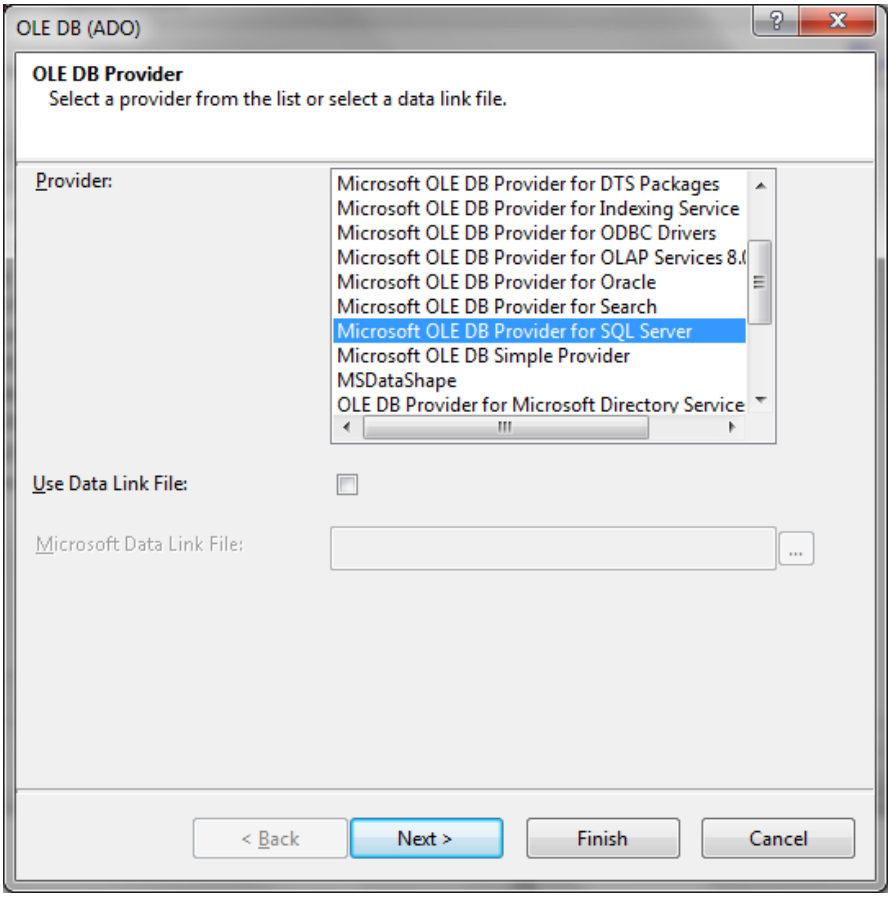
+ Tạo Crystal Report (nhấn phải chuột vào tên project trong cửa sổ Solution Explorer, chọn Add, chọn Crystal Report).

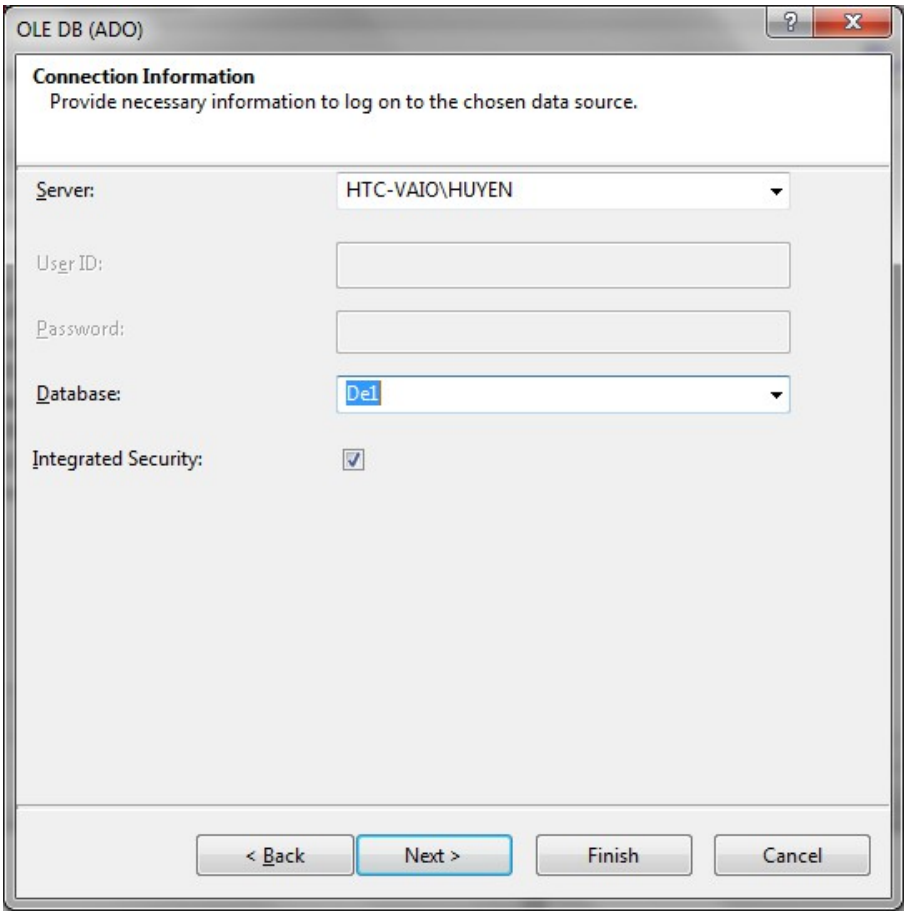
+ Đặt tên report là rptTK_TheoNV. Cửa sổ Crystal Reports Gallery xuất hiện như hình dưới đây. Chọn kiểu Standard và nhấn OK.



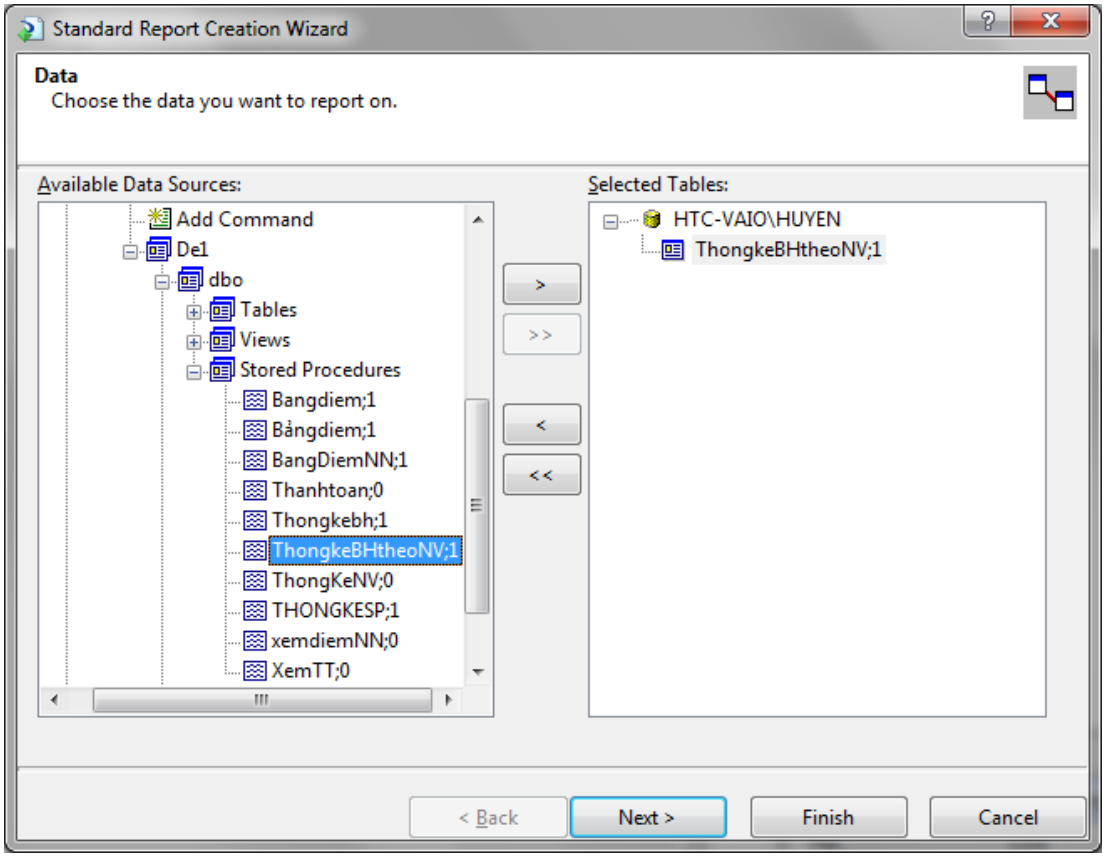
- Bước 3: Nháy đúp vào mục Create New Connection; chọn OLEDB, chọn Microsoft OLEDB Provider for SQL Server, chọn Next


Standard Report Creation Wizard
Data
Choose the data you want to report on.
Available Data Sources:
Project Data
Current Connections
Favorites
History
Create New Connection
Access/Excel (DAO)
ADO.NET
Database Files
ODBC (RDO)
OLE DB (ADO)
...no items found...
More Data Sources
Selected Tables:
>
>>
<
<<
< Back
Next >
Finish
Cancel

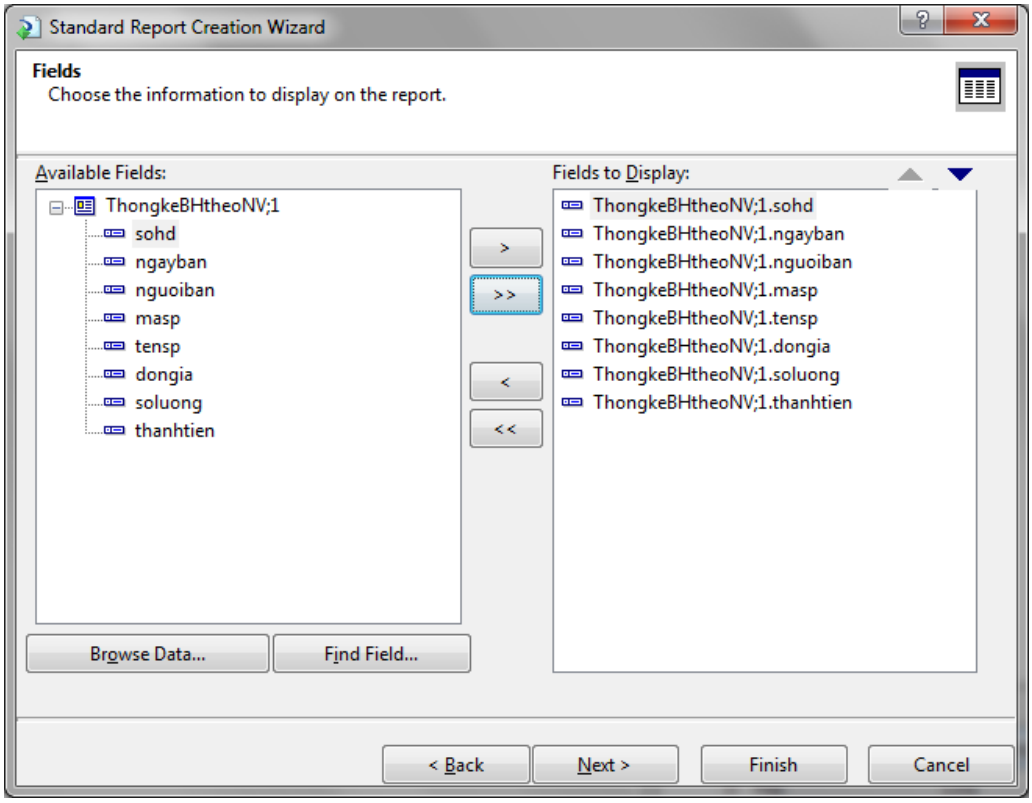

OLE DB (ADO)
OLE DB Provider
Select a provider from the list or select a data link file.
Provider:
Microsoft OLE DB Provider for DTS Packages
Microsoft OLE DB Provider for Indexing Service
Microsoft OLE DB Provider for ODBC Drivers
Microsoft OLE DB Provider for OLAP Services 8.(
Microsoft OLE DB Provider for Oracle
Microsoft OLE DB Provider for Search
Microsoft OLE DB Provider for SQL Server
Microsoft OLE DB Simple Provider
MSDataShape
OLE DB Provider for Microsoft Directory Service
Use Data Link File:
Microsoft Data Link File:
...
< Back
Next >
Finish
Cancel



+ Chọn tên Server và tên Database. Nhấn Next sang màn hình tiếp theo.

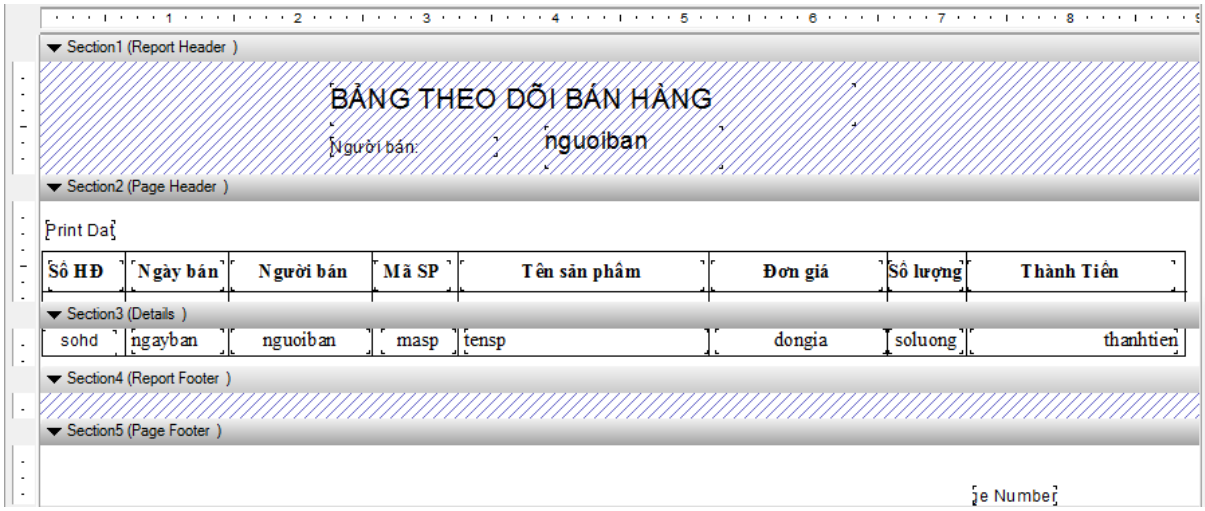


+ Tìm tên thủ tục ThongkeBHtheoNV trong danh mục các thủ tục nội tại của cơ sở dữ liệu. Add sang bảng Selected Table. Nhấn Next để tiếp tục.



+ Add toàn bộ các trường của thủ tục ThongkeBHtheoNV sang cửa sổ bên phải. Nhấn Finish để hoàn tất bước 3. Report rptTK_theoNV đã xuất hiện trên project.

- Bước 4: Định dạng lại report như hình dưới đây. Section1 là phần tiêu đề của báo biểu; Section 2 là tiêu đề các cột của bảng (lặp lại khi hết một trang); Section3 là phần dữ liệu tương ứng với một bản ghi.



+ Dùng Text Object để đặt các label lên báo biểu; Line Object để vẽ các đường thẳng; dùng cửa sổ Field Explorer để chèn các trường vào báo biểu.

+ Để định dạng nội dung, font chữ, kiểu dữ liệu của tiêu đề cột và dữ liệu trong cột, nhấn phải chuột vào đối tượng và chọn các lệnh cần thiết.

Thủ thuật: Để gióng hàng (hoặc đặt cùng kích cỡ) cho nhiều đối tượng, bạn chọn chúng và nhấn phải chuột chọn Align hoặc Size.

+ Để xem trước báo biểu, chọn nút Main Report Preview.

- Bước 5: Tạo form làm lớp cơ sở cho báo biểu:

+ Tạo form mới, đặt tên form là frmTK_theoNV.

+ Đưa vào form một điều khiển CrystalReportViewer. Điều khiển có tên ngầm định là CrystalReportViewer1

- Bước 6: Viết code cho nút Report và chạy thử chương trình

```
Dim myreport As New rptTK_TheoNV
myreport.SetDataSource(dst.Tables("bangthongke"))
Dim f As New frmTK_TheoNV
f.CrystalReportViewer1.ReportSource = myreport
f.ShowDialog()
```

NỘI DUNG PHẦN THẢO LUẬN

1. Các phương pháp xây dựng form hiển thị dữ liệu
2. Mô hình thiết kế form lọc dữ liệu
3. Các phương pháp xây dựng form cập nhật dữ liệu
4. Tính toán trên cơ sở dữ liệu
5. Xử lý các ràng buộc trên cơ sở dữ liệu
6. Xử lý trùng khóa trên cơ sở dữ liệu.

TÓM TẮT NỘI DUNG CỐT LÕI

Trong chương này sinh viên cần chú ý đến các nội dung sau:

• Tạo module kết nối cơ sở dữ liệu

• Hiển thị dữ liệu bằng Data Reader

• Hiển thị dữ liệu bằng DataAdapter kết hợp với DataSet, DataTable

• Tính toán, cập nhật dữ liệu bằng đối tượng Command

• Tính toán, cập nhật dữ liệu bằng đối tượng DataSet, DataTable

• Cập nhật dữ liệu bằng Binding Navigator

• Tạo và sử dụng báo biểu.

BÀI TẬP THỰC HÀNH

1. Tạo cơ sở dữ liệu SQL có tên De1 với cấu trúc các bảng và dữ liệu như sau:

SanPham (**MaSP** int, TenSP nvarchar(50), NhaCC nvarchar(30), DonGia decimal)

BanHang (**SoHD** int, NgayBan datetime, MaSP int ràng buộc khóa ngoại tham chiếu đến MaSP trong bảng SanPham, NguoiBan nvarchar(50), SoLuong int, ThanhTien decimal)

SanPham

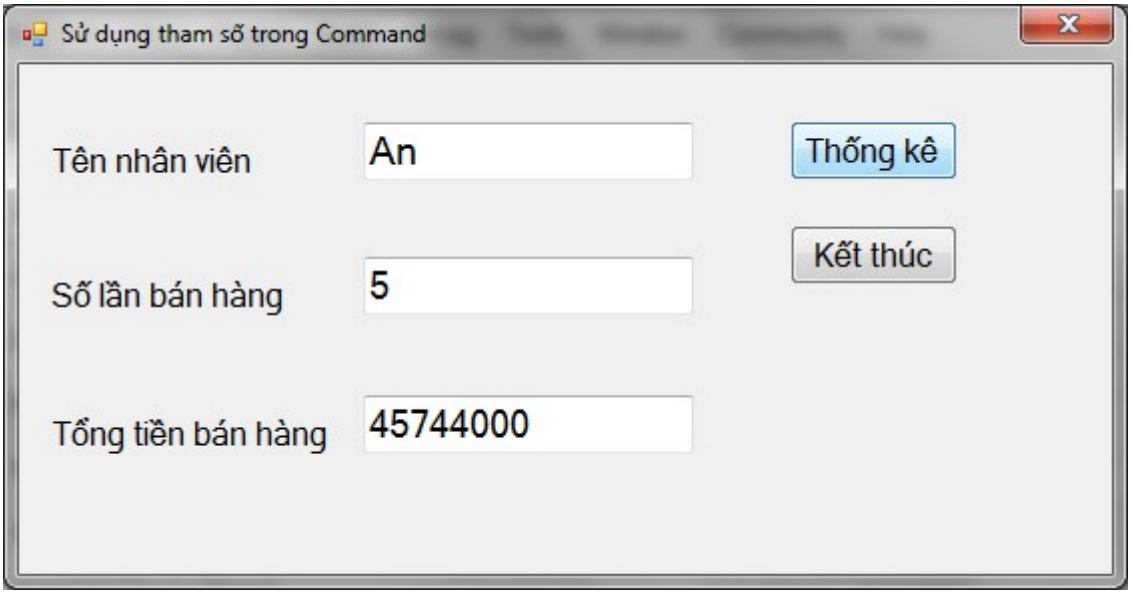
MaSP (mã sản phẩm	TenSP (tên sản phẩm)	NhaCC (nhà cung cấp)	DonGia (đơn giá)
1	LCD Toshiba 32 inch	Toshiba	6690000
2	LCD Toshiba 40 inch	Toshiba	10900000
3	LCD LG 32 inch	LG	5290000
4	LCD Sony 40 inch	Sony	14900000
5	LED Samsung 22 inch	Samsung	5790000

BanHang

SoHD	NgayBan (ngày bán)	MaSP (mã sản phẩm)	Ngươiban (người bán)	SoLuong (số lượng)	ThanhTien (thành tiền)
1	'2/1/2011'	1	An	2	
2	'2/1/2011'	2	Bình	3	
3	'2/1/2011'	3	Vân	2	
4	'2/3/2011'	4	Bình	3	
5	'3/1/2011'	5	An	1	
6	'3/1/2011'	1	An	1	
7	'3/2/2011'	1	Vân	2	
8	'4/1/2011'	2	Vân	2	
9	'4/1/2011'	3	An	2	
10	'4/1/2011'	3	An	1	
11	'4/1/2011'	4	Bình	5	
12	'4/1/2011'	4	Vân	3	

2. Tạo project mới. Trên đó tạo module và viết thủ tục Tao_Ket_noi với cơ sở dữ liệu De1 (xem mẫu trong ví dụ 5.2b).

3. Tạo form mới theo mẫu dưới đây. Form cho phép nhập tên một nhân viên bán hàng và thống kê số lần bán hàng, tổng số tiền bán hàng. (xem mẫu trong ví dụ 5.3, 5.4)

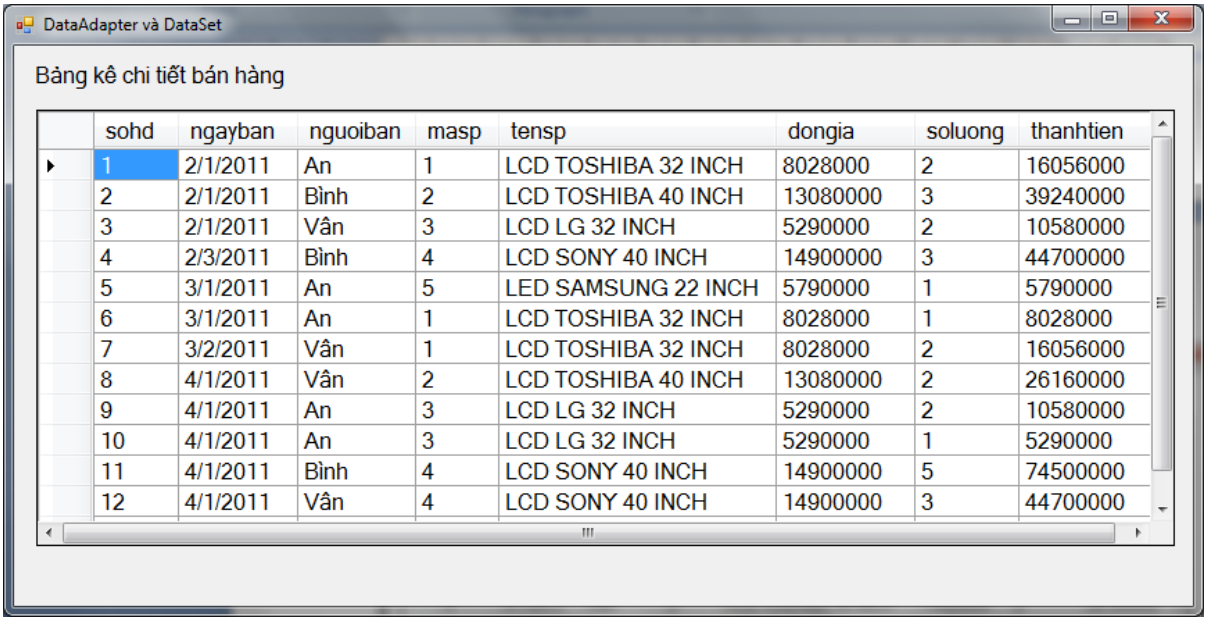


4. Thiết kế form theo mẫu của ví dụ 5.5

5. Tạo thủ tục thongkenv và thêm vào project một form theo ví dụ 5.6

6. Bổ sung form và thiết kế theo mẫu của ví dụ 5.7

7. Tạo form hiển thị bảng kê chi tiết bán hàng (xem mẫu ví dụ 5.8a hoặc 5.8b)



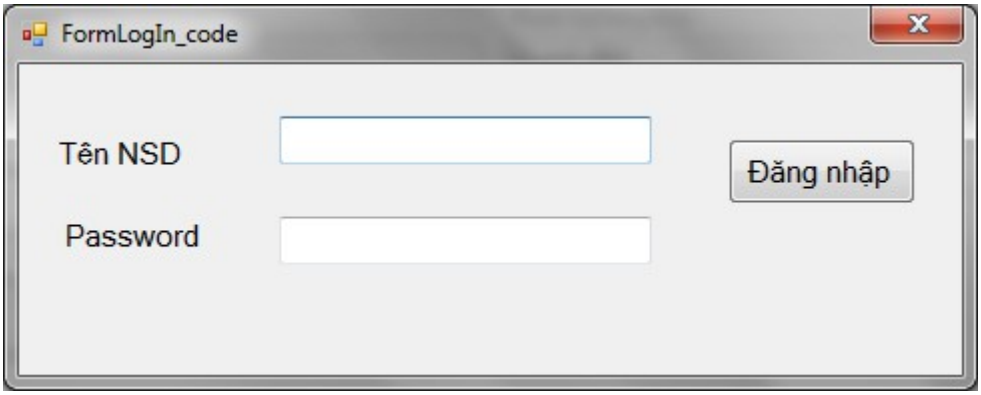
sohd	ngayban	nguoiban	masp	tensp	dongia	soluong	thanhtien
1	2/1/2011	An	1	LCD TOSHIBA 32 INCH	8028000	2	16056000
2	2/1/2011	Bình	2	LCD TOSHIBA 40 INCH	13080000	3	39240000
3	2/1/2011	Vân	3	LCD LG 32 INCH	5290000	2	10580000
4	2/3/2011	Bình	4	LCD SONY 40 INCH	14900000	3	44700000
5	3/1/2011	An	5	LED SAMSUNG 22 INCH	5790000	1	5790000
6	3/1/2011	An	1	LCD TOSHIBA 32 INCH	8028000	1	8028000
7	3/2/2011	Vân	1	LCD TOSHIBA 32 INCH	8028000	2	16056000
8	4/1/2011	Vân	2	LCD TOSHIBA 40 INCH	13080000	2	26160000
9	4/1/2011	An	3	LCD LG 32 INCH	5290000	2	10580000
10	4/1/2011	An	3	LCD LG 32 INCH	5290000	1	5290000
11	4/1/2011	Bình	4	LCD SONY 40 INCH	14900000	5	74500000
12	4/1/2011	Vân	4	LCD SONY 40 INCH	14900000	3	44700000

8. Bổ sung thêm các form tương ứng với các mẫu của ví dụ 5.9, 5.10, 5.11, 5.12, 5.13, 5.14

9. Thêm form MDI có tên frmChinh vào project. Thiết kế menu cho form. Các mục chọn của menu cho phép gọi các form tương ứng của project.

- Thống kê, tính toán: + Ví dụ 5.4
  + Ví dụ 5.5
  + Ví dụ 5.6
- Hiển thị dữ liệu + Ví dụ 5.7
  + Ví dụ 5.8
  + Ví dụ 5.9
  + Ví dụ 5.9 b - Dùng Server Explorer để kết nối CSDL
- Lọc dữ liệu - Ví dụ 5.10
- Cập nhật dữ liệu + Thêm bản ghi - Ví dụ 5.11
  + Xóa bản ghi - Ví dụ 5.12
  + Cập nhật chung - Ví dụ 5.13
  + Binding Navigator - Ví dụ 5.14

10. Bổ sung vào cơ sở dữ liệu bảng NguoiSD với hai trường là UserName và Pass. Nhập dữ liệu cho 3 bản ghi. Thiết kế form đăng nhập cho hệ thống như mẫu sau. Form cho phép người sử dụng nhập tên và password. Khi bấm nút Đăng nhập, nếu tên hoặc password sai thì sẽ đưa ra thông báo lỗi, nếu đúng với tên và password lưu trong cơ sở dữ liệu thì chuyển đến làm việc với form frmChinh đã tạo trong câu 9.



TÀI LIỆU THAM KHẢO

Tiếng Việt

1. Trung tâm tin học Đại học Khoa học tự nhiên thành phố Hồ Chí Minh (2006), *Tài liệu hướng dẫn giảng dạy chương trình kỹ thuật viên ngành lập trình Học phần 3 Visual Basic.NET*.
2. Nguồn ebook SachVBNET.pdf (3-1-2011), *Giáo trình Lập trình hướng đối tượng với VB.NET*.
3. Nguyễn Ngọc Tuấn (2005), *Visual Studio.Net* , Nhà xuất bản Thống kê.
4. Website http://www.thayphet.net (2011).

Tiếng Anh

5. Dave Grundgeiger (2002), *Programming Visual Basic .NET*, Publisher: O'Reilly.
6. Nguồn ebook dotnet tutorial for beginner.pdf (3-1-2011), India Community Initiative, *NET Tutorial for Beginners*.
7. Nguồn ebook 2878ch06.pdf (4-2011), *Mastering™ Visual Basic® .NET - Chapter 6: A First Look at ADO.NET* , Evangelos Petroutsos; Asli Bilgin, Copyright © 2002 SYBEX Inc.
8. The National Department of Education (2008), *Introduction to VB.NET Manual*.