

Bài giảng lập trình trực quan

CHƯƠNG 1: .NET FRAMEWORK VÀ VISUAL STUDIO 2010.....	6
1.1. Tổng quan về .NET Framework	6
1.1.1. Giới thiệu về .NET Framework	6
1.1.2. Các thành phần chính của .NET Framework.....	6
1.1.3. Các giai đoạn biên dịch và thi hành chương trình.....	8
1.1.4. Kiến trúc của .NET Application.....	9
1.1.5. Các phiên bản của .NET Framework.....	10
1.2. Các công cụ lập trình và cài đặt.....	13
1.2.1. Các công cụ lập trình	13
1.2.2. Cài đặt Visual studio 2010.....	13
1.3. Khởi động Visual studio 2010 và giao diện.....	18
1.3.1. Khởi động Visual studio 2010.....	18
1.3.2. Giao diện môi trường lập trình Visual C# trên WinForm	19
CHƯƠNG 2: VIẾT CHƯƠNG TRÌNH ĐẦU TIÊN	25
2.1. Đề bài	25
2.2. Mở dự án	25
2.3. Thiết kế giao diện.....	26
2.3.1. Đặt tên và tiêu đề cho form	26
2.3.2. Thêm điều khiển hộp văn bản TextBox	26
2.3.3. Thêm điều khiển nút lệnh Button	26
2.4. Viết mã lệnh	27
2.4.1. Viết mã lệnh cho nút Display	27
2.4.2. Viết mã lệnh nút Clear.....	29
2.4.3. Viết mã lệnh nút Exit.....	29
2.5. Chạy chương trình.....	29
2.6. Dừng chương trình	29

Bài giảng lập trình trực quan

2.7. Mở dự án đã có	30
CHƯƠNG 3: CƠ BẢN VỀ NGÔN NGỮ C#	31
3.1. Đặc điểm của C#	31
3.2. Biến, hằng và các kiểu dữ liệu	31
3.2.1. Biến.....	31
3.2.2 Hằng.....	31
3.2.3. Các kiểu dữ liệu	32
3.2.4. Chuyển đổi giữa các kiểu dữ liệu	38
3.3. Hộp thoại thông báo - Message Box.....	39
3.3.1. Khái niệm.....	39
3.3.2. Hộp thông báo MessageBox.....	39
3.3.3. Hàm thông báo MessageBox	40
3.4. Các cấu trúc điều khiển.....	41
3.4.1. Câu lệnh lựa chọn if.....	41
3.4.2. Câu lệnh lựa chọn switch... case.....	41
3.4.3. Cấu trúc lặp for	42
3.4.4. Cấu trúc lặp while.....	43
3.4.5. Cấu trúc lặp do...while.....	43
3.4.6. Cấu trúc lặp foreach.....	44
3.4.7. Câu lệnh try...catch	44
3.4.8. Câu lệnh break	44
3.4.9. Câu lệnh Continue	44
3.5. Phương thức trong C#.....	44
3.5.1. Định nghĩa phương thức	44
3.5.2. Gọi phương thức	45
3.6. Gỡ rối chương trình.....	45

Bài giảng lập trình trực quan

3.6.1. Một số giải pháp gỡ rối chương trình	45
3.6.2. Dò lỗi từng dòng lệnh	46
CHƯƠNG 4: TÌM HIỂU CÁC ĐIỀU KHIỂN CƠ BẢN	50
4.1. Thuộc tính, phương thức, sự kiện và các mối quan hệ giữa chúng	50
4.2. Thuộc tính, phương thức, sự kiện của một số điều khiển cơ bản	51
4.2.1. Form.....	51
4.2.2. Hộp văn bản – TextBox.....	52
4.2.3. Nút lệnh – Button	54
4.2.4. Nhãn – Label.....	55
4.2.5. Dòng mách nước – ToolTip	55
4.2.6. Bài tập.....	56
4.3. Một số điều khiển cơ bản khác	58
4.3.1. Nhóm – GroupBox	58
4.3.2. Hộp đánh dấu – CheckBox	59
4.3.3. Nút tùy chọn – RadioButton	59
4.3.4. Hộp danh sách – ListBox.....	61
4.3.5. Hộp lựa chọn – ComboBox	65
4.3.6. Điều khiển CheckedListBox.....	68
4.3.7. Điều khiển NumericUpDown	73
4.3.8. Thanh cuộn HscrollBar và VscrollBar	75
4.3.9. Điều khiển Timer	76
4.3.10. Điều khiển RichTextBox	78
CHƯƠNG 5: CÁC HỘP HỘI THOẠI THÔNG DỤNG	79
5.1. Hộp hội thoại Open File	79
5.2. Hộp thoại SaveFile và luồng FileStream	81
5.2.1. Hộp thoại SaveFile	81

Bài giảng lập trình trực quan

5.2.2. Luồng FileStream	81
5.3. Hộp thoại Color	84
5.4. Hộp thoại Font	85
CHƯƠNG 6: MENU VÀ CÁC ĐỒ ÁN NHIỀU BIỂU MẪU	88
6.1. Điều khiển ToolStrip.....	88
6.2. Popup menu – ContextMenuStrip.....	90
6.3. Dự án nhiều biểu mẫu	92
6.3.1. Bổ sung biểu mẫu	92
6.3.2. Biểu mẫu khởi động.....	93
6.3.3. Mở biểu mẫu.....	93
6.3.4. Đóng biểu mẫu.....	94
6.3.5. Xóa biểu mẫu.....	94
CHƯƠNG 7: LẬP TRÌNH CƠ SỞ DỮ LIỆU	98
7.1. Giới thiệu về bài toán.....	98
7.1.1. Lập trình cơ sở dữ liệu và bài toán quản lý	98
7.1.2. Cách tổ chức các tài nguyên trong một dự án của bài toán quản lý	98
7.2. Cách tạo cơ sở dữ liệu (Database) trong môi trường visual studio 2010	100
7.2.1. Tạo mới một DataBase	100
7.2.2. Tạo các bảng CSDL.....	103
7.2.3. Tạo quan hệ Relationship cho CSDL	105
7.3. ADO.NET (ActiveX Data Objects for .NET Framework)	107
7.3.1. Giới thiệu ADO.NET.....	107
7.3.2. Kiến trúc của ADO.NET	109
7.3.3. Các bước làm việc với CSDL sử dụng ADO.NET	110
7.3.4. Các đối tượng trong ADO.NET và một số phương thức.....	110

Bài giảng lập trình trực quan

7.4. Làm việc với CSDL SQL qua các đối tượng: SqlConnection, SqlDataAdapter, SqlCommand.....	120
7.4.1. Đối tượng SqlConnection	120
7.4.2. Đối tượng SqlDataAdapter và SqlCommand	121
7.4.3. Các thao tác dữ liệu	122
7.5. Làm việc với Dataset và DataTale	134
7.5.1. Cấu trúc của Dataset và DataTable.....	134
7.5.2. Các phương thức	134
7.5.3. Tự tạo DataTable	135
7.6. Xuất dữ liệu ra Excel	136
7.6.1. Làm việc với đối tượng Excel	136
7.6.2. Ví dụ	139
7.7. In báo cáo với CrystalReport	144
7.7.1. Cài đặt Crystal report cho visual studio 2010	145
7.7.2. Làm việc với CrystalReport thông qua DataSet	145
7.7.3. Điều khiển CrystalReportViewer	Error! Bookmark not defined.
7.7.4. Bài tập.....	Error! Bookmark not defined.

Bài giảng lập trình trực quan

<http://www.youtube.com/watch?v=J2I6P1kNdjE&index=1&list=PLNudknPLnZhn6Z1nVu21mdf88MFD8VTNF>

Cao Thị Luyên – cnpm

caoluyengt@gmail.com

0912 403345

Tú, Hiệu, Khánh, Lâm, Giang, Mạnh, Văn Tiến, Phượng, Mạnh
Tiến (1) cntt.ltk20@gmail.com

Giờ học: 6h30-8h30, P401A2

CHƯƠNG 1: .NET FRAMEWORK VÀ VISUAL STUDIO 2010

1.1. Tổng quan về .NET Framework

1.1.1. Giới thiệu về .NET Framework

.NET Framework là môi trường mà đoạn mã của bạn sẽ hoạt động. Đây có nghĩa là .NET sẽ quản lý việc thi hành chương trình (bao gồm việc khởi động chương trình, cấp phép hoạt động, cấp phát bộ nhớ, cho thu hồi bộ nhớ khi không dùng đến...). Tuy nhiên, ngoài việc tiến hành những công tác kể trên, .NET còn chuẩn bị sẵn một thư viện lớp gọi là **.NET base class** (lớp cơ bản .NET) cho phép thực hiện vô số các tác vụ trên Windows.

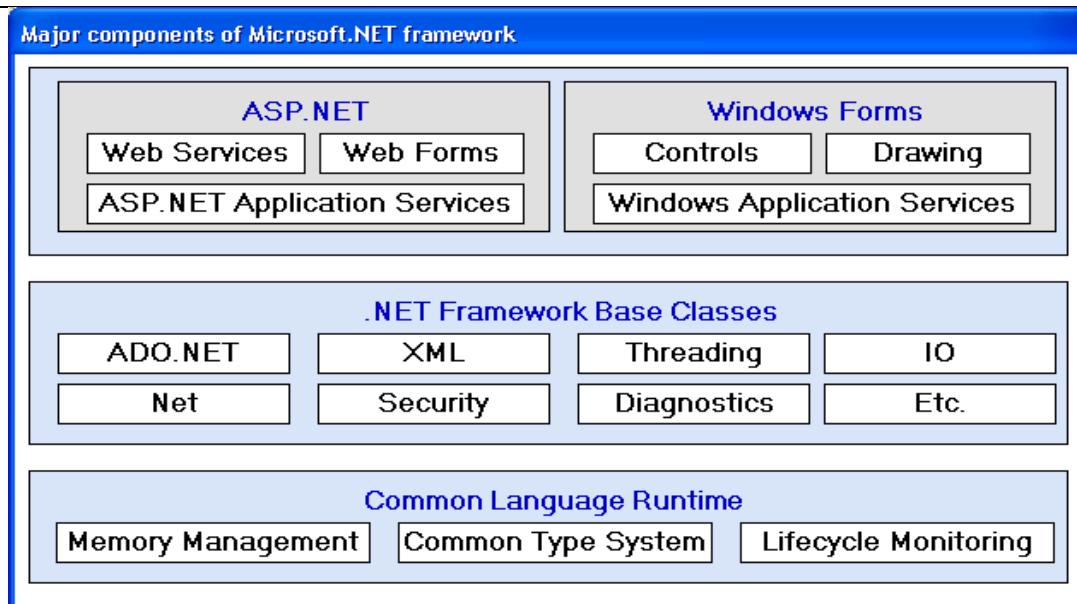
Nói tóm lại, .NET giữ 2 vai trò: quản lý việc thi hành chương trình của bạn và cung cấp các dịch vụ mà chương trình của bạn cần đến.

1.1.2. Các thành phần chính của .NET Framework

.NET gồm có 2 thành phần: Framework và Intergrated Development Environment (IDE). Framework cung cấp tất cả những gì cần thiết căn bản để chương trình của bạn có thể thi hành được, còn IDE cung cấp một môi trường giúp ta phát triển dễ dàng, nhanh chóng hơn. Nếu không có IDE ta vẫn có thể dùng notepad và command line để triển khai nhưng chậm hơn. Trong .NET thì với bất kỳ một ngôn ngữ lập trình nào bạn dùng như C#, VB.NET... đều dùng cùng một IDE.

Hình vẽ sau đây, cho thấy các thành phần chính của .NET Framework

Bài giảng lập trình trực quan



Hình 1.1: Các thành phần chính của .NET Framework

Common Language Runtime (CLR)

CLR được xem như là linh hồn của kiến trúc .NET, nó là bộ phận lo quản lý việc thi hành mã của bạn: nạp chương trình, cho chạy đoạn mã theo các mạch trình (thread) nhất định cũng như quản lý các mạch trình đó và cung cấp tất cả các dịch vụ hỗ trợ cho đoạn mã của bạn có thể thực thi. Tóm lại, CLR tạo ra môi trường cho đoạn mã của bạn có thể thực thi.

Tất cả các đoạn mã mà được chạy trên .NET đều được gọi là đoạn mã được quản lý.

Các chương trình .NET không được biên dịch thành tập tin khả thi mà được biên dịch thành một ngôn ngữ trung gian gọi là Microsoft Intermediate Language (MSIL) hay gọi tắt là Intermediate Language (IL).

Khi bạn biên dịch đoạn mã được quản lý thì trình biên dịch sẽ cho ra IL, rồi CLR sẽ dịch IL thành mã máy cụ thể.

.NET Framework base classes (các lớp cơ sở trong .NET Framework)

Tầng giữa của .NET Framework gồm những dịch vụ tổng quát thiết yếu của hệ thống. Các dịch vụ này có thể dùng với tất cả các ngôn ngữ lập trình. Chúng bao gồm:

- + ADO.NET: Để truy cập dữ liệu.
- + I/O: Để thực hiện vào ra dữ liệu.
- + XML: Định dạng dữ liệu thông qua các thẻ định dạng.

+ ...

ADO.NET là công nghệ truy cập cơ sở dữ liệu mới nhất của Microsoft. Mặc dù ADO.NET được viết tắt từ cụm từ Active Data Objects for .NET Framework, nhưng có lẽ nó được đặt sai tên vì ADO.NET không phải là một công nghệ Active/COM (Component Object Model). ADO.NET là một tập hợp các lớp hướng đối tượng, cung cấp một tập hợp phong phú các cấu kiện dữ liệu cho phép tạo các ứng dụng truy cập dữ liệu với hiệu năng cao, mức độ bảo mật tốt và có thể dễ dàng tăng quy mô theo thời gian trong một môi trường Client-server hay môi trường phân tán trên Internet hay Intranet. Trên mô hình ADO.NET, các ứng dụng sẽ kết nối với nguồn dữ liệu khi đọc hoặc nhật tu dữ liệu, sau đó thì đóng đường dây kết nối lại. Điểm này rất quan trọng vì trong các ứng dụng Client – server hoặc phân tán, việc để mở liên tục các đường dây kết nối là một tiêu hao nguồn lực vô lối.

XML là một ngôn ngữ tổng quát định dạng dữ liệu thông qua các thẻ tự định nghĩa, nó là một chuẩn mới mà ngày càng được định dạng rộng rãi.

Ngoài ra còn có các lớp hỗ trợ lập trình đa luồng, xử lý vào ra dữ liệu (I/O), hỗ trợ bảo mật.

ASP.NET và Windows Forms

Tầng trên cùng nhất liên quan đến người sử dụng và giao diện chương trình bao gồm ASP.NET và Windows Forms.

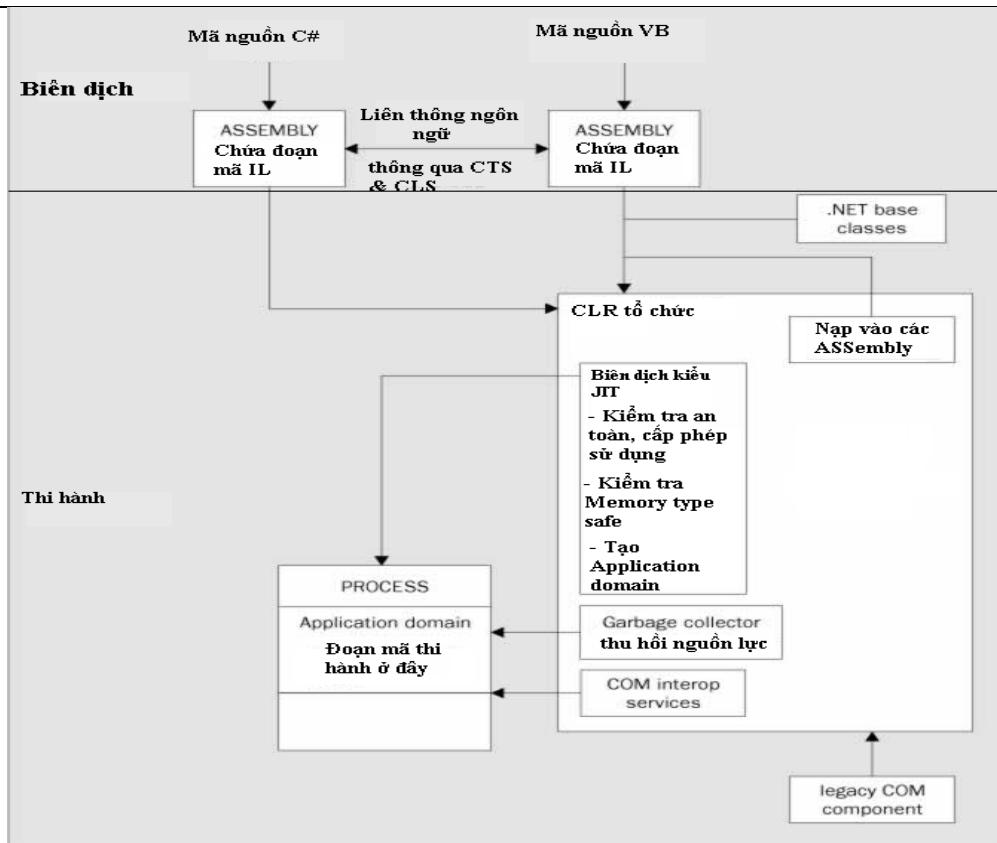
Windows Forms (còn gọi tắt là Winform) là một cách mới và hay hơn để làm giao diện trong Win32. Winforms có nhiều điểm khác với VB6.

ASP.NET bao gồm Web forms và web services, các ứng dụng ASP.NET.

1.1.3. Các giai đoạn biên dịch và thi hành chương trình

Hình vẽ sau sẽ cho thấy quá trình này, các ô hình chữ nhật tượng trưng cho các cấu kiện (Component) chính tham gia vào việc biên dịch và thi hành chương trình, trong khi những mũi tên cho biết những công tác được thực hiện.

Bài giảng lập trình trực quan



Hình 1.2: Các giai đoạn biên dịch và thi hành chương trình

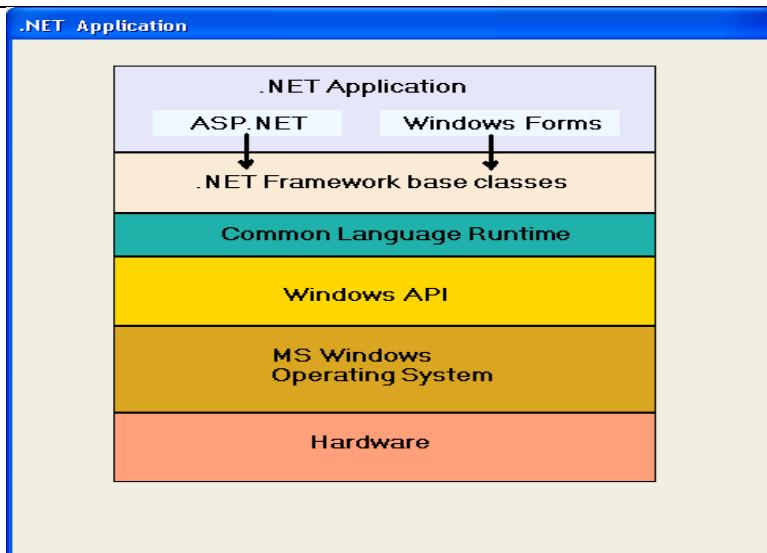
Phần trên đính của hình cho thấy tiến trình biên dịch riêng rẽ mỗi dự án (project) thành một Assembly. Các assembly có khả năng tương tác với nhau nhờ vào chức năng liên thông ngôn ngữ của .NET thông qua CTS (Common Type System) - đặc tả dữ liệu thông dụng, và CLS (Common Language Specification) – đặc tả ngôn ngữ thông dụng.

Phần dưới cho thấy tiến trình biên dịch JIT (Just in time) từ IL trên các assembly thành đoạn mã hiện hành.

Just – in – time (JIT) – biên dịch vừa đúng lúc: Đây là tiến trình thực hiện giai đoạn biên dịch từ IL sang mã máy nguyên sinh. Trình biên dịch JIT chuẩn sẽ chạy theo yêu cầu. JIT compiler khá thông minh để có thể biết được đoạn mã nào đã được biên dịch, nên việc biên dịch chỉ xảy ra khi cần thiết. Do đó khi các ứng dụng .NET chạy thì chúng chạy ngày càng nhanh.

1.1.4. Kiến trúc của .NET Application

Kiến trúc chung của các ứng dụng được phát triển trong môi trường .NET như sau:



Hình 1.3: Kiến trúc của .NET Application

Các tầng dưới cùng là phần cứng, hệ điều hành và Windows API (đây là phần mềm hệ thống cung cấp tất cả các chức năng và các tài nguyên mà các lập trình viên có thể rút ra từ đó để tạo nên các tính năng giao tiếp người – máy, như trình đơn kéo xuống, tên tệp, lệnh bàn phím).

Tiếp theo là trình diễn dịch ngôn ngữ thực thi chung (CLR). Nhờ nó mà một ứng dụng có thể thực thi được cho dù nó được viết trên nhiều ngôn ngữ khác nhau (phần này đã trình bày chi tiết ở phần trên).

Các lớp cơ sở trong .NET Framework cung cấp các hàm chức năng có sẵn, đi từ hiển thị những cửa sổ và biểu mẫu (form), triệu gọi các dịch vụ cơ bản của windows, đọc/viết các tập tin, thâm nhập vào mạng, Internet cũng như truy xuất các nguồn dữ liệu (trên căn cứ dữ liệu chặng hạn).

Trên cùng là giao diện ứng dụng, có thể gồm 2 loại đó là web forms và windows forms.

- Windows forms giống như forms của VB6. Nó hỗ trợ Unicode hoàn toàn, rất tiện cho chữ Việt và thật sự hướng đối tượng.
- Web forms có những server control làm việc giống như các điều khiển trong Windows forms, nhất là có thể dùng mã lệnh để xử lý sự kiện y hệt như của Windows forms.

1.1.5. Các phiên bản của .NET Framework

Bài giảng lập trình trực quan

- .NET framework 1.0 - 2002
- .NET framework 1.1 - 2003
- .NET framework 2.0 - 2005
- .NET framework 3.5 - 2008
- .NET framework 4.0 – 2010

Ver 1.0 – phát hành năm 2002

Ngày 12/2/2002 đánh dấu bước quan trọng đầu tiên trong “cuộc đời” của .NET Framework, khi phiên bản 1.0 cùng với Visual Studio.NET 2002 được chính thức ra mắt. Chính .NET Framework 1.0 là điểm nhấn đáng chú ý nhất và làm cho Visual Studio. NET 2002 khác biệt hẳn với Visual Studio 6.0 đã phát hành năm 1998. Lần đầu tiên, Microsoft giới thiệu về “lập trình hợp nhất”, với việc lấy .NET Framework làm nền tảng.

Ver 1.1 - phát hành năm 2003

Một năm sau ngày .NET Framework 1.0 ra đời, ngày 24/4/2003, Microsoft đã có ngay bản cập nhật 1.1 ra mắt cùng với Visual Studio.NET 2003. Không có nhiều nâng cấp đáng chú ý trong lần ra mắt này, đáng kể nhất là sự ra đời của .NET Compact Framework, phiên bản thu gọn của .NET Framework cho các thiết bị di động. Điều đáng tiếc là mặc dù có nền tảng rất tốt, cùng với sự hỗ trợ mạnh mẽ từ Microsoft, cho đến nay, .NET Compact Framework vẫn chưa phát triển như “lẽ ra nó phải thế”. Hiện nay số thiết bị di động chạy Windows Mobile/Windows Phone khá khiêm tốn so với các hệ điều hành (HĐH) còn lại. .NET Framework 1.1 cũng mở ra một “truyền thống” là kể từ đây, các HĐH Windows đều được cài đặt sẵn phiên bản .NET Framework mới nhất. Windows Server 2003 tiên phong với phiên bản 1.1, sau đó là Windows Vista với .NET 3.0, và gần đây nhất là Windows 7/Server 2008 với .NET 3.5 SP1.

Ver 2.0 phát hành năm 2005

Microsoft mất đến hơn 2 năm để phát triển .NET Framework 2.0 và Visual Studio 2005, và thời gian bỏ ra là thật sự đáng giá. Tháng 11/2005, hai sản phẩm này ra mắt với hàng loạt tính năng mới, trong đó đáng kể nhất là việc hỗ trợ hoàn toàn cho tính toán 64-bit, .NET Micro Framework, bổ sung và nâng cấp nhiều control của ASP.NET và đặc biệt là hỗ trợ Generics. .NET 2.0 hoàn toàn khác biệt so với các phiên bản trước. Generic cho phép chúng ta định kiểu an toàn (type safety). Chúng cho phép ta tạo ra một cấu trúc dữ liệu mà không cần phải xác định đó là kiểu dữ liệu gì. Tuy nhiên khi cấu trúc dữ liệu này được sử

Bài giảng lập trình trực quan

dụng, trình biên dịch phải đảm bảo rằng kiểu dữ liệu được sử dụng với nó là kiểu an toàn. Generic cũng tương đương với Template trong C tuy nhiên việc sử dụng Generic trong .NET dễ dàng hơn nhiều so với Template. Phiên bản 1.0 và 1.1 của .NET Framework không hỗ trợ generics. Thay vào đó, lập trình viên sử dụng lớp Object với các tham số và thành viên sẽ phải chuyển đổi tới các lớp khác dựa trên lớp Object. Generics mang đến hai tính năng cải tiến đáng kể đối với việc sử dụng lớp Object: Giảm bớt lỗi vận hành (Reduced run-time errors), Hiệu suất được cải thiện (Improved performance).

Ver 3.0 & Ver 3.5 (phát hành năm 2008)

Nếu như 3 phiên bản trước đó, .NET Framework đều gắn liền với một phiên bản Visual Studio nào đó, thì .NET Framework 3.0 đã “phá” truyền thống này khi ra mắt cùng với hệ điều hành Windows Vista vào cuối năm 2006. Ba “điểm nhấn” trong lần nâng cấp này là thành phần được kỳ vọng thay thế Winform - Windows Presentation Foundation – WPF, Windows Communication Foundation – WCF, Windows Workflow Foundation - WF, và Windows Card Space. .NET Framework 3.0 không phải là một phiên bản mới hoàn toàn, thực tế là một bản nâng cấp của .NET 2.0, hay đúng hơn là một bản nâng cấp cho thư viện của .NET 2.0. Chính vì không có Visual Studio “đi kèm”, mà .NET 3.0 đành phải “ký gửi” vào Visual Studio 2005 với một bộ công cụ mở rộng. Người dùng phải đợi đến tháng 11 năm 2007 mới được sử dụng một phiên bản Visual Studio hỗ trợ đầy đủ và toàn diện cho .NET 3.0, và hơn thế nữa. Vâng, chúng ta đang nói đến VS 2008 và .NET Framework 3.5. Cũng như phiên bản 3.0, .NET 3.5, là một mở rộng trên nền .NET 2.0. LINQ [LINQ: Language Integrated Query - đây là thư viện mở rộng cho các ngôn ngữ lập trình C# và Visual Basic .NET (có thể mở rộng cho các ngôn ngữ khác) cung cấp khả năng truy vấn trực tiếp dữ liệu đối tượng, CSDL và XML] là phần nổi bật và đáng chú ý nhất trong .NET 3.5.

Ver 4.0 – phát hành năm 2010

Ngày 12/4 vừa qua, Microsoft lại nâng cấp .NET Framework và Visual Studio. Đây là phiên bản đầu tiên sau .NET 2.0 kể từ 2005, có một CLR hoàn toàn mới: CLR 4.0. Cũng cần nhắc lại là cả .NET 3.0 và 3.5 đều sử dụng CLR 2.0, và không có CLR 3.0. Việc Microsoft chuyển thẳng lên 4.0 không chỉ để “đồng bộ” phiên bản, mà còn nhằm khẳng định đây là một bước tiến lớn. .NET Framework 4 giới thiệu một model an ninh được cải thiện. Các tính năng mới và cải tiến trong .NET Framework 4 là:

Bài giảng lập trình trực quan

- + Application Compatibility and Deployment (Khả năng tương thích ứng dụng và triển khai)
- + Core New Features and Improvements (Các tính năng mới và cải tiến của phần nhân)
- + Managed Extensibility Framework (Quản lý mở rộng Framework)
- + Parallel Computing (Điện toán song song)
- + Networking
- + Web
- + Client
- + Data
- + Windows Communication Foundation (WCF)
- + Windows Workflow Foundation (WF)

1.2. Các công cụ lập trình và cài đặt

1.2.1. Các công cụ lập trình

Để có thể phát triển các ứng dụng .NET một cách nhanh chóng ta có thể cài đặt các công cụ lập trình sau:

- + Visual studio 2005 – hỗ trợ đến .NET Framework 2.0
- + Visual studio 2008 – hỗ trợ đến .NET Framework 3.0 và 3.5
- + Visual studio 2010 – hỗ trợ đến .NET Framework 4.0

Ở tài liệu này tôi sẽ hướng dẫn các bạn cài đặt Visual Studio 2010

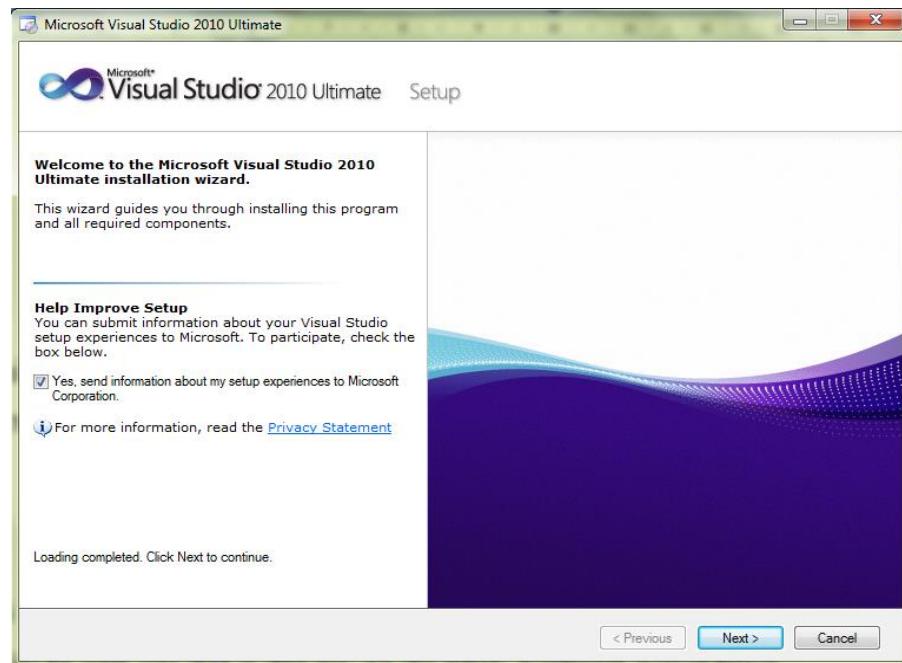
1.2.2. Cài đặt Visual studio 2010

Để cài đặt được Visual Studio 2010 ta cần phải có đĩa DVD Visual Studio 2010 khoảng 2.3 GB. Ta có thể cài trên Windows XP hoặc Window7 khoảng 30GB đĩa cứng, RAM 512MB trở lên.

1. Kích đúp chuột vào File autorun trên đĩa cài hoặc trong thư mục chứa trình cài đặt VS 2010, xuất hiện cửa sổ cài đặt, kích chuột trái chọn mục *Install Microsoft Visual Studio 2010*

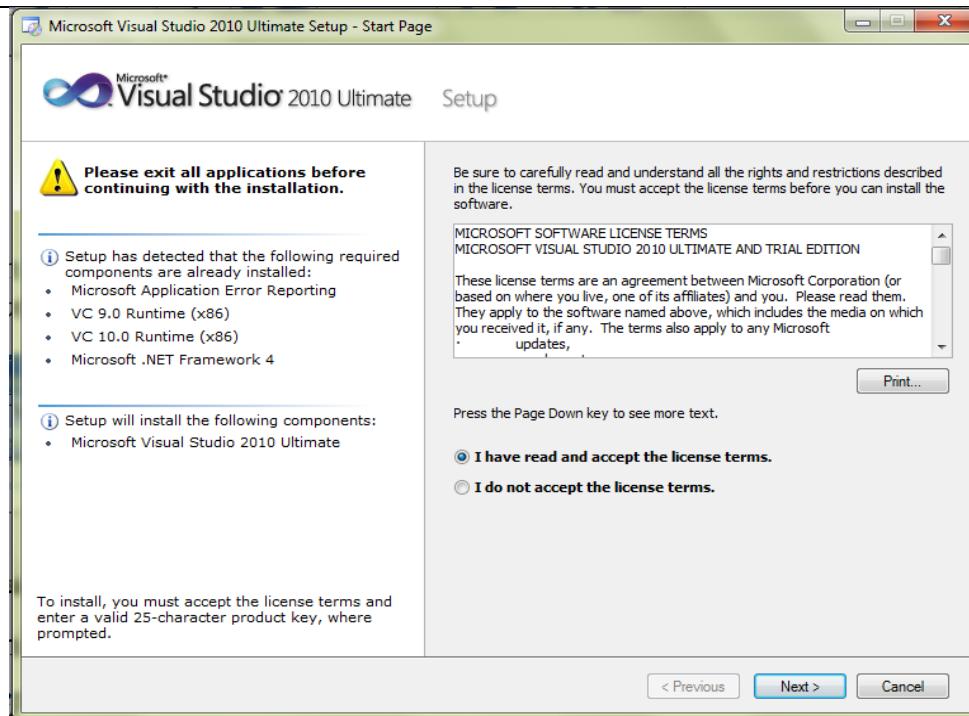


2. Xuất hiện cửa sổ chào mừng đến với chương trình cài đặt Visual studio 2010 chọn nút Next.

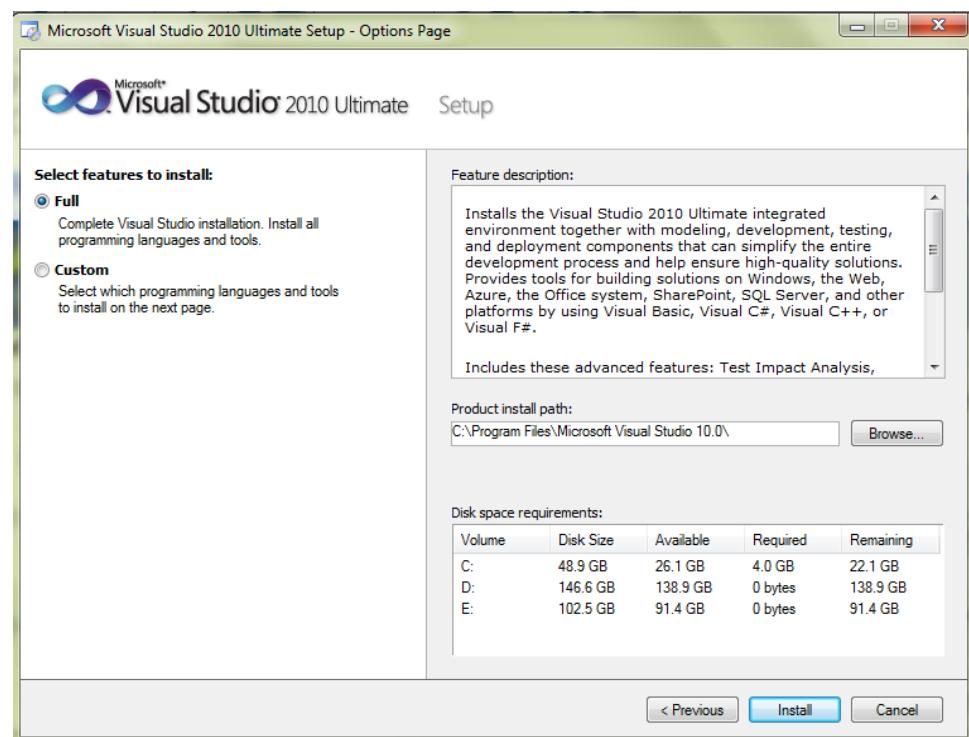


3. Màn hình tiếp theo là những thông tin về bản quyền, cần phải đọc và kích chọn mục *I have read and accept the license terms*, rồi chọn nút Next.

Bài giảng lập trình trực quan

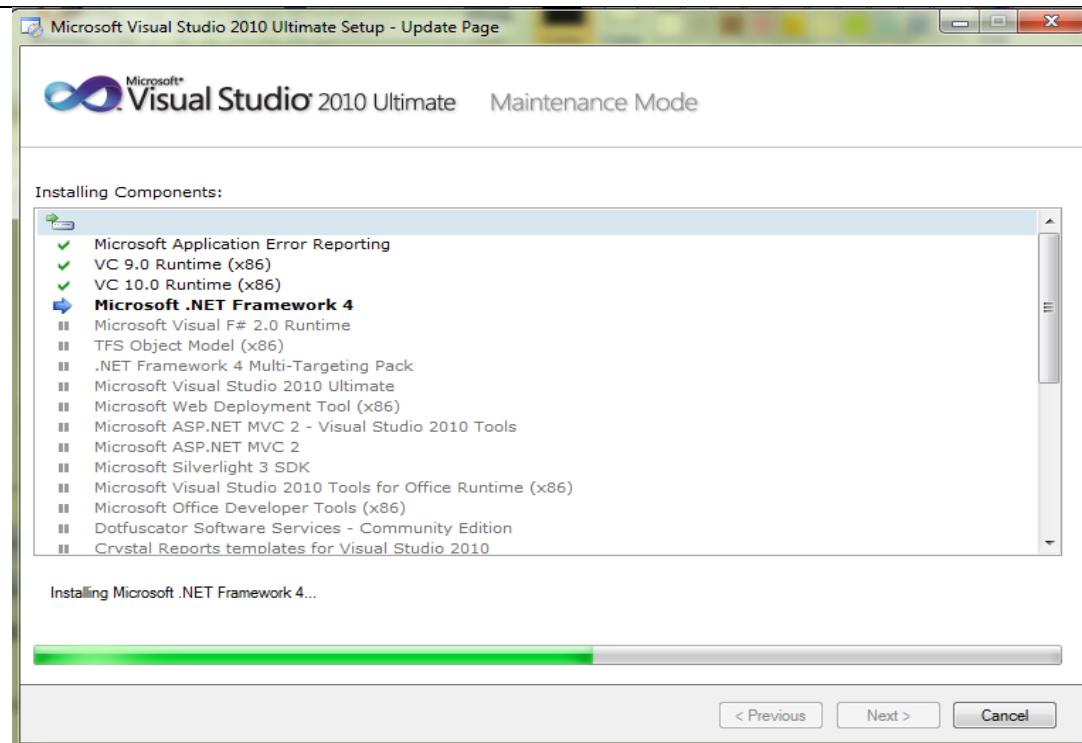


- Xuất hiện màn hình yêu cầu chọn đường dẫn sẽ cài Visual Studio 2010 và chế độ cài đặt. Chọn Full – cài tất cả các thành phần trong Visual studio 2010, Custom – tự chọn các thành phần cần cài đặt. Nhấn Install để tiếp tục.

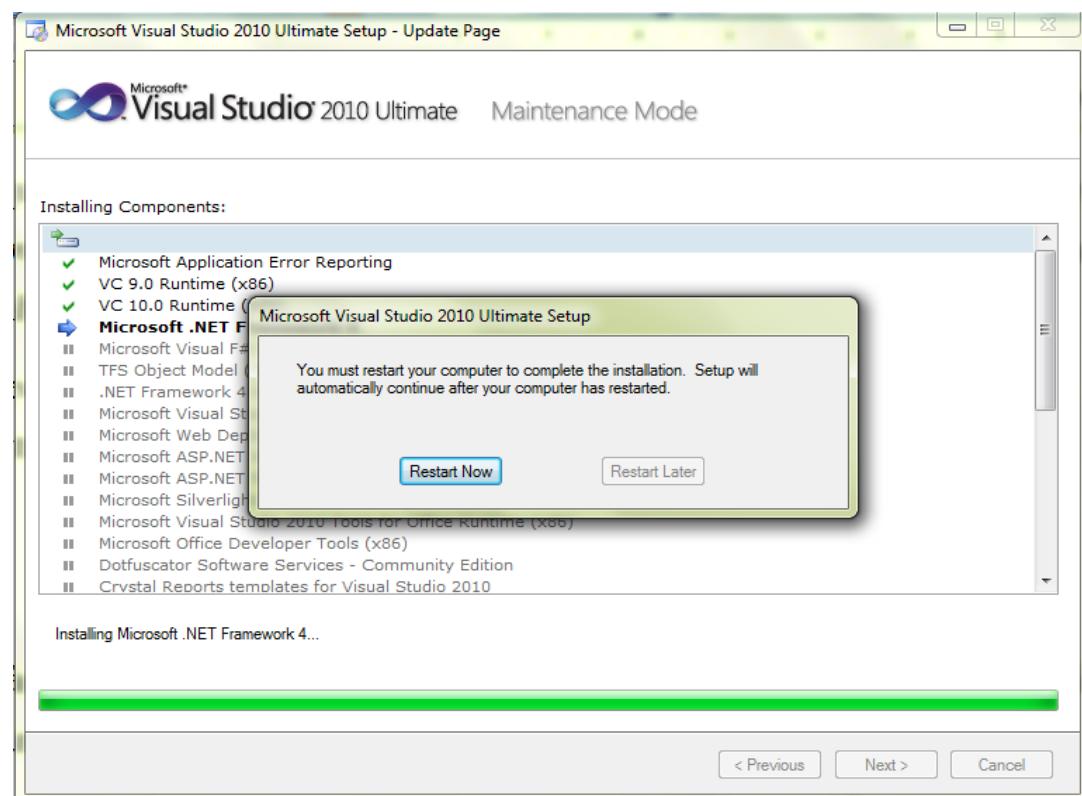


- Quá trình cài đặt bắt đầu được tiến hành.

Bài giảng lập trình trực quan

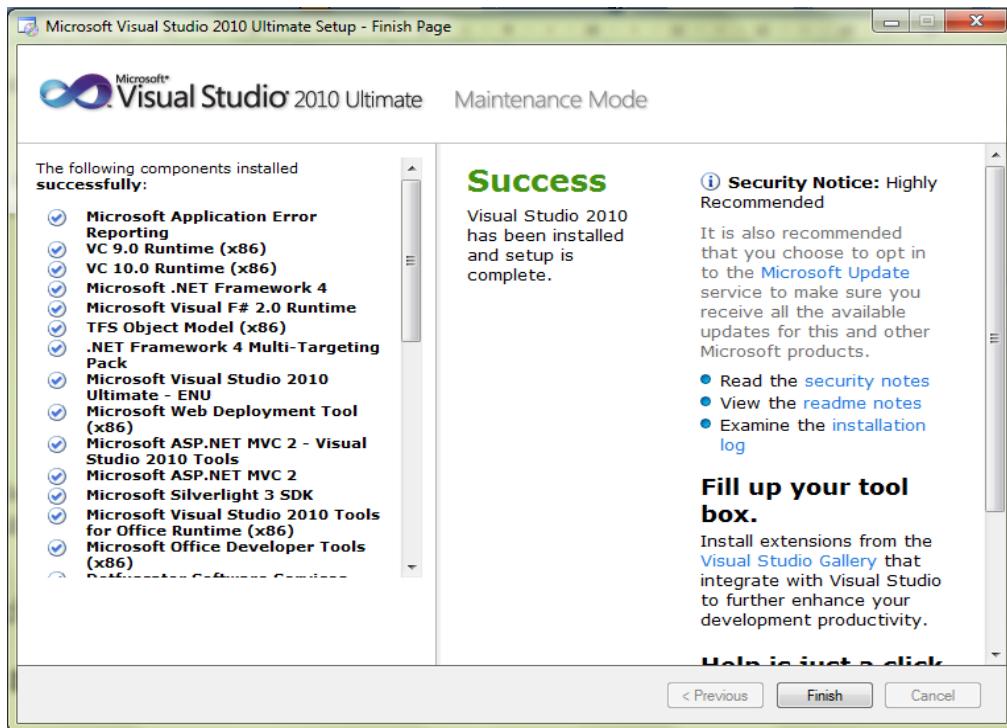


- Trong quá trình cài đặt có thể xuất hiện cửa sổ yêu cầu Restart lại máy, bạn chọn *Restart now*



Bài giảng lập trình trực quan

7. Sau khi khởi động lại máy, cửa sổ cài đặt lại xuất hiện trở lại sau một vài phút. Khi tất cả các thành phần đã xong bạn hãy chọn Next để tiếp tục quá trình cài đặt.
8. Tiếp theo màn hình xuất hiện cửa sổ Finish, thông báo bạn đã cài đặt thành công Visual studio 2010. Chọn nút Finish để kết thúc quá trình cài đặt.



Chú ý:

+ Ở bài giảng này tôi cung cấp cho các bạn sẵn file cài đặt là file .iso. Để chạy được file .iso này ta cần cài đặt ổ đĩa ảo, ví dụ như: VirtualCloneDrive

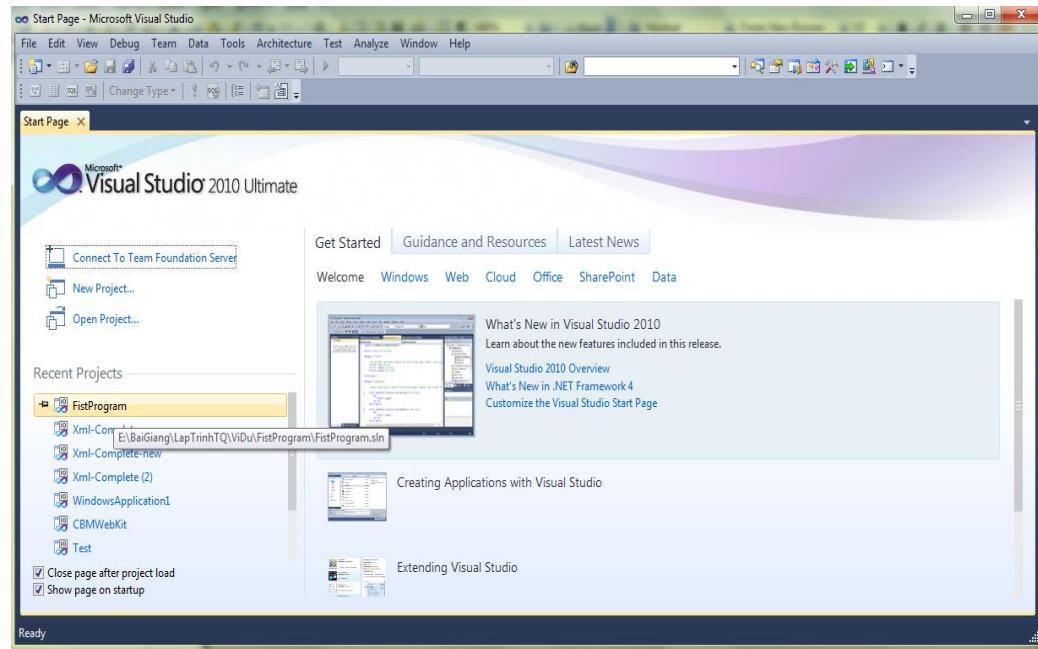
+ Sau khi cài xong Visual studio 2010 ta có bản dùng thử, để crack ta làm theo hướng dẫn sau:

1. Sau khi cài xong, vào Control Panel
2. Vào Add or Remove Programs
3. Tìm Microsoft Visual Studio 2010 Ultimate ENU
4. Chọn Change or Remove hoặc click đúp vào đó
5. Đợi chạy xong thanh chờ chọn Next
6. Sau đó bạn sẽ thấy chỗ nhập key, nhập key YCFHQ-9DWCY-DKV88-T2TMH-G7BHP vào rồi chọn Active.

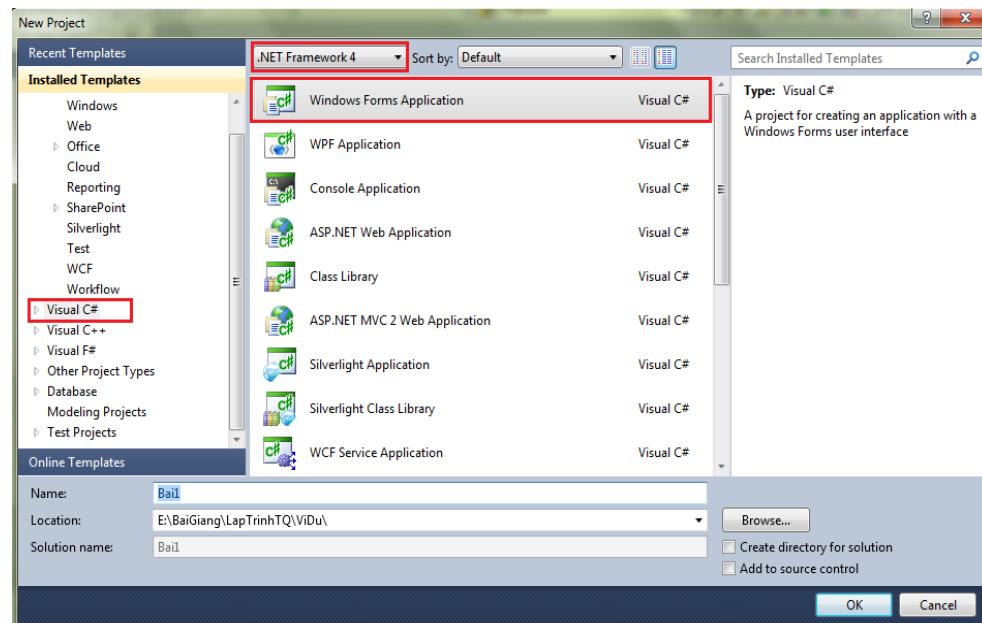
1.3. Khởi động Visual studio 2010 và giao diện

1.3.1. Khởi động Visual studio 2010

Vào Start\All Programs\Microsoft Visual Studio 2010\Microsoft Visual Studio 2010, xuất hiện cửa sổ Start Page của sổ có hiển thị danh sách các dự án được mở gần đây nhất như sau:



Để mở một dự án mới ta vào File/New/Project... hoặc nhấn Ctrl + Shift + N, xuất hiện cửa sổ New Project như sau:



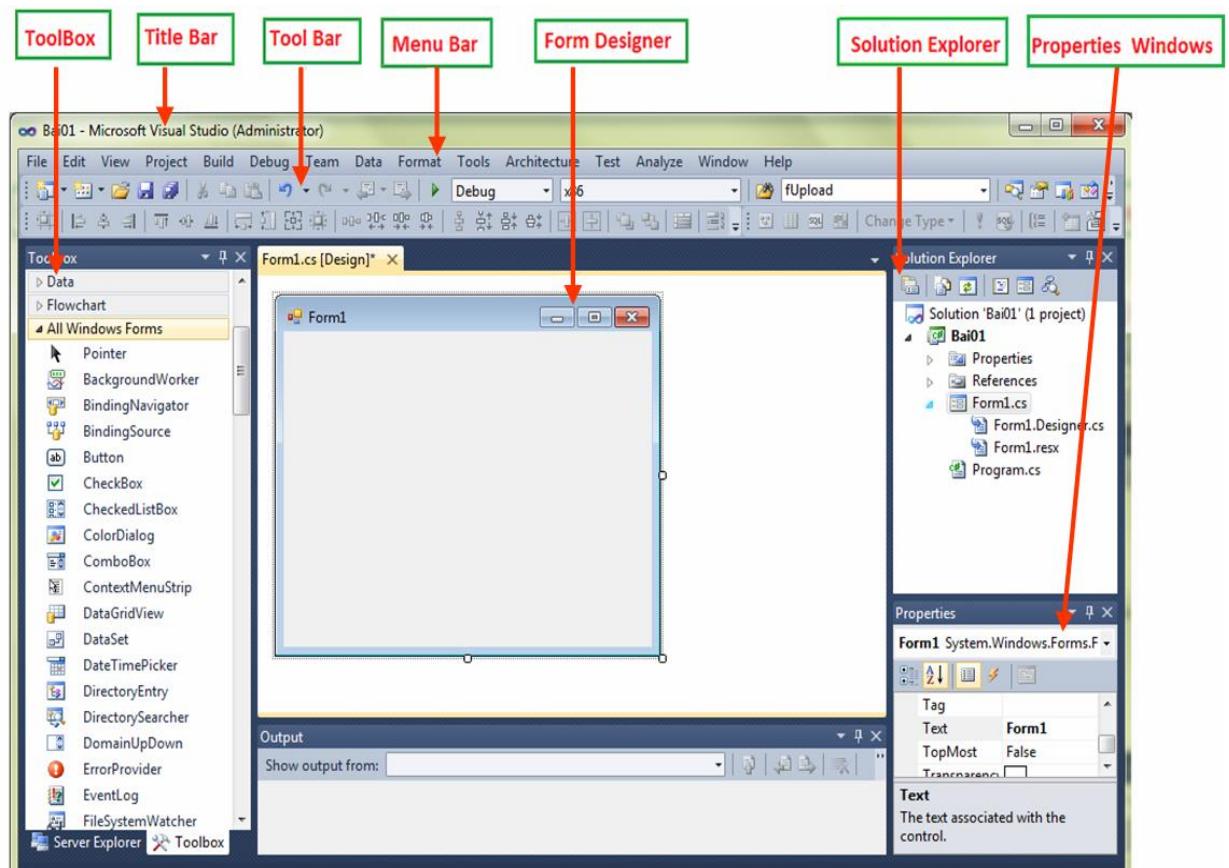
Bài giảng lập trình trực quan

Ta chọn ngôn ngữ lập trình là Visual C#, chọn loại ứng dụng là Windows Forms Application, chọn phiên bản mới nhất là .NET Framework 4.0. Tại ô Name ta viết tên của dự án, tại ô Location ta chọn thư mục chứa dự án bằng cách nhấn nút Browse...

Cuối cùng ta nhấn OK.

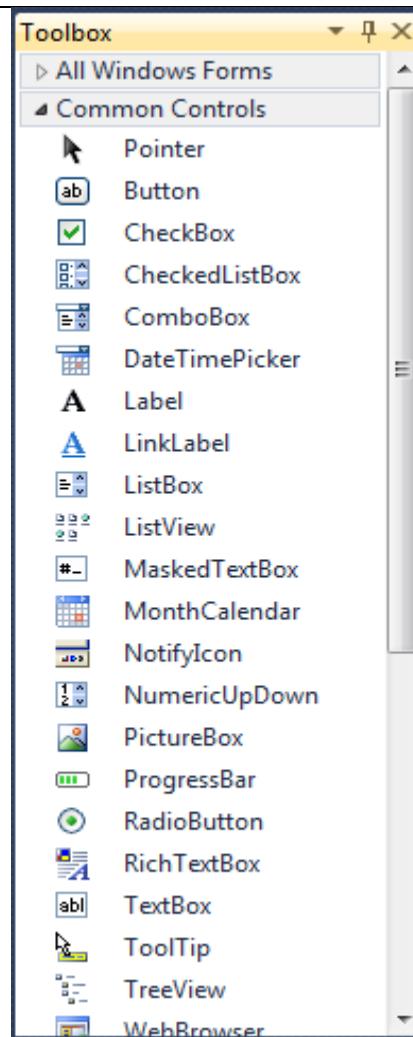
1.3.2. Giao diện môi trường lập trình Visual C# trên WinForm

Sau khi khởi động Visual Basic trong Visual studio 2010 theo các bước như trên xuất hiện cửa sổ môi trường phát triển IDE, có giao diện gồm các thành phần như sau:



❖ **ToolBox:** là hộp công cụ chứa các điều khiển – controls được đặt lên form khi thiết kế giao diện người dùng. Để hiển thị hộp công cụ ta thực hiện một trong các cách sau:

- + Vào View/Toolbox
- + Bấm tổ hợp phím Ctrl + Alt + X
- + Bấm vào biểu tượng Toolbox trên thanh công cụ standard



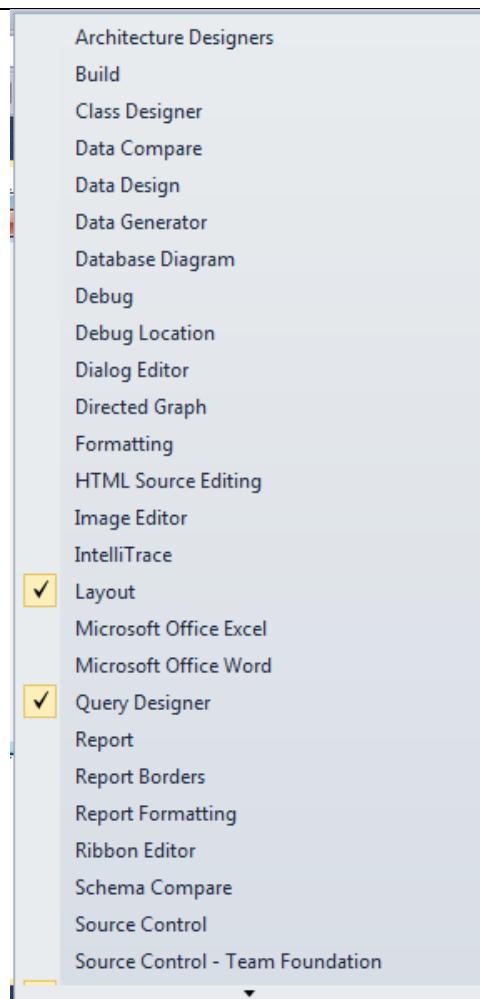
- ❖ **Title Bar:** Thanh tiêu đề chứa tên dự án
- ❖ **Menu Bar:** Thanh menu chứa đầy đủ các công cụ cần để phát triển, thực thi và cài đặt ứng dụng...
 - + File: Cho phép mở, thêm mới và lưu trữ dự án...
 - + Edit: gồm các thao tác hỗ trợ soạn thảo mã lệnh như: Copy, cắt, dán, tìm kiếm...
 - + View: cho phép hiển thị các công cụ hỗ trợ người dùng trong quá trình xây dựng dự án như: cửa sổ mã lệnh – code, form thiết kế - designer, hộp công cụ - toolbox, thanh công cụ - toolbar, cửa sổ thuộc tính – Properties...
 - + Project: cho phép bổ sung các đối tượng khác nhau vào dự án như: Form, các Component, Các class...
 - + Built: cho phép biên dịch dự án

Bài giảng lập trình trực quan

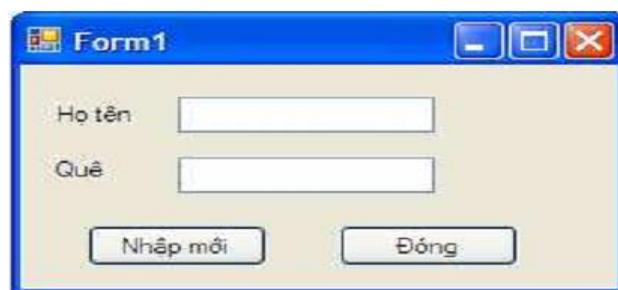
- + Debug: cho phép chạy và gỡ lỗi chương trình
 - + Data: Cho phép thêm mới và hiển thị cơ sở dữ liệu của dự án.
 - + Format: Cho phép căn lề, định dạng kích thước, chế độ hiển thị... của các điều kiển được đặt trên Form.
 - + Tools: Cung cấp các công cụ cho phép kết nối tới các thiết bị ngoại vi như Pocket PC, SmartPhone... hoặc kết nối tới các hệ quản trị CSDL cũng như kết nối tới máy chủ Server...
 - + Window: kiểm soát cách bố trí cửa sổ
 - + Help: Truy cập hệ trợ giúp trực tuyến MSDN
- ❖ **Tool Bar:** Thanh công cụ gồm một tập hợp các nút lệnh, mỗi nút lệnh chứa các biểu tượng icons và các chức năng tương ứng với chức năng của một mục lựa chọn trong thanh Menu. Thanh công cụ rất hữu ích và trực quan, giúp người dùng dễ dàng và nhanh chóng thực hiện một số chức năng mong muốn chỉ thông qua cái Click chuột.



Để gọi các thanh công cụ ra ta vào View/Toolbars hoặc kích chuột phải tại thanh Menu, Khi đó sẽ xuất hiện danh sách tất cả các thanh công cụ. Muốn ẩn/hiện thanh nào ta kích chuột tại dòng chứa tên thanh công cụ đó.



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

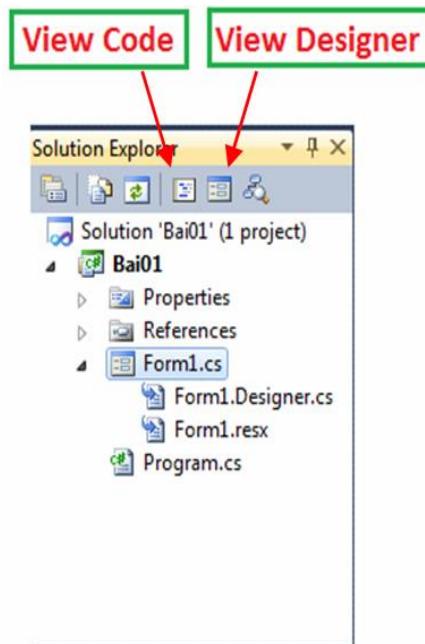


- ❖ **Solution Explorer:** Cửa sổ giải pháp – đây là phần cửa sổ giúp ta quả lý tất cả các tài nguyên và tệp tin của dự án. Solution Explorer được tổ chức thành một cấu trúc cây bao gồm các mục khác nhau như: danh sách các Form, danh sách các hình ảnh, danh sách các class, ...

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

Bài giảng lập trình trực quan

- + Vào View\Solution Explorer
- + Bấm tổ hợp phím Ctrl+Alt+L hoặc Ctrl+R
- + Kích chuột vào biểu tượng Solution Explorer  trên thanh công cụ Standard.



Trong cửa sổ Solution Explorer có 2 thành phần quan trọng chúng ta rất hay dùng là *View Code* và *View Designer*.

View Code: Có tác dụng hiển thị cửa sổ soạn thảo mã lệnh cho Form đang được chọn. Ngoài ra, để hiển thị cửa sổ soạn thảo mã lệnh ta có thể làm một trong các cách sau:

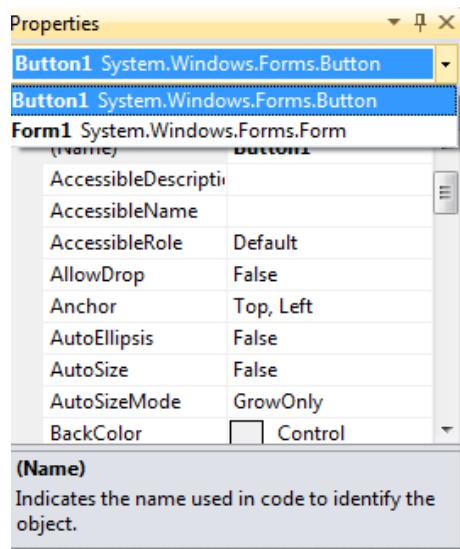
- + Vào View/Code
- + Bấm phím F7
- + Kích đúp chuột tại cửa sổ thiết kế của Form.

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

- + Vào View\Designer
- + Bấm phím Shift + F7

Bài giảng lập trình trực quan

- ❖ **Properties Window:** Cửa sổ này liệt kê tất cả các thuộc tính của form và các điều khiển trong dự án. Muốn hiển thị thuộc tính của đối tượng nào ta kích chuột chọn đối tượng ấy trong cửa sổ thiết kế giao diện, hoặc chọn tên đối tượng trong danh sách thả xuống ở phần đầu của cửa sổ Properties.



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

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

+ Vào View\Properties Window

+ Kích chuột vào biểu tượng Properties Window trên thanh công cụ Standard, hoặc biểu tượng Properties trong cửa sổ giải pháp Solution Explorer

+ Bấm phím F4

+ Bấm tổ hợp phím Alt + Enter

CHƯƠNG 2: VIẾT CHƯƠNG TRÌNH ĐẦU TIÊN

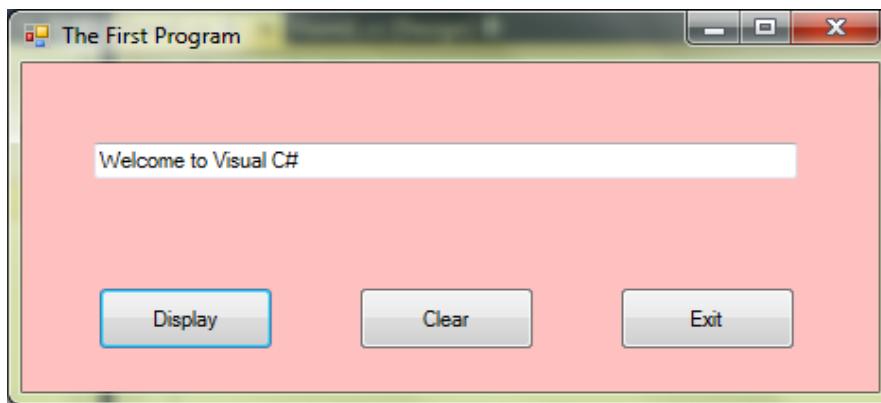
Một chương trình ứng dụng windows được xây dựng theo 2 bước sau:

- + Bước 1: Thiết kế giao diện
- + Bước 2: Viết mã lệnh cho chương trình

2.1. Đề bài

Viết chương trình gồm một hộp văn bản TextBox và 3 nút lệnh Button: *Display*, *Clear*, *Exit* với các yêu cầu sau:

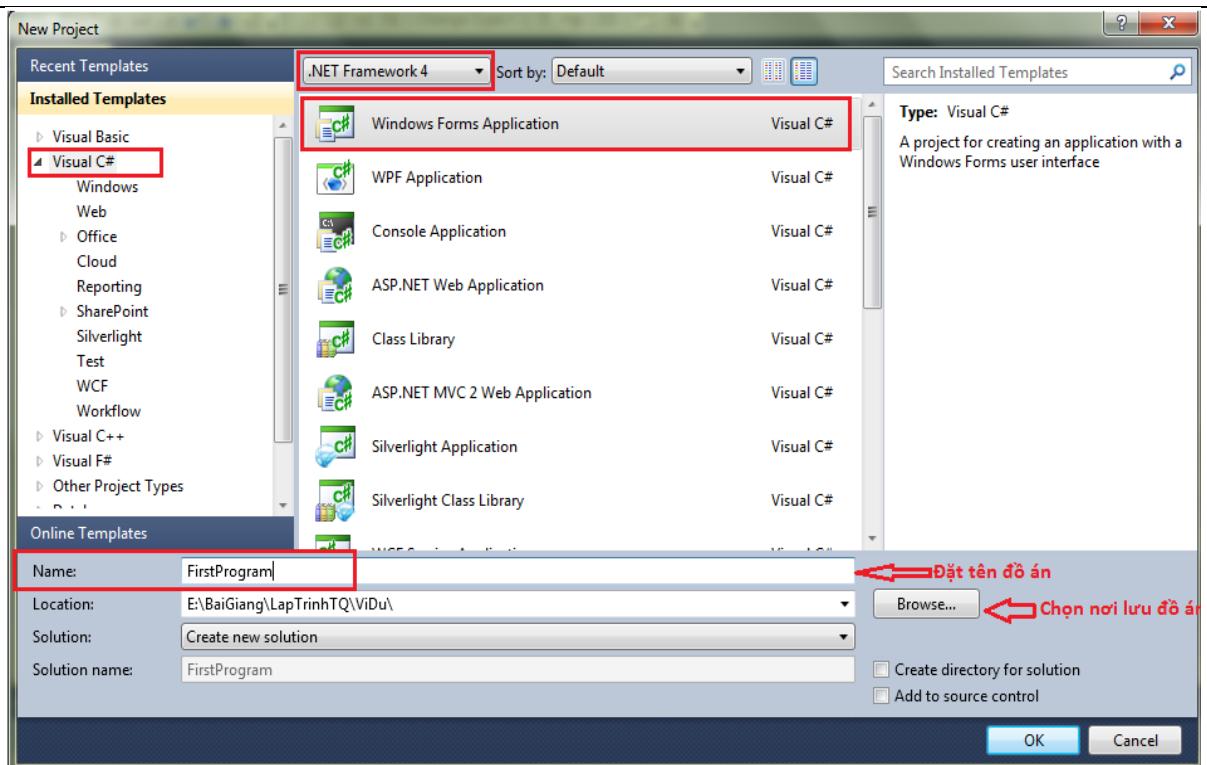
- + Kích chuột vào nút *Display* thì trong hộp văn bản xuất hiện dòng chữ “Welcome to Visual Basic .NET”
- + Kích chuột vào nút *Clear* chữ ở hộp văn bản biến mất.
- + Kích chuột vào nút *Exit* thì thoát khỏi chương trình quay lại cửa sổ soạn thảo.



2.2. Mở dự án

Mở **Visual studio 2010**, chọn **File/New/Project...** để khởi động một dự án mới. Xuất hiện hộp thoại **New Project** như sau:

Bài giảng lập trình trực quan



Chọn ngôn ngữ là **Visual C#**, ứng dụng là **Windows Forms Application**, phiên bản **.NET Framework 4.0**. Đặt tên dự án tại mục **Name** là **FistProgram**, chọn nơi lưu dự án bằng nút **Browse**, Sau đó nhấn **OK**.

2.3. Thiết kế giao diện

2.3.1. Đặt tên và tiêu đề cho form

Kích chuột vào vị trí bất kỳ trong Form, trong cửa sổ Properties sửa thuộc tính Name thành *frmWelcome*, sửa thuộc tính Text là *The First Program*.

2.3.2. Thêm điều khiển hộp văn bản TextBox

Kích chuột vào biểu tượng  **TextBox** trên hộp công cụ Toolbox, bấm phím trái cùng với giữ và kéo chuột để đặt hộp văn bản trên Form.

Khi mới xuất hiện hộp văn bản trên Form thì hộp văn bản tên mặc định là **TextBox1**, Kích chọn hộp văn bản trong cửa sổ Properties sửa thuộc tính *Name* sửa thành *txtWelcome*.

2.3.3. Thêm điều khiển nút lệnh Button

Kích chuột vào biểu tượng  **Button** trên hộp công cụ ToolBox, bấm phím trái cùng với giữ và kéo chuột để đặt nút lệnh trên Form. Nút lệnh này có tên mặc định là **Button1** và nội dung cũng là **Button1**.

Bài giảng lập trình trực quan

Thực hiện tương tự ta đưa thêm 2 nút lệnh Button2 và Button3. Sau đó ta tiến hành thay đổi 2 thuộc tính Name và Text của các nút lệnh như sau:

Nút lệnh	Name	Text
Button1	btnDisplay	Display
Button2	btnClear	Clear
Button3	btnExit	Exit

Chú ý: Mọi điều khiển đều có thuộc tính Name, để dễ quản lý, gõ rồi và viết chương trình ta nên đặt tên điều khiển tương ứng với chức năng của nó và có tiếp đầu ngữ chỉ loại điều khiển ở đầu.

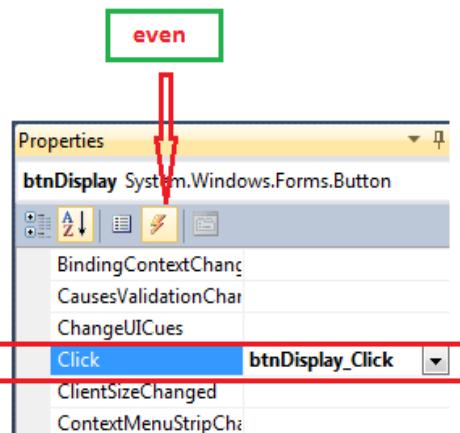
Ví dụ: TextBox có tiếp đầu ngữ - txt, Button – btn, Form – frm, ... Các tiếp đầu ngữ viết thường, tên điều khiển viết hoa chữ cái đầu tiên của mỗi từ, ví dụ: **txtWelcome**, **btnDisplay**...

2.4. Viết mã lệnh

2.4.1. Viết mã lệnh cho nút Display

Ta mở cửa sổ soạn thảo code bằng một trong hai cách sau:

- + C1: Kích đúp chuột vào nút Display
- + C2: Kích chọn đồi tượng (nút nhấn) btnDisplay. Trong cửa sổ **Properties** chọn biểu tượng even, sau đó kích đúp vào sự kiện Click như hình vẽ sau:



Bài giảng lập trình trực quan

Trong cửa sổ code xuất hiện một hàm để xử lý sự kiện khi người dùng nhấn vào nút **Display** như sau:

```
private void btnDisplay_Click(object sender, EventArgs e)
{
    |
}
```

Tiếp theo, ta gõ vào giữa hàm btnDisplay_Click ... dòng lệnh gán giá trị “Welcome to Visual C#”. Ta cho thuộc tính Text của điều khiển txtWelcome như sau:

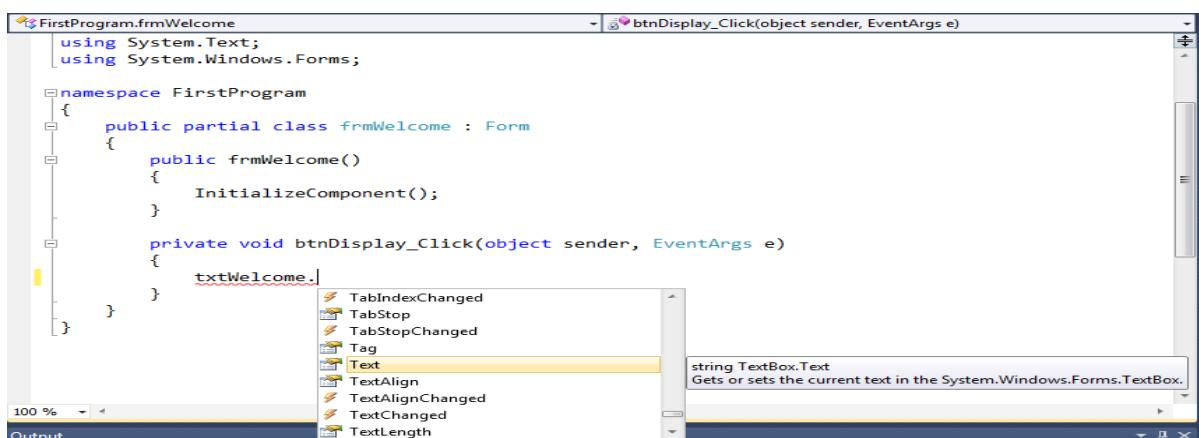
```
private void btnDisplay_Click(object sender, EventArgs e)
{
    txtWelcome.Text = "Welcome to Visual C#";
}
```

Căn dặn:

1. Hầu như tất cả các hàm xử lý một sự kiện nào đó của các điều khiển trên Form đều có 2 đối số. Đối số *Sender* có kiểu *Object* đại diện cho đối tượng đã phát sinh sự kiện, đối số *e* có kiểu *EventArgs* chứa các thông tin về sự kiện như: vị trí chuột, thời gian phát sinh sự kiện, thời điểm phát sinh sự kiện, ... Từ khóa *Handles* nhằm xác định rõ thủ tục đang xử lý dùng cho sự kiện nào của đối tượng.
2. Ta có cấu trúc chung để gán giá trị cho thuộc tính của một điều khiển khi viết mã lệnh như sau:

<Tên điều khiển>.<Thuộc tính>=<Giá trị>

Các thuộc tính của các điều khiển trong **Visual C#** rất phong phú, trong Visual studio cung cấp tiện ích **Intellisense** tự động hiển thị danh sách các thuộc tính của điều khiển sau khi gõ tên điều khiển và dấu chấm ‘.’, ví dụ như sau:



Bài giảng lập trình trực quan

Để lựa chọn thuộc tính ta dùng phím mũi tên lén xuống, hoặc gõ chữ cái đầu của thuộc tính. Sau đó ta dùng phím Enter để chèn tự động tên thuộc tính vào dòng lệnh.

3. Trong môi trường soạn thảo, nếu bạn gõ sai cú pháp thì môi trường lập trình sẽ bắt lỗi ngay bằng cách hiển thị đường gạch chân hình răng cưa ngay dưới chỗ sai, khi sửa xong lập tức đường gạch hình răng cưa bị mất.

2.4.2. Viết mã lệnh nút Clear

```
private void btnClear_Click(object sender, EventArgs e)
{
    txtWelcome.Text = "";
}
```

2.4.3. Viết mã lệnh nút Exit

```
private void btnExit_Click(object sender, EventArgs e)
{
    this.Close();
}
```

Lệnh `this.Close()` có tác dụng đóng Form hiện hành lại và xóa tất cả các đối tượng của Form ra khỏi bộ nhớ.

2.5. Chạy chương trình

Để chạy chương trình, ta thực hiện một trong các cách sau:

- + C1: Chọn Debug/Start Debugging
- + C2: Kích chuột trái vào biểu tượng Start Debugging  trên thanh công cụ Standard
- + C3: Nhấn phím F5

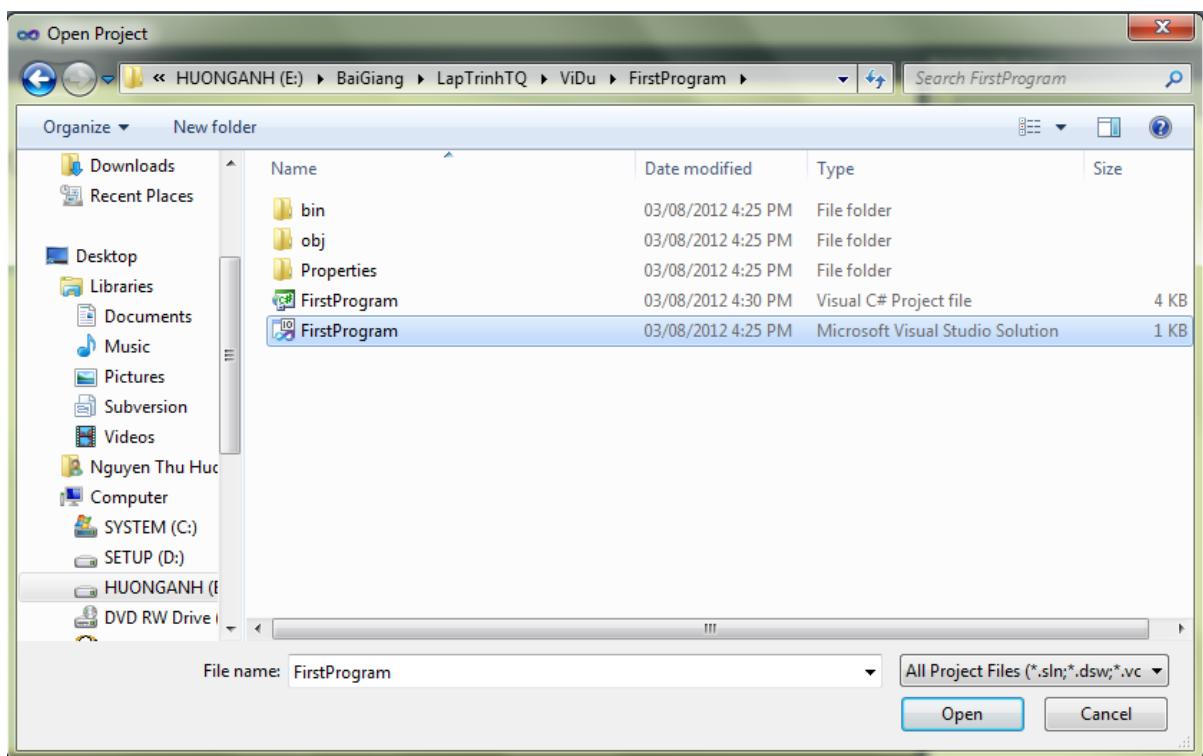
2.6. Dừng chương trình

Khi chạy chương trình có thể xảy ra lỗi, khi xảy ra lỗi thì bạn cần dừng chương trình để quay lại môi trường soạn thảo code để sửa bằng một trong các cách sau:

- + C1: Kích chuột vào biểu tượng Stop  trên thanh công cụ.
- + C2: Vào Debug\Stop Debugging
- + C3: Nhấn tổ hợp phím Shift + F5

2.7. Mở dự án đã có

Vào File/Open/Project.... Xuất hiện hộp thoại Open Project, ta chọn đường dẫn đến thư mục chứa dự án. Ta chọn file có tên là tên dự án có kiểu là **Microsoft Visual studio Solution** hoặc **Visual C# Project File**, sau đó nhấn OK.



CHƯƠNG 3: CƠ BẢN VỀ NGÔN NGỮ C#

3.1. Đặc điểm của C#

C# có những đặc điểm sau:

- ❖ Trong C# mọi thứ đều là lập trình hướng đối tượng.
- ❖ Biên dịch.
- ❖ Phân biệt chữ hoa, chữ thường.
- ❖ Mỗi lệnh kết thúc bởi dấu chấm phẩy (;
- ❖ C# cũng hỗ trợ 2 kiểu chú thích như trong C++

3.2. Biến, hằng và các kiểu dữ liệu

3.2.1. Biến

Là một giá trị dùng để chứa dữ liệu tạm thời trong quá trình tính toán. Ta có thể khai báo biến theo cú pháp sau:

[Phạm vi truy xuất] <Kiểu biến> <tên biến> [=<giá trị>]

+ **Tên biến:** Có thể dài 256 ký tự bao gồm các chữ cái, chữ số và dấu gạch dưới. Tên biến phải bắt đầu bằng một chữ cái. C# phân biệt chữ hoa chữ thường.

Phạm vi truy xuất gồm có:

+ **Public:** Những biến khai báo public có thể được dùng ở bất kỳ chỗ nào trong lớp đó và cả các lớp khác trong dự án.

+ **private:** Những biến khai báo private thì chỉ được truy cập ở trong lớp đó.

+ **protected:** Những biến khai báo protected chỉ được truy cập bên trong lớp chứa nó và những lớp dẫn xuất từ lớp đó.

3.2.2 Hằng

Dùng để chứa những dữ liệu có giá trị không đổi trong suốt quá trình tính toán. Sử dụng hằng số làm chương trình sáng sủa, dễ đọc nhờ tên gọi gợi nhớ thay vì các con số. Hằng được khai báo theo cú pháp sau:

<const> <kiểu dữ liệu> <tên hằng> = <giá trị>;

Ví dụ: const double Pi=3.14

Bài giảng lập trình trực quan

Phạm vi hoạt động: Hàng được khai báo với từ khóa Private trong phương thức chỉ hoạt động trong phương thức, khai báo cho biểu mẫu chỉ hoạt động trong biểu mẫu đó. Hàng được khai báo với từ khóa Public hoạt động trên toàn ứng dụng.

3.2.3. Các kiểu dữ liệu

1. Kiểu số

- Kiểu số nguyên gồm có:

Kiểu dữ liệu	Số byte	Mô tả
byte	1	Số nguyên dương không dấu từ 0 – 255
sbyte	1	Số nguyên có dấu (từ -128 đến 127)
short	2	Số nguyên có dấu giá trị từ -32768 đến 32767
ushort	2	Số nguyên không dấu từ 0 – 65535
int	4	Số nguyên có dấu từ -2.147.483.647 đến 2.147.483.647
uint	4	Số nguyên không dấu 0 – 4.294.967.295
long	8	Kiểu số nguyên có dấu có giá trị trong khoảng -9.223.370.036.854.775.808 đến 9.223.372.036.854.775.807
ulong	8	Số nguyên không dấu từ 0 đến 0xffffffffffffffff

- Kiểu số thực gồm có:

Kiểu dữ liệu	Số byte	Mô tả
float	4	Kiểu dấu chấm động có giá trị xấp xỉ từ 3,4E-38 đến 3,4E+38 với 7 chữ số có nghĩa
double	8	Kiểu dấu chấm động có độ chính xác gấp đôi, giá trị xấp xỉ từ 1,7E-308 đến 1,7E+308 với 15, 16 chữ số có nghĩa
decimal	8	Có độ chính xác đến 28 con số và giá trị thập phân.

Bài giảng lập trình trực quan

Các phép toán số học

+ Cộng (+), trừ (-), nhân (*), chia (/). Các phép toán $x[+,-,*,/]=y$ tương đương với phép toán $x=x[+,-,*,/]y$.

+ Phép chia lấy phần dư (%). Ví dụ: $5\%3=2$

+ Math.Round(x,n): hàm làm tròn số thực x đến n chữ số sau dấu phẩy thập phân, ví dụ: $\text{Math.Round}(11.346,2) = 11.35$

+ Math.Sin(x): hàm tính giá trị của $\sin(x)$

+ Math.Cos(x): hàm tính giá trị của $\cos(x)$

+ Math.Exp(x): hàm tính giá trị của e^x

+ Math.Pow(x,y): hàm tính giá trị x^y

+ Math.Abs(x): hàm tính giá trị tuyệt đối của x

+ Math.Sqrt(x): hàm tính căn bậc hai của x

+ Sử dụng lớp Random để sinh số ngẫu nhiên:

- B1: Khởi tạo đối tượng Random.

```
Random rd=new Random();
```

- B2: Dùng các hàm với ý nghĩa như sau:

* rd.Next(): Sinh ngẫu nhiên một số kiểu int

* rd.Next(int MinValue,int MaxValue): Sinh ngẫu nhiên một số kiểu int có giá trị từ MinValue đến MaxValue.

* rd.Next(int.MaxValue): Sinh ngẫu nhiên một số kiểu int có giá trị từ lớn nhất là MaxValue

* rd.NextDouble(): sinh ngẫu nhiên một số kiểu double có giá trị từ 0 đến 1.

Các phép toán so sánh

+ Lớn hơn (>), lớn hơn hoặc bằng (\geq)

+ Nhỏ hơn (<), nhỏ hơn hoặc bằng (\leq)

+ Bằng ($=$), khác (\neq).

2. Kiểu chuỗi

Đối tượng kiểu string thường chứa một chuỗi ký tự. Để khai báo chuỗi ta dùng cú pháp như sau: `string myString;`

Sau đó ta có thể gán giá trị của chuỗi: `myString="Chào bạn";`

Hoặc ta có thể vừa khai báo vừa khởi tạo như sau: `string myString="Chào bạn";`

Trong C# có hỗ trợ font Unicode nên ta có thể gõ tiếng việt có dấu.

Các phép toán trên kiểu chuỗi

- + `s + t`: toán tử ghép chuỗi. Ví dụ: "Hà Nội" + "mùa thu" cho kết quả "Hà Nội mùa thu"
- + `s.Length`: trả về chiều dài của chuỗi s. Ví dụ: "Đạt Anh".Length có kết quả =7.
- + `s.Replace(str1,str2)`: Thay thế chuỗi str1 trong chuỗi s bằng chuỗi str2. Ví dụ: "Nguyễn Thu Hường".Replace("Thu", "Anh") cho kết quả chuỗi "Nguyễn Anh Hường".
- + `s.Substring(vt,n)`: trả về một chuỗi con gồm n ký tự trong chuỗi s, bắt đầu từ ký tự ở vị trí vt. Ví dụ: "Hoa hồng".Substring(1,2) cho kết quả "oa".
- + `s.Insert(vt,str)`: chèn thêm giá trị của chuỗi str vào chuỗi s tại vị trí vt. Ví dụ: "Trời xanh".Insert(4,"màu") cho kết quả "Trời màu xanh".
- + `s.ToLower`: biến đổi chuỗi s về chữ in thường. Ví dụ: "Hà Nội".ToLower cho kết quả "hà nội"
- + `s.ToUpper`: biến đổi chuỗi s về chữ in hoa. Ví dụ: "Hà Nội".ToUpper có kết quả "HÀ NỘI"
- + `s.Remove(vt,n)`: xóa n ký tự trong chuỗi s, bắt đầu từ ký tự ở vị trí vt. Ví dụ: "Hoa hồng ".Remove(1, 2) cho kết quả "H hồng". (Chuỗi được tính từ vị trí 0)
- + `s.TrimStart`: xóa các ký tự rỗng ở đầu chuỗi s. Ví dụ: " Hoa hồng ".TrimStart cho kết quả "Hoa hồng ".
- + `s.TrimEnd`: xóa các ký tự rỗng ở cuối chuỗi s. Ví dụ: " Hoa hồng ".TrimEnd cho kết quả " Hoa hồng "
- + `s.Trim`: xóa các ký tự rỗng ở đầu và cuối chuỗi s. Ví dụ: " Hoa hồng ".Trim cho kết quả "Hoa hồng".

Bài giảng lập trình trực quan

-
- + `s.Split('c')`: Trả về mảng chuỗi được phân định bởi ký tự c.
 - + `Convert.ToInt16(ch)`: trả về mã ASCII của ký tự ch. Ví dụ: `Convert.ToInt16('A')`=65
 - + `Convert.ToChar(n)`: trả về ký tự có mã ASCII = n. Ví dụ: `Convert.ToChar(65)`=”A”

3. Kiểu đúng sai

Là kiểu dữ liệu chỉ nhận một trong hai giá trị True/False.

Các phép toán trên kiểu dữ liệu bool

Phép toán và – &&: xét biểu thức A && B chỉ nhận giá trị đúng khi và chỉ khi cả A và B cùng nhận giá trị đúng. Còn nhận giá trị sai trong tất cả các trường hợp còn lại.

Phép toán hoặc – //: xét biểu thức A || B chỉ nhận giá trị sai khi và chỉ khi cả A và B cùng nhận giá trị sai. Còn nhận giá trị đúng trong tất cả các trường hợp còn lại.

Phép toán phủ định – !: xét biểu thức !A sẽ nhận giá trị đúng khi A nhận giá trị sai và ngược lại.

Bảng giá trị chân lý của các phép toán:

A	B	A && B	A // B	A	! A
True	True	True	True	False	True
True	False	False	True	True	False
False	True	False	True		
False	False	False	False		

4. Kiểu ngày tháng

Kiểu ngày tháng trong C# được dùng qua lớp `DateTime`. Để tạo thời gian ta làm như sau: `DateTime dt=new DateTime(int year, int month, int day, int hour, int minute, int second)`

Ví dụ: Khởi tạo thời gian `DateTime` ngày 23/08/2012 8:30:20 như sau

```
DateTime dt=new DateTime(2012, 8, 23, 8, 30, 20)
```

Các định dạng kiểu ngày tháng

Có các định dạng tùy chỉnh sau đây: `y` (năm), `m` (tháng), `d` (ngày), `h` (giờ 12), `H` (giờ 24), `m` (phút), `s` (giây), `f` (phần nhỏ của giây), `F` (phần nhỏ của giây, không bao gồm số 0 sau cùng), `t` (PM hoặc AM) và `z` (múi giờ).

Bài giảng lập trình trực quan

```
String.Format("{0:y YY YYY YYYY}", dt); // "8 08 008 2008" năm
String.Format("{0:M MM MMM MMMM}", dt); // "3 03 Mar March" tháng
String.Format("{0:d dd ddd dddd}", dt); // "9 09 Sun Sunday" ngày
String.Format("{0:h hh H HH}", dt); // "4 04 16 16" giờ 12/24
String.Format("{0:m mm}", dt); // "5 05" phút
String.Format("{0:s ss}", dt); // "7 07" giây
String.Format("{0:f ff fff ffff}", dt); // "1 12 123 1230" phần nhỏ của giây
String.Format("{0:F FF FFF FFFF}", dt); // "1 12 123 123" phần nhỏ của giây
String.Format("{0:t tt}", dt); // "P PM" A.M. or P.M.
String.Format("{0:z zz zzz}", dt); // "-6 -06 -06:00" muối giờ
```

Chèn vào dấu phân cách “/” hoặc “:”, các ký tự này sẽ được chèn vào kết quả đầu ra để xác định thời gian:

```
//Dấu phân cách trong định dạng của người Đức là "." (chuyển "/" => ".")
String.Format("{0:d/M/yyyy HH:mm:ss}", dt); // "9/3/2008 16:05:07" - english (en-US)
String.Format("{0:d/M/yyyy HH:mm:ss}", dt); // "9.3.2008 16:05:07" - german (de-DE)
//Đuôi đây là một vài ví dụ về tùy chỉnh định dạng kiểu DateTime
//Định dạng tháng/ngày/năm với ký tự "0" và không có ký tự "0"
String.Format("{0:M/d/yyyy}", dt); // "3/9/2008"
String.Format("{0:MM/dd/yyyy}", dt); // "03/09/2008"
//Lấy tên của ngày/tháng
String.Format("{0:ddd, MMM d, yyyy}", dt); // "Sun, Mar 9, 2008"
String.Format("{0:dddd, MMMM d, yyyy}", dt); // "Sunday, March 9, 2008"
//Lấy năm với 2, 4 ký tự
String.Format("{0:MM/dd/yy}", dt); // "03/09/08"
String.Format("{0:MM/dd/yyyy}", dt); // "03/09/2008"
```

Các phép toán trên kiểu ngày tháng

DateTime.Now: trả về ngày và giờ hiện hành, ví dụ: 15/08/2009 5:20:28PM

DateTime.DaysInMonth(int year,int month): trả về số ngày trong tháng month của năm year.

dt.Month: trả về tháng của DateTime dt

dt.Year: trả về năm của DateTime dt

dt.Day: trả về ngày của DateTime dt

dt.AddDays(n): trả về một ngày mới cách ngày DateTime dt n ngày.

Date.AddMonths(n): trả về một ngày mới cách ngày DateTime dt n tháng.

Date.AddYears(n): trả về một ngày mới cách ngày Date n năm.

Để tính khoảng cách giữa 2 ngày ta làm như ví dụ sau:

Bài giảng lập trình trực quan

```
DateTime dt1=new DateTime(2012,8,5); //Tạo thời gian 5/8/2012  
DateTime dt2 = Convert.ToDateTime("24/08/2012"); //Tạo thời gian 24/08/2012  
TimeSpan time = dt2 - dt1; //Dùng đối tượng TimeSpan để tính khoảng cách 2 ngày  
int ngay = time.Days; //Lấy khoảng cách từ ngày dt1 đến dt2. Kết quả =19
```

Chú ý: định dạng hiển thị của ngày tháng phụ thuộc vào quy định của Windows, giá trị mặc định là hiển thị theo định dạng kiểu Mỹ (tháng trước - ngày sau). Ta có thể thay đổi dạng hiển thị ngày trước - tháng sau bằng cách thay đổi trong hộp hội thoại Date tại mục Regional Setting của cửa sổ Control Panel trong Windows.

5. Kiểu mảng

Mảng là một tập hợp các biến có cùng tên và cùng kiểu dữ liệu. Dùng mảng làm chương trình đơn giản và ngắn gọn hơn.

Ta có thể truy xuất các phần tử của mảng thông qua chỉ số, chỉ số của mảng bắt đầu từ 0.

Có mảng một chiều và mảng nhiều chiều.

Mảng một chiều

Cú pháp: `type[] array-name;`

Ta có thể khai báo mảng với kích thước cố định ngay từ đầu. Ví dụ:

```
int[5] a; // Khai báo mảng a có 5 phần tử nguyên.
```

Ta cũng có thể khai báo mảng, sau đó cấp phát bộ nhớ cho mảng với từ khóa new. Điều này giúp ta quản lý bộ nhớ tốt hơn. Kích thước của mảng sẽ được xác định lúc thi hành chương trình.

```
type[] array-name;
```

```
array-name =new type[n]; //khai báo mảng có n phần tử.
```

Ví dụ: đoạn chương trình sau sẽ tạo ra 5 phần tử nguyên cho mảng a

```
int[] a;  
int n = 5;  
Random rd = new Random();  
a = new int[n];  
for(int i=0;i<n;i++)  
{  
    a[i] = rd.Next();  
}
```

Ta cũng có thể gán các phần tử của mảng khi khai báo như sau:

```
string[] myArray = {"first element", "second element", "third element"};
```

Bài giảng lập trình trực quan

Ta có thể làm việc với mảng:

Array.Reverse(a): Đảo ngược mảng a.

Array.Sort(a): Sắp xếp mảng a theo chiều tăng dần.

a.Length: Trả về chiều dài của mảng a.

Mảng nhiều chiều

Cú pháp: `type[,] array-name;`

Ta có thể khai báo mảng với kích thước cố định ngay từ đầu. Ví dụ:

`int[2,3] a; // Khai báo mảng nguyên a có 2 hàng 3 cột.`

Cũng như mảng một chiều, ta cũng có thể khai báo mảng, sau đó cấp phát bộ nhớ cho mảng với từ khóa new.

`type[,] array-name;`

`array-name =new type[n,m]; //khai báo mảng có n hàng, m cột`

3.2.4. Chuyển đổi giữa các kiểu dữ liệu

Hàm chuyển đổi	Đổi giá trị sang kiểu
Convert.ToBoolean(Giatri)	Boolean
Convert.ToByte(Giatri)	Byte
Convert.ToDateTime(Giatri)	Date
Convert.ToDouble(Giatri)	Double
Convert.ToInt16(Giatri)	Integer – 2 byte
Convert.ToInt32(Giatri)	Integer – 4 byte
Convert.ToInt64(Giatri)	Integer – 8 byte
Convert.ToString(Giatri)	String

Chú ý rằng giá trị truyền cho hàm phải hợp lệ, nghĩa là phải thuộc miền giá trị của kiểu kết quả nếu không C# sẽ báo lỗi. Ví dụ:

- + Convert.ToDateTime("12/12/1988") trả về giá trị kiểu ngày tháng 12/12/1988
- + Convert.ToInt32("25") = 25.
- + Convert.ToInt32("25a") hoặc Convert.ToInt32("a25") sẽ báo lỗi.

3.3. Hộp thoại thông báo - Message Box

3.3.1. Khái niệm

Hộp thông báo là hộp thoại cung cấp thông tin để tương tác với người sử dụng, đồng thời cũng là nơi hiển thị các kết quả trung gian trong quá trình tính toán. Trong thời gian hiển thị thông báo. C# ngừng mọi hoạt động của biểu mẫu và người dùng chỉ có thể làm việc với hộp thông báo.

3.3.2. Hộp thông báo MessageBox

Cú pháp hộp thông báo:

`MessageBox.Show(Nội dung thông báo, Tiêu đề, Kiểu chức năng, Kiểu biểu tượng)`

Trong đó:

- + Cú pháp hộp thông báo không nhất thiết phải có đầy đủ 4 thành phần trên.
- + Nội dung cần thông báo và tiêu đề của hộp thông báo được đặt trong cặp dấu nháy kép.
- + Kiểu chức năng và kiểu biểu tượng có các giá trị như sau:

Hàng tượng trưng	Thể hiện	Ý nghĩa
Các kiểu chức năng: được bắt đầu bởi MessageBoxButtons		
.OK		Chỉ hiển thị nút OK.
.OKCancel		Hiển thị các nút OK và Cancel.
.AbortRetryIgnore		Hiển thị các nút Abort, Retry và Ignore.
.YesNoCancel		Hiển thị các nút Yes, No và Cancel.
.YesNo		Hiển thị các nút Yes, No.
.RetryCancel		Hiển thị các nút Retry, Cancel.
Các kiểu biểu tượng: được bắt đầu bởi MessageBoxIcon		
.Error hoặc .Hand hoặc .Stop		Dùng cho những thông báo lỗi thất bại khi thi hành một công việc nào đó.
.Question		Dùng cho những câu hỏi yêu cầu người sử dụng chọn lựa.
.Exclamation hoặc .Warning		Dùng cho các thông báo của chương trình.
.Asterisk hoặc .Information		Dùng cho các thông báo cung cấp thêm thông tin cho người sử dụng.
.None		Không hiển thị biểu tượng.

Bài giảng lập trình trực quan

Ví dụ hiển thị hộp thông báo “Bạn chưa nhập dữ liệu” với một nút lệnh OK và biểu tượng Information ta viết như sau:

```
MessageBox.Show("Bạn chưa nhập dữ liệu", "Thông báo", MessageBoxButtons.OK,  
                MessageBoxIcon.Information);
```

Kết quả ta có:



3.3.3. Hàm thông báo MessageBox

Ngoài chức năng thông báo, hàm MessageBox còn trả về giá trị của các nút chức năng mà người dùng đã chọn.

Cú pháp của hàm MessageBox như sau:

MessageBox.Show(Nội dung thông báo, Tiêu đề, Kiểu chức năng, Kiểu biểu tượng)=
Giá trị trả về.

Trong đó giá trị trả về bao gồm:

- + System.Windows.Forms.DialogResult.OK
- + System.Windows.Forms.DialogResult.Cancel
- + System.Windows.Forms.DialogResult.Abort
- + System.Windows.Forms.DialogResult.Retry
- + System.Windows.Forms.DialogResult.Ignore
- + System.Windows.Forms.DialogResult.Yes
- + System.Windows.Forms.DialogResult.No

Ví dụ: Ta có thể viết code cho nút btnExit với yêu cầu chỉ thoát khi người dùng trả lời có muốn thoát như sau:

```
private void btnThoat_Click(object sender, EventArgs e)  
{  
    if (MessageBox.Show("Bạn có muốn thoát không?", "Thông báo", MessageBoxButtons.YesNo,  
                        MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)  
        this.Close();  
}
```

3.4. Các cấu trúc điều khiển

3.4.1. Câu lệnh lựa chọn if

Lệnh **if** có 2 dạng sau:

Dạng 1:

```
if (<Biểu thức điều kiện>
    <khoi lệnh>
```

Hoạt động: Khi gặp câu lệnh này, máy sẽ tính giá trị của biểu thức. Nếu biểu thức đúng thì máy sẽ thực hiện khối lệnh và sau đó thực hiện các lệnh tiếp theo. Nếu biểu thức sai thì máy sẽ bỏ qua khối lệnh và chuyển đến các lệnh tiếp theo.

Dạng 2:

```
if (<Biểu thức điều kiện>
    <Khối lệnh 1>
else
    <Khối lệnh 2>
```

Hoạt động: Khi gặp câu lệnh này, máy sẽ tính giá trị của biểu thức. Nếu biểu thức đúng máy sẽ thực hiện khối lệnh 1 sau đó nhảy tới các lệnh sau khối lệnh 2. Nếu biểu thức sai thì máy không thực hiện khối lệnh 1 mà chỉ thực hiện khối lệnh 2 và sau đó thực hiện các lệnh viết sau nó.

Chú ý: Lệnh if có thể lồng nhau.

3.4.2. Câu lệnh lựa chọn switch... case

```
switch (Biểu thức kiểm tra)
{
    case n1:
        Các câu lệnh
        break;
    case n2:
        Các câu lệnh
        break;
    case nk:
        Các câu lệnh
        break;
    [default:
        Các câu lệnh]
}
```

Hoạt động: Sự hoạt động của toán tử switch phụ thuộc vào giá trị của biểu thức kiểm tra viết trong dấu ngoặc tròn.

Bài giảng lập trình trực quan

- Khi giá trị này bằng ni máy sẽ nhảy tới câu lệnh có nhãn case ni.
- Khi giá trị của biểu thức khác tất cả các ni thì cách làm việc của máy lại phụ thuộc vào sự có mặt hay vắng mặt của default
 - o Khi có default, máy nhảy tới câu lệnh có nhãn default.
 - o Khi không có default, máy ra khỏi toán tử switch.

3.4.3. Cấu trúc lặp for

Lệnh for cho phép thực hiện một khối lệnh một số lần hữu hạn. Lệnh for thường dùng để giải các bài toán có tính chu trình, ví dụ như các bài toán về dãy số, về ma trận.

Cú pháp của for

Lệnh for có dạng sau:

```
for (<Biểu thức khởi tạo>; <Biểu thức kiểm tra>; <Biểu thức tăng>)  
    <Khối lệnh>
```

Hoạt động của for

Lệnh for làm việc theo các bước sau:

- Bước 1: Tính giá trị biểu thức khởi tạo.
- Bước 2: Tính giá trị của biểu thức kiểm tra. Nếu đúng tới bước 3, nếu sai thoát khỏi lệnh for.
- Bước 3: Thực hiện khối lệnh.
- Bước 4: Tính giá trị biểu thức tăng sau đó qua lại bước 2.

Chú ý:

- Nếu biểu thức kiểm tra vắng mặt thì nó được xem là đúng. Để tránh lặp vô hạn thì trong phần thân của lệnh for phải có lệnh nhảy ra khỏi for bằng các lệnh break, return hoặc goto.

3.4.4. Cấu trúc lặp while

Cú pháp của while

Cú pháp của while có dạng:

while (<Biểu thức kiểm tra>)

<Khối lệnh>

Trong cú pháp trên:

- while: từ khóa của lệnh while.
- Biểu thức kiểm tra bắt buộc phải đặt trong cặp ngoặc tròn.
- Khối lệnh, còn gọi là phần thân của while, là khối lệnh cần thực hiện nhiều lần.

Hoạt động của while

Lệnh while làm việc theo các bước sau:

- Bước 1: Tính giá trị biểu thức kiểm tra. Nếu biểu thức đúng sang bước 2, ngược lại thoát khỏi lệnh while.
- Bước 2: Thực hiện khối lệnh sau đó quay trở lại bước 1.

Nhận xét: Thân của lệnh while có thể được thực hiện một lần hoặc nhiều lần và cũng có thể không được thực hiện lần nào nếu ngay từ đầu biểu thức kiểm tra đã sai.

3.4.5. Cấu trúc lặp do...while

Lệnh **do ... while** có dạng sau:

do

<Khối lệnh>

while (Biểu thức kiểm tra);

Hoạt động của do ... while

Lệnh **do ... while** thực hiện như sau:

Bài giảng lập trình trực quan

- Bước 1: Thực hiện khối lệnh.
- Bước 2: Tính giá trị biểu thức kiểm tra. Nếu đúng quay trở lại bước 1, ngược lại thoát khỏi lệnh **do ... while**.

3.4.6. Cấu trúc lặp foreach

Cho phép bạn rão qua tất cả các phần tử của mảng hoặc các tập hợp khác, và tuần tự xem xét từng phần tử một. Cú pháp như sau:

```
foreach (<kiểu dl> <tên biến truy cập> in <mảng hoặc tập hợp>)
    <khối lệnh>
```

3.4.7. Câu lệnh try...catch

Được dùng trong các câu lệnh bẫy lỗi ngoại lệ của chương trình. Cú pháp như sau:

```
try
{
    //Lệnh thực thi trong trường hợp bình thường
}
catch
{
    //Lệnh xử lý lỗi
}
```

3.4.8. Câu lệnh break

Ta dùng câu lệnh break khi muốn ngưng ngang xương việc thi hành và thoát khỏi vòng lặp.

3.4.9. Câu lệnh Continue

Câu lệnh continue được dùng trong vòng lặp khi bạn muốn khởi động lại một vòng lặp nhưng lại không muốn thi hành phần lệnh còn lại trong vòng lặp, ở một điểm nào đó trong thân vòng lặp.

3.5. Phương thức trong C#

3.5.1. Định nghĩa phương thức

Cú pháp xây dựng một phương thức như sau:

```
[modifiers] return_type MethodName([parameters])
{
    // Thân phương thức
}
```

Modifiers: Là phạm vi truy xuất có thể là private, public... thành phần này có thể vắng mặt, nếu vắng mặt thì Modifiers được coi mặc định là private.

Bài giảng lập trình trực quan

Có 2 loại phương thức như trong C++, java đó là:

- + Phương thức có giá trị trả về (Có kiểu khác void). Loại này trong thân phương thức phải có lệnh return.
- + Phương thức không có giá trị trả về, phương thức void. Loại này không có lệnh return trong thân hàm.

3.5.2. Gọi phương thức

Phương thức có giá trị trả về được sử dụng như một thành phần của biểu thức, nên ta có thể dùng theo cú pháp sau:

<biến> = MethodName([parameters])

Còn phương thức kiểu void (không có giá trị trả về) được dùng như một lệnh độc lập, nó không được dùng trong lệnh gán hay biểu thức

MethodName([parameters])

3.6. Gõ rối chương trình

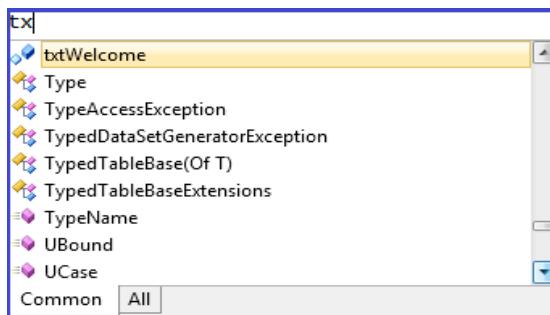
Khi lập trình, người lập trình không tránh khỏi những sai sót và mắc lỗi, tuy nhiên ta có thể giảm khả năng mắc lỗi đến mức tối thiểu.

3.6.1. Một số giải pháp gõ rối chương trình

- ❖ Thiết kế cẩn thận, ghi chú các vấn đề quan trọng và cách giải quyết cho từng phần.
Ghi chú từng phương thức và mục đích của nó.
- ❖ Chú thích rõ ràng trong chương trình.
 - Để đặt chú thích cho các dòng lệnh, ta có thể gõ dấu nháy đơn // tại vị trí cần đặt chú thích, hoặc bôi đen các dòng lệnh rồi kích vào biểu tượng  trên thanh công cụ standard.
 - Để xóa dấu chú thích ở các dòng lệnh, ta có thể xóa dấu // hoặc bôi đen các dòng lệnh muốn xóa chú thích rồi kích chọn biểu tượng  trên thanh công cụ Standard.
- ❖ Dùng cửa sổ danh sách các thuộc tính, phương thức, các hằng số, các lớp đối tượng... trong C# để tránh gõ sai tên thuộc tính, phương thức, ...

Bài giảng lập trình trực quan

- Để gọi cửa sổ này, trong cửa sổ soạn thảo code bấm tổ hợp phím Ctrl + J, kết quả là khi người dùng gõ các ký tự bất kỳ, con trỏ sẽ tự động cuộn tới dòng đầu tiên chứa các ký tự đó cho người dùng chọn.



- Sử dụng câu lệnh try...catch để bắt các lỗi ngoại lệ của chương trình.

3.6.2. Dò lỗi từng dòng lệnh

Một trong những cách dò lỗi hiệu quả là thực hiện chương trình bằng cách chạy từng dòng lệnh. Khi đó, ta có thể kiểm tra sự thay đổi giá trị của các biến, các thuộc tính... qua từng bước thực hiện chương trình. Để tìm hiểu cách dò lỗi từng dòng lệnh ta thực hiện theo các bước của ví dụ sau:

Đề bài: viết một phương thức tính giai thừa, sau đó tính và in ra kết quả của 6!

Bước 1: mở một dự án mới rồi gõ đoạn mã sau vào cửa sổ code.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace MinhHoaChuong3
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        long GiaiThua(int n)
        {
            int i;
            long gt;
            gt = 1;
            for (i = 1; i <= n; i++)
                gt *= i;
            return gt;
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            long a;
            a = GiaiThua(6);
            MessageBox.Show(a.ToString());
        }
    }
}
```

Bài giảng lập trình trực quan

Bước 2: bấm F5 để thực hiện chương trình, ta thấy kết quả của chương trình là $6!=22$. Đây là một kết quả sai. Nhiệm vụ của chúng ta là xác định chương trình sai ở đâu?

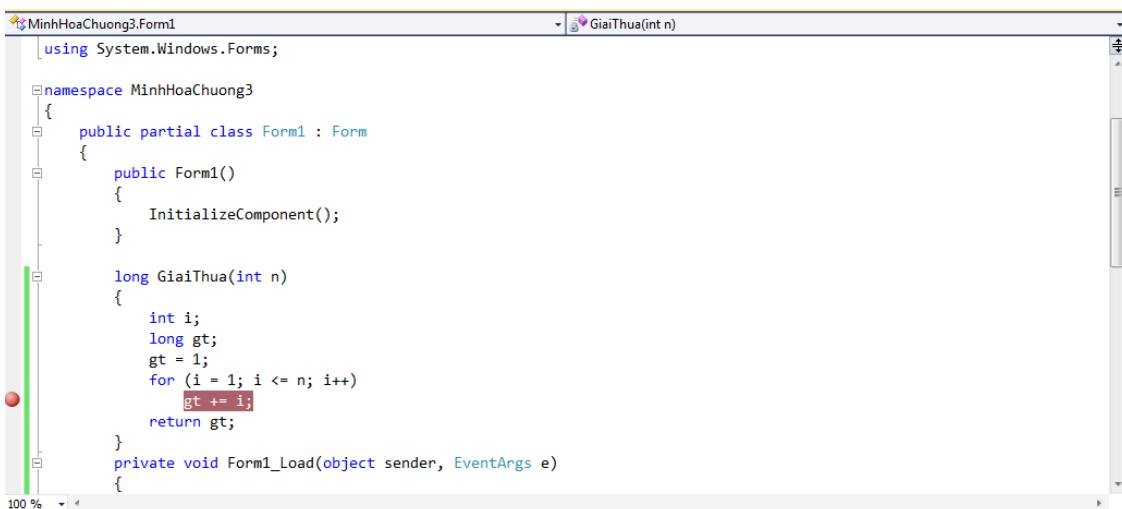
Bước 3: khoanh vùng và đặt điểm dừng Breakpoint tại dòng mã lệnh ta nghi ngờ là sai để theo dõi kết quả thực hiện của nó. Trong cửa sổ soạn thảo code, đặt con trỏ tại dòng lệnh muốn đặt Breakpoint (ví dụ dòng `gt += i`) và thực hiện theo 1 trong 3 cách sau:

+ Vào Debug\Toggle Breakpoint

+ Bấm phím F9

+ Di chuột ra bên lề trái của cửa sổ soạn thảo code, kích trái chuột tại dòng lệnh muốn đặt Breakpoint

Kết quả C# sẽ hiển thị một vòng tròn màu đỏ và tô nền màu đỏ tại dòng lệnh đã đặt Breakpoint. Khi thực hiện đến dòng lệnh này, chương trình sẽ dừng lại để ta tự chạy chương trình kiểm tra từng bước.



The screenshot shows the Visual Studio code editor with the following code:

```
MinhHoaChuong3.Form1
using System.Windows.Forms;

namespace MinhHoaChuong3
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        long GiaiThua(int n)
        {
            int i;
            long gt;
            gt = 1;
            for (i = 1; i <= n; i++)
                gt += i;
            return gt;
        }
        private void Form1_Load(object sender, EventArgs e)
        {
        }
    }
}
```

A red circular breakpoint icon is placed on the line `gt += i;`. The background of this line is also highlighted in red, indicating it is the current line of execution.

Bước 4: bấm F5 để thực hiện chương trình, C# dừng chương trình tại dòng lệnh đặt Breakpoint và màu nền của dòng lệnh này đã chuyển từ màu đỏ sang màu vàng cho biết tiếp theo chương trình sẽ thực hiện lệnh này.

Bước 5: nhấn nút Step Into  trên thanh công cụ Standard để thực hiện dòng lệnh có nền màu vàng, rồi vào Debug\Windows\Autos để mở cửa sổ Autos xem giá trị của các biến tại thời điểm hiện thời.

Bài giảng lập trình trực quan

```
long GiaiThua(int n)
{
    int i;
    long gt;
    gt = 1;
    for (i = 1; i <= n; i++)
        gt *= i;
    return gt;
}

private void Form1_Load(object sender, EventArgs e)
{
    long a;
    a = GiaiThua(6);
    MessageBox.Show(a.ToString());
}
```

Name	Value	Type
gt	2	long
i	1	int
n	6	int
this	{MinhHoaChuong3.Form1, Text: Form1}	MinhHoaChuong3.Form1

Trong cửa sổ Autos giá trị của gt = 2.0, i=1 và n=6. Ta thấy giá trị của gt sai vì với i=1 thì gt cũng phải có giá trị là 1.0

Bước 6: tiếp tục nhấn 2 lần nút Step Into, cửa sổ Autos có giá trị như sau

Name	Value	Type
gt	4	long
i	2	int
n	6	int
this	{MinhHoaChuong3.Form1, Text: Form1}	MinhHoaChuong3.Form1

Bước 7: tiếp tục nhấn 2 lần nút Step Into, ta có:

Name	Value	Type
gt	7	long
i	3	int
n	6	int
this	{MinhHoaChuong3.Form1, Text: Form1}	MinhHoaChuong3.Form1

Đến đây ta thấy giá trị gt mới được tính bằng cách cộng thêm giá trị của i hiện thời vào giá trị gt cũ, chứ không phải là nhân theo cách tính giai thừa. Như vậy trong công thức tính gt ta đã gõ nhầm dấu * thành dấu +.

Bước 8: nhấn nút Stop Debugging để thoát khỏi chế độ ngắt trở về môi trường phát triển của Visual Studio và sửa lại công thức tính Gt thành Gt *= i.

Bước 9: ta thấy khi trở về môi trường phát triển của Visual Studio thì điểm dừng Breakpoint vẫn tồn tại. Để loại bỏ điểm dừng Breakpoint, trong cửa sổ soạn thảo code, đặt con trỏ tại dòng lệnh muốn bỏ Breakpoint và thực hiện theo 1 trong 3 cách sau:

+ Vào Debug\Toggle Breakpoint

+ Bấm phím F9

+ Di chuột ra bên lề trái của cửa sổ soạn thảo code, kích trái chuột tại dòng lệnh đang đặt Breakpoint.

Bài giảng lập trình trực quan

Ngoài ra, nếu chương trình có nhiều điểm Breakpoint và để loại bỏ hết các điểm này cùng một lúc thì ta thực hiện theo 1 trong 2 cách sau:

+ Vào Debug\Delete All Breakpoints

+ Bấm tổ hợp phím Ctrl + Shift + F9

Bước 10: đến đây hãy bấm F5 để chạy lại chương trình, ta thấy kết quả là $6! = 720$

CHƯƠNG 4: TÌM HIỂU CÁC ĐIỀU KHIỂN CƠ BẢN

4.1. Thuộc tính, phương thức, sự kiện và các mối quan hệ giữa chúng

Mỗi một đối tượng trong C# đều có 3 đặc tính là *Thuộc tính - Properties*, *Phương thức - Methods* và *Sự kiện - Events*. Trong đó:

+ **Properties:** là tập hợp các đặc tính để mô tả một đối tượng như: tên, chiều cao, chiều rộng, màu chữ, màu nền... Các thuộc tính có thể xác định trong khi thiết kế (Design time) hoặc trong lúc thi hành (Run time).

+ **Methodes:** là những đoạn chương trình chứa trong điều khiển, cho điều khiển biết cách thức để thực hiện một công việc nào đó, chẳng hạn làm ẩn sự xuất hiện của một điều khiển (phương thức Hide).

+ **Evens:** nếu như thuộc tính mô tả đối tượng, phương thức chỉ ra cách thức đối tượng hành động thì sự kiện là những phản ứng của đối tượng. Khi tạo một chương trình trong C#, ta lập trình chủ yếu theo sự kiện, lập trình theo cách này có nghĩa là ta phải biết khi nào sự kiện xảy ra và làm gì khi sự kiện đó xảy ra? Điều này có nghĩa là chương trình chỉ thi hành khi người dùng thực hiện một thao tác nào đó trên giao diện.

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

Mặc dù thuộc tính, phương thức và sự kiện có vai trò khác nhau nhưng chúng thường xuyên liên hệ với nhau. Ví dụ, nếu ta di chuyển một điều khiển bằng phương thức Move thì một số thuộc tính như Top, Height, Left, Width sẽ thay đổi theo, khi đó kích cỡ của điều khiển thay đổi tức là sự kiện Resize xảy ra.

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

Ví dụ: ta có 2 cách để làm biến frmWelcome xuất hiện và biến mất trên màn hình
Thực hiện bằng thuộc tính:

Xuất hiện: frmWelcome.Visible = True

Biến mất: frmWelcome.Visible = False

Thực hiện bằng phương thức:

Xuất hiện: frmWelcome.Show()

Biến mất: frmWelcome.Hide()

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

4.2.1. Form

1. Thuộc tính

Name	Tên form, bắt đầu bởi tiếp đầu ngữ frm
BackColor	Thiết lập màu nền cho Form.
BackgroundImage	Thiết lập ảnh nền cho Form.
BackgroundImageLayout	Thiết lập chế độ hiển thị ảnh nền trên Form. Tile: hiển thị ảnh từ trên xuống, Center: hiển thị ảnh từ giữa ra, Stretch: dãn đều ảnh trên Form.
Cursor	Thiết lập chế độ hiển thị con trỏ trên Form.
Enabled	Nếu nhận giá trị True thì cho phép người dùng tác động lên Form, ngược lại thì nhận giá trị False.
Font	Thiết lập kiểu chữ, cỡ chữ cho các điều khiển trên Form.
ForeColor	Thiết lập màu chữ cho các điều khiển trên Form.
FormBorderStyle	Thiết lập kiểu đường viền cho Form. Fixed Single: không thể thay đổi kích thước của Form, Sizable: có thể phóng to thu nhỏ và thay đổi kích thước của Form, Sizable ToolWindow: có thể thay đổi kích thước của Form...
Icon	Thiết lập biểu tượng cho Form (các tệp ảnh có đuôi .ico).
MainMenuStrip	Gắn kết Form với Menu.
Opacity	Thiết lập độ trong suốt cho nền của Form, nếu độ trong suốt < 100% thì Form sẽ dần trở nên trong suốt có thể nhìn xuyên qua thấy những gì nằm bên dưới Form.
ShowIcon	Nếu nhận giá trị True thì cho phép hiển thị biểu tượng đã được thiết lập ở thuộc tính Icon, ngược lại thì nhận giá trị False.
StartPosition	Thiết lập vị trí xuất hiện của Form trên màn hình. Manual: xuất hiện ở góc trên bên trái màn hình, CenterScreen: xuất hiện ở giữa màn hình...
Text	Thiết lập dòng tiêu đề của Form.
WindowState	Thiết lập trạng thái của Form khi chạy chương trình. Normal: hiển thị Form đúng theo kích cỡ thiết kế, Maximized: phóng to Form bằng màn hình, Minimized: thu nhỏ Form trên thanh Taskbar của hệ điều hành.

2. Sự kiện

Form có một số sự kiện thông dụng như sau:

Load	Sự kiện Load được kích hoạt khi Form được nạp vào bộ nhớ, nó thường được dùng để khởi tạo các giá trị và trạng thái cho các biến, các điều khiển... trên Form.
Click	Được kích hoạt khi người dùng dùng chuột trên Form.

Bài giảng lập trình trực quan

FormClosed	Được kích hoạt khi người dùng kích chuột vào nút Close x ở góc trên bên phải để đóng Form.
FormClosing	Cũng được kích hoạt khi người dùng kích chuột vào nút Close x, nhưng xảy ra trước sự kiện FormClosed tức là được phát sinh trước khi cửa sổ Form chuẩn bị đóng lại.

Chú ý: Nếu không muốn người dùng đóng Form bằng cách bấm chọn biểu tượng Close thì trong sự kiện **FormClosing** ta đặt thuộc tính Cancel = True như sau:

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    e.Cancel = true;
}
```

4.2.2. Hộp văn bản – TextBox

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

1. Thuộc tính

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

Bài giảng lập trình trực quan

ScrollBars	Thiết lập thanh cuộn ngang và dọc cho hộp văn bản, có hiệu lực khi thuộc tính Multiline = True. Chú ý: thanh cuộn ngang chỉ có hiệu lực khi thuộc tính WordWrap = False.
TabIndex	Thứ tự truy cập của hộp văn bản khi người dùng bấm phím Tab, thứ tự đầu tiên là 0.
Text	Chứa nội dung của hộp văn bản.
 TextAlign	Thiết lập chế độ căn chỉnh: trái, phải hoặc giữa của dữ liệu trong hộp TextBox.
Visible	Visible = True: hiển thị hộp văn bản, Visible = False: ẩn hộp văn bản.
WordWrap	Chỉ có hiệu lực khi thuộc tính Multiline = True. WordWrap = True: dòng văn bản được tự động cuộn xuống dòng khi gấp lè bên phải của hộp TextBox, ngược lại thì nhận giá trị False.

2. Sự kiện

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

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

Ví dụ 1: Để hiển thị mã Ascii của một ký tự bất kỳ được gõ vào hộp văn bản **txta** ta có đoạn chương trình như sau cho sự kiện KeyDown:

```
private void txta_KeyDown(object sender, KeyEventArgs e)
{
    MessageBox.Show(e.KeyValue.ToString());
}
```

Bài giảng lập trình trực quan

Ví dụ 2: Dùng thủ tục **KeyPress** để kiểm tra việc nhập dữ liệu: chỉ cho phép nhập vào hộp văn bản **txtb** các số từ 0 tới 9 và phím Backspace (có mã Ascii = 8) để xóa dữ liệu, ta có đoạn chương trình như sau:

```
private void txtb_KeyPress(object sender, KeyPressEventArgs e)
{
    if((Convert.ToInt16(e.KeyChar) < Convert.ToInt16('0') || Convert.ToInt16(e.KeyChar) > Convert.ToInt16('9'))
        && (Convert.ToInt16(e.KeyChar) != 8))
    {
        MessageBox.Show("Bạn phải nhập số");
        e.Handled = true;
    }
}
```

Nếu mỗi ký tự được nhập vào hộp Textbox nằm ngoài khoảng từ ký tự "0" đến ký tự "9" và mã Ascii của ký tự đó khác 8 thì thông báo phi nhập số và đổi ký tự đó thành ký tự rỗng. Vì vậy hộp văn bản chỉ nhận các số từ 0 đến 9 và phím xóa Backspace.

4.2.3. Nút lệnh – Button

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

1. Thuộc tính

Name	Tên nút lệnh, bắt đầu bởi tiếp đầu ngữ btn
BackColor	Thiết lập màu nền cho nút lệnh.
BackgroundImage	Thiết lập ảnh nền cho nút lệnh.
Enabled	Enabled=False: người dùng không thể tác động lên nút lệnh, ngược lại thì bằng True.
Font	Xác lập kiểu chữ và cỡ chữ cho nút lệnh.
ForeColor	Thiết lập màu chữ cho nút lệnh.
Image	Thiết lập ảnh hiển thị trên nút lệnh.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của nút lệnh trên Form, ngược lại thì nhận giá trị False.
TabIndex	Thứ tự truy cập của nút lệnh khi người dùng bấm phím Tab.
Text	Tiêu đề của nút lệnh. Ta có thể quy định phím nóng cho nút lệnh bằng cách đặt dấu "&" trước một ký tự của Text, ví dụ &Quit sẽ được hiển thị là <u>Quit</u> , khi người sử dụng bấm Alt+Q chương trình sẽ kích hoạt nút lệnh Quit.
Visible	Visible = True: hiển thị nút lệnh, Visible = False: ẩn nút lệnh.

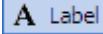
2. Sự kiện

Nút lệnh có một số sự kiện cơ bản sau:

Bài giảng lập trình trực quan

Click	Được kích hoạt khi người dùng kích chuột vào nút lệnh.
Enter	Được kích hoạt khi người dùng chuyển tiêu điểm tới nút lệnh.
Leave	Được kích hoạt khi nút lệnh mất tiêu điểm.
MouseDown	Được kích hoạt khi người dùng đặt chuột vào nút lệnh.
MouseUp	Được kích hoạt khi người dùng đưa chuột ra khỏi nút lệnh.
MouseMove	Được kích hoạt khi người dùng di chuyển chuột trên nút lệnh.
MouseLeave	Được kích hoạt khi người dùng dời chuột ra khỏi nút lệnh.

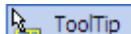
4.2.4. Nhãn – Label

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

Nhãn có một số thuộc tính hay dùng sau:

Name	Tên nhãn, bắt đầu bởi tiếp đầu ngữ lbl
BackColor	Thiết lập màu nền cho nhãn, nếu thiết lập BackColor = Transparent (mục lựa chọn đầu tiên trong tab Web) thì nhãn sẽ có nền giống với nền của Form.
BorderStyle	Thiết lập kiểu đường viền cho nhãn.
Font	Thiết lập kiểu chữ và cỡ chữ cho nhãn.
ForeColor	Thiết lập màu chữ cho nhãn.
Image	Thiết lập ảnh hiển thị trên nhãn.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của nhãn trên Form, ngược lại thì nhận giá trị False.
TabIndex	Thứ tự truy cập của nhãn khi người dùng bấm phím Tab.
Text	Tiêu đề của nhãn.
 TextAlign	Thiết lập chế độ căn chỉnh: trái, phải hoặc giữa của tiêu đề nhãn.
Visible	Hiện hoặc ẩn nhãn.

4.2.5. Dòng mách nước – ToolTip

Dòng mách nước  cho phép hiển thị các thông tin chú thích khi người dùng đưa chuột qua một điều khiển dùng ToolTip. Ví dụ dòng mách nước “Hãy nhập tên truy cập” như hình dưới đây.

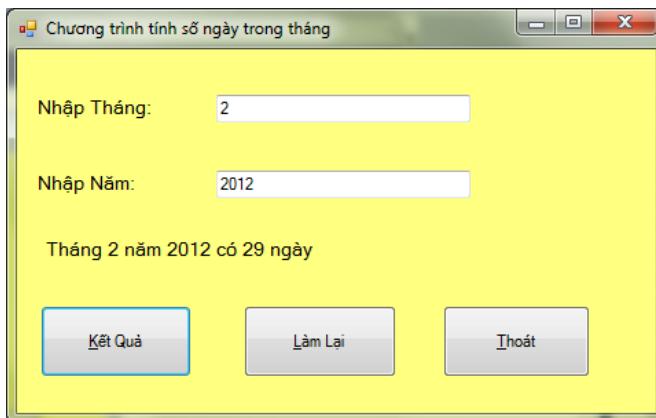


Để tạo dòng mách nước, ta phải kéo một điều khiển ToolTip vào Form (điều khiển ToolTip không được hiển thị ở trên Form mà được hiển thị ở thanh ngang cuối Form). Sau đó kích chuột chọn một điều khiển bất kỳ muốn tạo ToolTip, trong cửa sổ Window Properties gõ nội dung dòng ToolTip tại thuộc tính ToolTip on ToolTip1.

4.2.6. Bài tập

Bài tập 1: (Giáo viên hướng dẫn trên lớp)

Lập chương trình nhập một tháng và một năm dương lịch, tính và đưa ra số ngày của tháng và năm đó?



Yêu cầu: Chỉ được phép nhập số nguyên vào hai hộp văn bản chứa tháng và năm. Tháng phải có giá trị từ 1 đến 12, năm gồm 4 chữ số. Kết quả chỉ được tính khi người dùng nhập đủ cả tháng và năm.

Bài tập 2:

Lập trình giải bài toán nhập 2 số nguyên a, b và tính tổng các số từ a đến b.

Bài giảng lập trình trực quan



Yêu cầu:

- + Tạo giao diện theo Form trên.
- + Tạo dòng ToolTip “Nhập số nguyên” cho 2 hộp văn bản ‘Nhập a’ và ‘Nhập b’.
- + Chỉ cho phép người dùng nhập số vào hai hộp văn bản.
 - + Nút Tổng, kiểm tra người dùng phải nhập dữ liệu cho cả hai số a và b, tính tổng các số từ a đến b nếu $a < b$, hoặc tính tổng các số từ b đến a nếu $b < a$, rồi hiển thị kết quả vào nhãn ở phía dưới.
 - + Nút Làm lại, xóa các dữ liệu cũ ở các điều khiển, sau đó đặt con trỏ vào hộp văn bản Nhập a.
 - + Nút Thoát, thoát khỏi chương trình quay về môi trường soạn thảo.

Bài tập 3:

Nhập số nguyên dương n, tạo n số nguyên ngẫu nhiên có giá trị từ 1 tới 100, và thực hiện các yêu cầu theo giao diện sau:

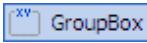


Yêu cầu:

- + Tạo giao diện theo Form trên.
- + Chỉ cho phép người dùng nhập số vào hộp văn bản Nhập n.
- + Nút Nhập, kiểm tra người dùng phải nhập giá trị cho n, sau đó tạo n số ngẫu nhiên và hiển thị các số ngẫu nhiên đó ở nhãn Dãy số.
- + Nút Tính tổng, tính tổng n số ngẫu nhiên và hiển thị kết quả ở nhãn Tổng dãy số.
- + Nút Sắp xếp, sắp xếp n số ngẫu nhiên theo thứ tự giảm dần và hiển thị kết quả ở nhãn Sắp xếp.
- + Nút Làm lại, xóa các dữ liệu cũ ở các điều khiển, sau đó đặt con trỏ vào hộp văn bản Nhập n.
- + Nút Thoát, thoát khỏi chương trình quay về môi trường soạn thảo.

4.3. Một số điều khiển cơ bản khác

4.3.1. Nhóm – GroupBox

Nhóm  có thể chứa các điều khiển khác và tạo thành các vùng làm việc độc lập trên một Form. GroupBox có một số thuộc tính thường dùng sau:

Name	Tên nhóm, bắt đầu bởi tiếp đàu ngũ grb
BackColor	Thiết lập màu nền cho nhóm, nếu BackColor = Transparent thì nhóm sẽ có màu nền giống với màu nền của Form.
BackgroundImage	Thiết lập ảnh nền cho nhóm.
BackgroundImageLayout	Thiết lập chế độ hiển thị ảnh nền của nhóm.
Enabled	Nếu Enabled = False nhóm sẽ không hoạt động.
Font	Xác lập kiểu chữ và cỡ chữ của tiêu đề nhóm.
ForeColor	Xác lập màu chữ của tiêu đề nhóm.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của nhóm trên Form, ngược lại thì nhận giá trị False.
TabIndex	Thứ tự truy cập của nhóm khi người dùng bấm phím Tab.
Text	Thiết lập tiêu đề của nhóm.
Visible	Visible = True: hiển thị nhóm, Visible = False: ẩn nhóm.

Bài giảng lập trình trực quan

4.3.2. Hộp đánh dấu – CheckBox

Hộp đánh dấu  cho phép đồng thời không chọn, chọn một, hoặc chọn nhiều khả năng trong một nhóm các lựa chọn.

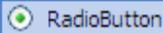
1. Thuộc tính

Name	Tên hộp CheckBox, bắt đầu bởi tiếp đầu ngữ chk
BackColor	Thiết lập màu nền cho hộp CheckBox.
BackgroundImage	Thiết lập ảnh nền cho hộp CheckBox.
Checked	Trả về giá trị của hộp CheckBox ứng với trạng thái của nó khi tương tác với người sử dụng. Checked = True: hộp CheckBox đang được chọn, Checked = False: hộp CheckBox không được chọn.
CheckState	Thiết lập trạng thái cho hộp CheckBox, CheckState = Checked: hộp CheckBox được chọn, CheckState = Unchecked: hộp CheckBox không được chọn.
Enabled	Nếu Enabled = False hộp CheckBox sẽ không hoạt động.
Font	Xác lập kiểu chữ và cỡ chữ của nội dung hộp CheckBox.
ForeColor	Xác lập màu chữ của nội dung hộp CheckBox.
Image	Thiết lập ảnh hiển thị trên hộp CheckBox.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của hộp
TabIndex	Thứ tự truy cập khi người dùng bấm phím Tab.
Text	Thiết lập nội dung của hộp CheckBox.
Visible	Visible = True: hiển thị hộp CheckBox, Visible = False: ẩn hộp

2. Sự kiện

Click	Được kích hoạt khi người dùng kích chuột vào hộp CheckBox.
Enter	Được kích hoạt khi người dùng chuyển tiêu điểm tới hộp CheckBox.
Leave	Được kích hoạt khi hộp CheckBox mất tiêu điểm.
CheckedChanged	Được kích hoạt khi hộp CheckBox thay đổi trạng thái.

4.3.3. Nút tùy chọn – RadioButton

Nút tùy chọn  chỉ cho phép người dùng chọn một khả năng trong một nhóm các lựa chọn.

1. Thuộc tính

Name	Tên nút tùy chọn, bắt đầu bởi tiếp đầu ngữ rdo
BackColor	Thiết lập màu nền cho nút tùy chọn.

Bài giảng lập trình trực quan

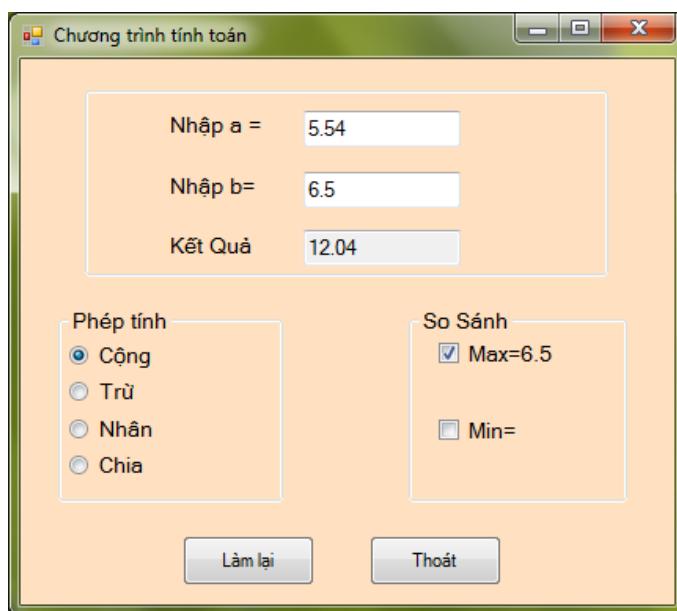
BackgroundImage	Thiết lập ảnh nền cho nút tùy chọn.
Checked	Trả về giá trị của nút tùy chọn khi tương tác với người sử dụng. Checked = True: nút tùy chọn đang được chọn, Checked = False: nút tùy chọn không được chọn.
Enabled	Nếu Enabled = False nút tùy chọn sẽ không hoạt động.
Font	Xác lập kiểu chữ và cỡ chữ của nội dung nút tùy chọn.
ForeColor	Xác lập màu chữ của nội dung nút tùy chọn.
Image	Thiết lập ảnh hiển thị trên nút tùy chọn.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của nút tùy chọn trên Form, ngược lại thì nhận giá trị False.
TabIndex	Thứ tự truy cập khi người dùng bấm phím Tab.
Text	Thiết lập nội dung của nút tùy chọn.
Visible	True: hiển thị nút tùy chọn, False: ẩn nút tùy chọn.

2. Sự kiện

Click	Được kích hoạt khi người dùng click chuột vào nút tùy chọn.
Enter	Được kích hoạt khi người dùng chuyển tiêu điểm tới nút tùy chọn.
Leave	Được kích hoạt khi nút tùy chọn mất tiêu điểm.
CheckedChanged	Được kích hoạt khi nút tùy chọn thay đổi trạng thái.

Bài tập 4: (Giáo viên hướng dẫn trên lớp)

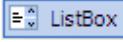
Lập chương trình nhập 2 số a và b, chọn và thực hiện các phép toán theo yêu cầu sau:



Bài giảng lập trình trực quan

- + Chỉ được nhập số cho a và b, không cho phép nhập dữ liệu vào hộp kết quả.
- + Các phép toán chỉ được thực hiện khi người dùng nhập đủ hai dữ liệu cho a và b. Trong phép chia kiểm tra nếu b = 0 thì thông báo “Mẫu = 0” tại hộp Kết quả.
- + Kích chọn phép toán nào thì thực hiện phép toán đó đối với a, b và lưu kết quả vào hộp Kết quả.
- + Nếu chọn hộp đánh dấu Max thì hiển thị “Max = <Giá trị max>” ngược lại chỉ hiển thị “Max”. Thực hiện tương tự cho hộp đánh dấu Min.

4.3.4. Hộp danh sách – *ListBox*

Hộp *ListBox*  là một tập hợp các chuỗi ký tự được trình bày dưới dạng liệt kê thành từng dòng trong một khung hình chữ nhật. Ta có thể chọn, bổ sung hoặc xoá một giá trị trong hộp danh sách.

Khi hiển thị dữ liệu, nếu chiều ngang của Listbox nhỏ hơn độ dài các phần tử thì một phần dữ liệu sẽ bị che khuất, còn nếu số phần tử của Listbox vượt quá chiều dài của Listbox thì Listbox tự động cung cấp thanh cuộn dọc để cuộn tới các phần tử phía dưới.

1. Thuộc tính

Name	Tên hộp <i>ListBox</i> , bắt đầu bởi tiếp đầu ngữ <i>lst</i>
BackColor	Thiết lập màu nền cho hộp danh sách.
DataSource	Thiết lập nguồn dữ liệu cho <i>ListBox</i>
Enabled	Nếu <i>Enabled</i> = <i>False</i> hộp danh sách sẽ không hoạt động.
Font	Xác lập kiểu chữ và cỡ chữ cho hộp danh sách.
ForeColor	Xác lập màu chữ cho hộp danh sách.
MultiColumn	<i>MultiColumn</i> = <i>True</i> : cho phép hiển thị dữ liệu theo nhiều cột. <i>MultiColumn</i> = <i>False</i> : chỉ cho phép hiển thị dữ liệu theo 1 cột.
ColumnWidth	Thiết lập độ rộng cho mỗi cột trong <i>ListBox</i> .
Items	Khởi tạo giá trị cho các phần tử của hộp danh sách trong thời gian thiết kế. Khi chọn thuộc tính <i>Items</i> trong cửa sổ <i>Properties</i> , C# mở ra một hộp soạn thảo cho phép người lập trình gõ vào giá trị các phần tử. Mỗi phần tử được đặt trên một dòng riêng biệt, để xuống dòng nhấn <i>Enter</i> .
Items.Count	Trả về tổng số phần tử của danh sách trong thời gian thi hành.
Items(n)	Trả về nội dung phần tử thứ n của danh sách trong thời gian thi hành.

Bài giảng lập trình trực quan

SelectedItem	Tương tự như thuộc tính Items(n), nhưng chỉ có thể trả về nội dung của phần tử hiện hành đang được chọn.
Locked	Locked = True: khóa không cho phép dịch chuyển vị trí của hộp danh sách trên Form, ngược lại thì nhận giá trị False.
SelectedIndex	Trả về số thứ tự của phần tử đang được chọn trong danh sách, phần tử đầu tiên có SelectedIndex = 0, nếu không có phần tử nào được chọn thì SelectedIndex = -1
SelectionMode	Quy định chế độ lựa chọn các phần tử trong hộp danh sách khi thực thi chương trình. SelectionMode có 4 giá trị: None - không cho phép lựa chọn các phần tử, One - cho phép chọn một phần tử, MultiSimple - cho phép lựa chọn nhiều phần tử riêng biệt, MultiExtended - cho phép chọn một khối các phần tử liền nhau.
SelectedItems	Trả về tập các phần tử đang được chọn.
Sorted	Nếu Sorted = True thì các phần tử trong danh sách được sắp xếp theo
TabIndex	Thứ tự truy cập khi người dùng bấm phím Tab.
Visible	True: hiển thị hộp danh sách, False: ẩn hộp danh sách.

2. Sự kiện

Click	Được kích hoạt khi người dùng kích chuột vào hộp danh sách.
DoubleClick	Được kích hoạt khi người dùng kích đúp chuột vào hộp danh sách.
Enter	Được kích hoạt khi người dùng chuyển tiêu điểm tới hộp danh sách.
Leave	Được kích hoạt khi hộp danh sách mất tiêu điểm.
SelectedIndex_Changed	Được kích hoạt khi người dùng thay đổi trạng thái lựa chọn các dòng dữ liệu trong hộp listbox.

3. Phương thức

Add: dùng để bổ sung một phần tử cho hộp danh sách trong thời gian thi hành và thường được viết trong thủ tục Form_Load. Cú pháp của phương thức này là:

ListName.Items.Add(Item)

Trong đó ListName là tên của hộp danh sách, Item là nội dung của phần tử ta muốn thêm vào hộp danh sách.

Ví dụ, bổ sung phần tử có giá trị “Ha Noi” vào hộp danh sách lstQue ta thực hiện như sau:

IstQue.Items.Add(“Ha Noi”)

Remove: dùng để loại bỏ một phần tử của hộp danh sách theo nội dung trong thời gian thi hành. Cú pháp của phương thức này là:

Bài giảng lập trình trực quan

ListName.Items.Remove(Item)

Ví dụ:, xóa phần tử có giá trị “Ha Noi” trong hộp danh sách lstQue ta viết như sau:

```
lstQue.Items.Remove("Ha Noi")
```

RemoveAt: dùng để loại bỏ một phần tử của hộp danh sách theo chỉ số trong thời gian thi hành. Cú pháp của phương thức này là:

ListName.Items.RemoveAt(Index)

Ví dụ, xóa phần tử ở vị trí 1 trong hộp danh sách lstQue ta viết như sau:

```
lstQue.Items.RemoveAt(1)
```

Clear: dùng để loại bỏ tất cả các phần tử của hộp danh sách trong thời gian thi hành. Cú pháp của phương thức này là:

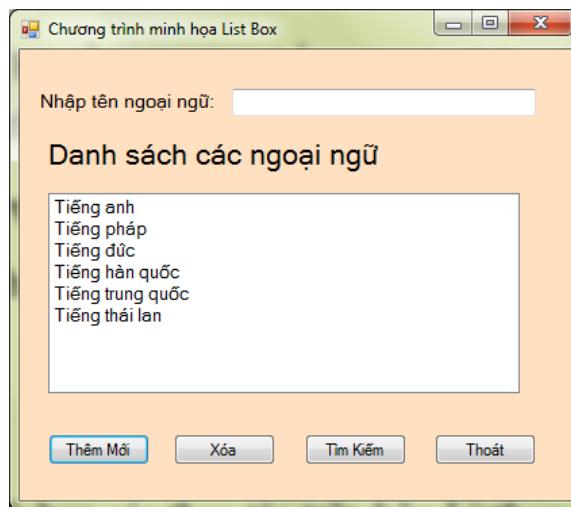
ListName.Items.Clear()

Ví dụ, xóa tất cả các phần tử trong hộp danh sách lstQue ta viết như sau:

```
lstQue.Items.Clear()
```

Bài tập 5: (Giáo viên hướng dẫn trên lớp)

Viết chương trình minh họa các thao tác trên hộp Listbox theo yêu cầu sau:



Nhập một tên ngoại ngữ vào hộp văn bản *Nhập tên ngoại ngữ*, chọn nút *Thêm mới* để thêm ngoại ngữ đó vào hộp danh sách, chọn nút *Tìm kiếm* để xem ngoại ngữ đó đã có trong hộp danh sách chưa?

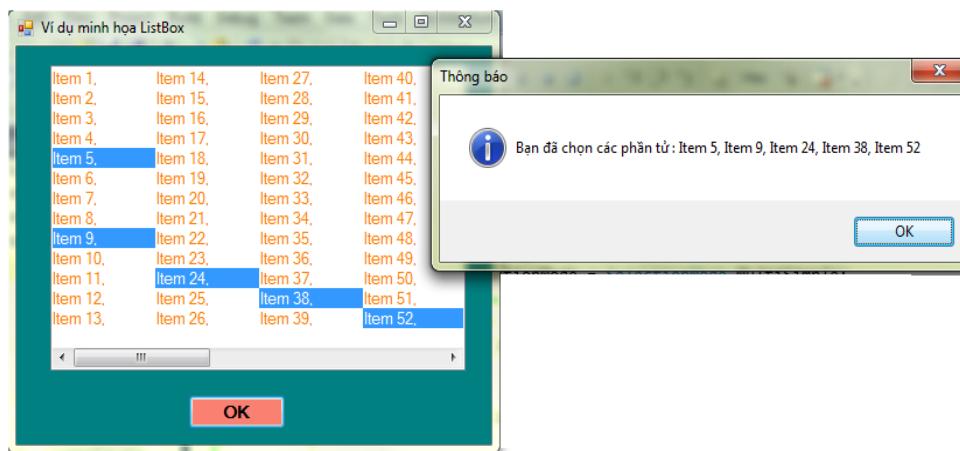
Bài giảng lập trình trực quan

Chọn nút **Xóa** để xoá một tên ngoại ngữ bất kỳ được chọn từ hộp danh sách. Chọn nút **Thoát** để thoát khỏi chương trình.

Bài tập 6:

Viết chương trình minh họa các thao tác trên hộp Listbox theo yêu cầu sau:

Nhập vào hộp danh sách 100 phần tử từ Items 1 đến Items 100, dữ liệu được hiển thị thành 4 cột trong một trang màn hình và người dùng có thể lựa chọn một hoặc nhiều phần tử cùng một lúc.



Hướng dẫn:

Vào Microsoft Visual Studio 2010 tạo một dự án mới, đặt một hộp danh sách *lstDanhSach* và một nút lệnh *btnOK* với tiêu đề là *OK* vào form Form1.

Viết mã lệnh:

Khai báo một biến toàn cục dùng để chứa chuỗi thông báo như sau:

```
string thongbao = "";
```

Trong sự kiện Load của Form1 ta viết lệnh như sau:

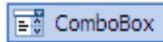
```
private void Form1_Load(object sender, EventArgs e)
{
    int i;
    //Cho phép hiển thị trên nhiều cột
    lstDanhSach.MultiColumn = true;
    //Hiển thị trên 4 cột
    lstDanhSach.ColumnWidth = lstDanhSach.Width / 4;
    //Cho phép chọn đồng thời nhiều phần tử
    lstDanhSach.SelectionMode = SelectionMode.MultiSimple;
    //Add dữ liệu vào hộp danh sách
    for (i = 1; i <= 100; i++)
        lstDanhSach.Items.Add("Item " + i.ToString());
}
```

Bài giảng lập trình trực quan

Trong sự kiện Click của button *btnOK* ta viết mã lệnh như sau:

```
private void btnOK_Click(object sender, EventArgs e)
{
    foreach (string item in lstDanhSach.SelectedItems)
    {
        thongbao = thongbao + item + ", ";
    }
    //Xóa dấu phẩy và dấu cách thừa ở cuối chuỗi
    thongbao = thongbao.Remove(thongbao.Length - 2, 2);
    MessageBox.Show("Bạn đã chọn các phần tử : " + thongbao,
                    "Thông báo", MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
}
```

4.3.5. Hộp lựa chọn – ComboBox

Hộp ComboBox  cho phép lưu trữ và lựa chọn một mục dữ liệu trong một hộp danh sách thả xuống.

1. Thuộc tính

Name	Tên hộp ComboBox, bắt đầu bởi tiếp đầu ngữ cbo
BackColor	Thiết lập màu nền cho hộp Combo.
DataSource	Thiết lập nguồn dữ liệu cho Combo.
DropDownStyle	DropDown gồm một hộp văn bản cho phép người sử dụng có thể nhập dữ liệu, kế bên có một mũi tên, nhấn vào đó sẽ mở ra một danh sách các mục dữ liệu cho phép người dùng chọn lựa. Simple luôn hiển thị sẵn danh sách các mục dữ liệu bên dưới hộp văn bản và cho phép người sử dụng có thể nhập dữ liệu vào hộp văn bản. DropDownList tương tự như DropDown nhưng người sử dụng chỉ có thể chọn các phần tử từ danh sách, khi gõ một ký tự vào hộp văn bản thì danh sách sẽ cuộn đến các phần tử được bắt đầu bởi ký tự đó.
Enabled	Nếu Enabled = False hộp Combo sẽ không hoạt động.
Font	Xác lập kiểu chữ và cỡ chữ cho hộp Combo.
ForeColor	Xác lập màu chữ cho hộp Combo.
Items	Khởi tạo giá trị các phần tử của hộp Combo trong thời gian thiết kế.
Items.Count	Trả về tổng số phần tử của hộp Combo trong thời gian thi hành.
Items[n]	Trả về nội dung phần tử thứ n của hộp Combo trong thời gian thi hành
SelectedItem hoặc Text	Trả về nội dung của phần tử hiện hành đang được chọn.

Bài giảng lập trình trực quan

SelectedIndex	Trả về số thứ tự của phần tử đang được chọn trong hộp Combo, phần tử đầu tiên có SelectedIndex = 0, nếu không có phần tử nào được chọn thì SelectedIndex = -1
Locked	True: không cho phép dịch chuyển vị trí của hộp Combo trên Form
Sorted	True: các phần tử trong danh sách được sắp xếp theo thứ tự ABC.
TabIndex	Thứ tự truy cập khi người dùng bấm phím Tab.
Visible	True: hiển thị hộp Combo, False: ẩn hộp Combo.

2. Sự kiện

Click	Được kích hoạt khi người dùng kích chuột vào hộp Combo.
DoubleClick	Được kích hoạt khi người dùng kích đúp chuột vào hộp Combo.
Enter	Được kích hoạt khi người dùng chuyển tiêu điểm tới hộp Combo.
Leave	Được kích hoạt khi hộp Combo mất tiêu điểm.
SelectedIndex_Changed	Được kích hoạt khi người dùng thay đổi trạng thái lựa chọn các dòng dữ liệu trong hộp văn bản.
TextChanged	Được kích hoạt khi người dùng nhập, sửa, xóa dữ liệu tại vùng văn bản của hộp Combo hoặc khi ta thay đổi thuộc tính Text của hộp Combo từ mã lệnh.
DropDown	Chỉ xảy ra đối với hộp Combo DropDown và DropDownList, sự kiện này được gọi ngay sau khi người dùng nhập mũi tên để thả hộp danh sách xuống (phím tắt Alt+). Vì thế sự kiện này chủ yếu được sử dụng để nhập dữ liệu cho các phần tử của hộp Combo.

3. Phương thức

Add: dùng để bổ sung một phần tử cho hộp Combo trong thời gian thi hành và thường được viết trong thủ tục Form_Load. Cú pháp của phương thức này là:

ComboName.Items.Add(Item)

Trong đó ComboName là tên của hộp Combo, Item là nội dung của phần tử ta muốn thêm vào hộp Combo. Ví dụ, bổ sung phần tử có giá trị “Ha Noi” vào hộp Combo cboQue ta thực hiện như sau:

```
cboQue.Items.Add("Ha Noi")
```

Remove: dùng để loại bỏ một phần tử của danh sách theo nội dung trong thời gian thi hành. Cú pháp của phương thức này là:

ComboName.Items.Remove(Item)

Ví dụ, xóa phần tử có giá trị “Ha Noi” trong hộp Combo cboQue ta viết như sau:

Bài giảng lập trình trực quan

cboQue.Items.Remove("Ha Noi")

RemoveAt: dùng để loại bỏ một phần tử của hộp Combo theo chỉ số trong thời gian thi hành. Cú pháp của phương thức này là:

ComboName.Items.RemoveAt(Index)

Ví dụ, xóa phần tử ở vị trí 1 trong hộp Combo cboQue ta viết như sau:

cboQue.Items.RemoveAt(1)

Clear: dùng để loại bỏ tất cả các phần tử của hộp Combo trong thời gian thi hành. Cú pháp của phương thức này là:

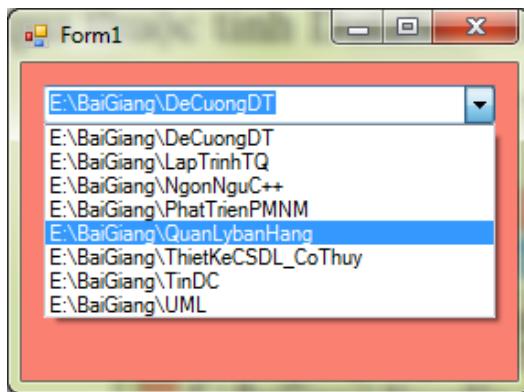
ComboName.Items.Clear()

Ví dụ, xóa tất cả các phần tử trong hộp Combo cboQue ta viết như sau:

cboQue.Items.Clear()

Bài tập 7

Lấy danh sách các thư mục có trong thư mục “E:\Baigiang” lưu vào hộp Combo thông qua thuộc tính DataSource.



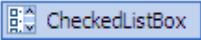
Hướng dẫn

Vào Microsoft Visual Studio 2010 tạo một dự án mới, đặt một hộp combo *cboThumuc* vào Form1.

Viết code: Mở sự kiện Load của Form1 ra và viết vào sự kiện đó mã lệnh sau:

```
private void Form1_Load(object sender, EventArgs e)
{
    String[] foder;
    foder = System.IO.Directory.GetDirectories("E:\\\\BaiGiang");
    cboThumuc.DataSource = foder ;
}
```

4.3.6. Điều khiển CheckedListBox

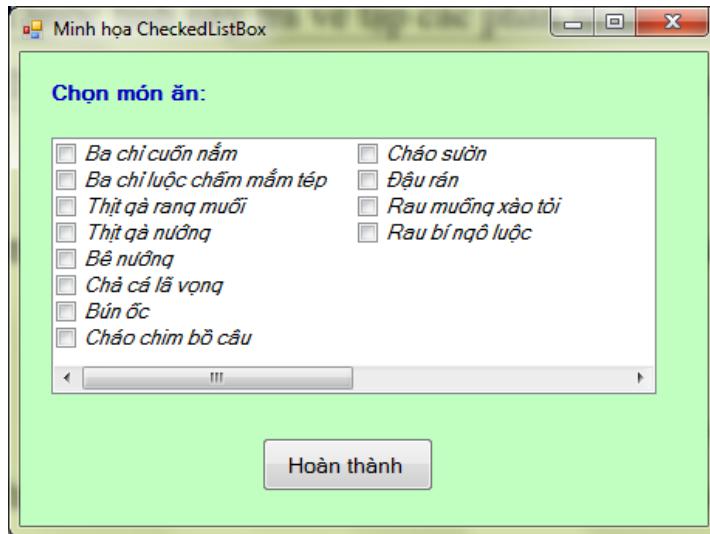
Điều khiển CheckedListBox  cho phép lưu trữ và hiển thị các mục dữ liệu theo dòng và có một hộp CheckBox ở đầu dòng.

Điều khiển CheckedListBox có tiếp đầu ngữ là clb và có các thuộc tính, sự kiện, phương thức tương tự như điều khiển ListBox. Ngoài ra nó có thêm một số thuộc tính và sự kiện khác như sau:

CheckedItems	Thuộc tính này trả về tập các phần tử được Check.
ItemCheck	Được kích hoạt khi người dùng kích đúp chuột vào một lựa chọn.

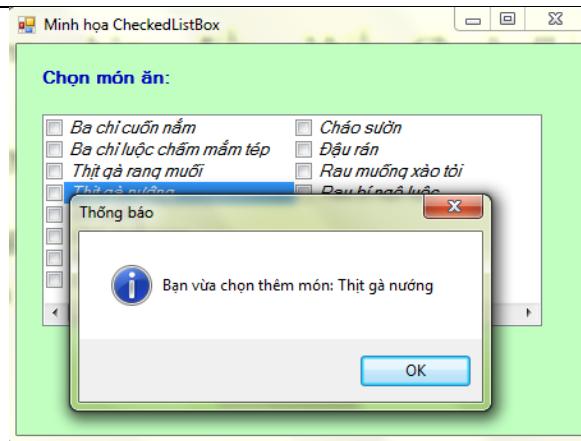
Bài tập 8:

Tạo một chương trình cho phép chọn các món ăn như sau:

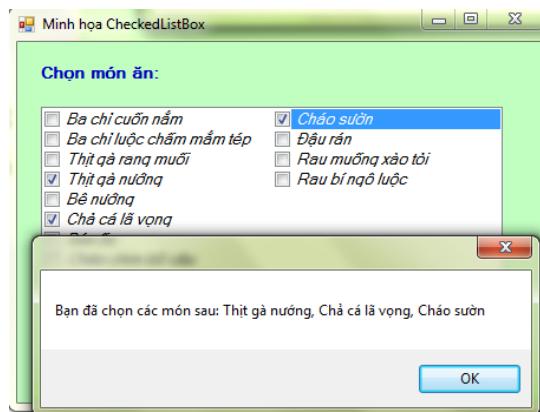


Đặt lên form hộp điều khiển CheckedListBox *clbMonAn* và nút lệnh *btnHoanThanh*. Viết chương trình hiển thị danh sách các món ăn được chia thành 2 cột vào trong hộp *clbMonAn*.

Khi người dùng đánh dấu vào hộp CheckBox của từng phần tử, sẽ xuất hiện hộp thông báo tên món ăn tương ứng với phần tử đó, như hình minh họa sau:



Khi người dùng kích chọn nút *Hoàn thành* sẽ xuất hiện hộp thoại thông báo tên tất cả các món ăn người dùng đã chọn.



Hướng dẫn

Mở sự kiện Load của Form1 ra và viết mã lệnh như sau:

```
private void Form1_Load(object sender, EventArgs e)
{
    //Cho phép hiển thị trên nhiều cột
    clbMonAn.MultiColumn = true;
    //Cho phép hiển thị trên 2 cột
    clbMonAn.ColumnWidth = clbMonAn.Width / 2;
    //Thêm các phần tử vào CheckedListBox
    clbMonAn.Items.Add("Bà chỉ cuốn nấm");
    clbMonAn.Items.Add("Bà chỉ luộc chấm mắm tép");
    clbMonAn.Items.Add("Thịt gà rang muối");
    clbMonAn.Items.Add("Thịt gà nướng");
    clbMonAn.Items.Add("Bé nướng");
    clbMonAn.Items.Add("Chả cá lã vọng");
    clbMonAn.Items.Add("Bún ốc");
    clbMonAn.Items.Add("Cháo chim bồ câu");
    clbMonAn.Items.Add("Cháo sườn");
    clbMonAn.Items.Add("Đậu rán");
    clbMonAn.Items.Add("Rau muống xào tỏi");
    clbMonAn.Items.Add("Rau bí ngô luộc");
}
```

Bài giảng lập trình trực quan

Mở sự kiện ItemCheck của CheckedListBox *clbMonAn* và viết lệnh:

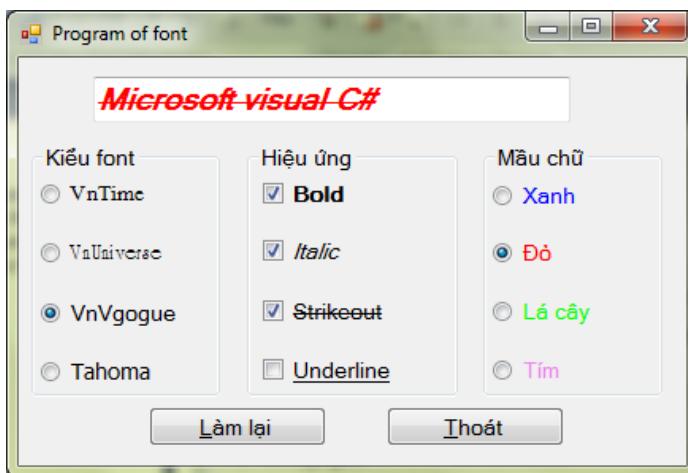
```
private void clbMonAn_ItemCheck(object sender, ItemCheckEventArgs e)
{
    if (e.NewValue == CheckState.Checked)
        MessageBox.Show("Bạn vừa chọn thêm món: " + clbMonAn.SelectedItem,
                        "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

Mở sự kiện Click của nút lệnh *btnHoanThanh* ra và viết mã lệnh như sau:

```
private void btnHoanThanh_Click(object sender, EventArgs e)
{
    string str = "";
    foreach (string item in clbMonAn.CheckedItems)
        str = str + item + ", ";
    //Xóa dấu phẩy và dấu cách thừa cuối chuỗi
    str = str.Remove(str.Length - 2, 2);
    MessageBox.Show("Bạn đã chọn các món sau: " + str);
}
```

Bài tập 9:

Lập chương trình thực hiện các công việc thay đổi Font chữ theo giao diện dưới đây:



Gợi ý: Để thay đổi kiểu Font chữ cho điều khiển ta thực hiện theo cú pháp sau:

```
<TênĐiều khiển>.Font = New Font("TênFont", Cỡchữ)
```

Để thay đổi hiệu ứng của font chữ ta thực hiện theo cú pháp sau:

```
<TênĐiều khiển>.Font = New Font(<TênĐiều khiển>.Font, FontStyle.HiệuỨng)
```

Trong đó Hiệu ứng có thể nhận các giá trị: Bold, Regular, Italic, UnderLine, Strikeout. Nếu tạo font với nhiều hiệu ứng thì ta dùng toán tử OR để kết hợp chúng với nhau.

Để thay đổi màu chữ ta thực hiện theo cú pháp sau:

```
<TênĐiều khiển>.ForeColor = Color.Màu
```

Bài giảng lập trình trực quan

Bài tập 10:

Lập chương trình ghép tên nước và tên thành phố theo giao diện và yêu cầu dưới đây:



Viết phương thức **EmptyOption()** bỏ chọn tất cả các RadioButton tên thành phố.

Khi kích chọn vào một nước, giả sử France thì xuất hiện dòng thông báo: “Hãy chọn thành phố cho France” và gọi phương thức **EmptyOption**.

Khi kích chọn một thành phố, nếu đúng là thành phố của tên nước đã chọn thì xuất hiện dòng thông báo, ví dụ: “Chúc mừng bạn, thủ đô của France là Paris”, ngược lại thông báo, ví dụ: “Bạn sai rồi, thủ đô của France không phải là London”

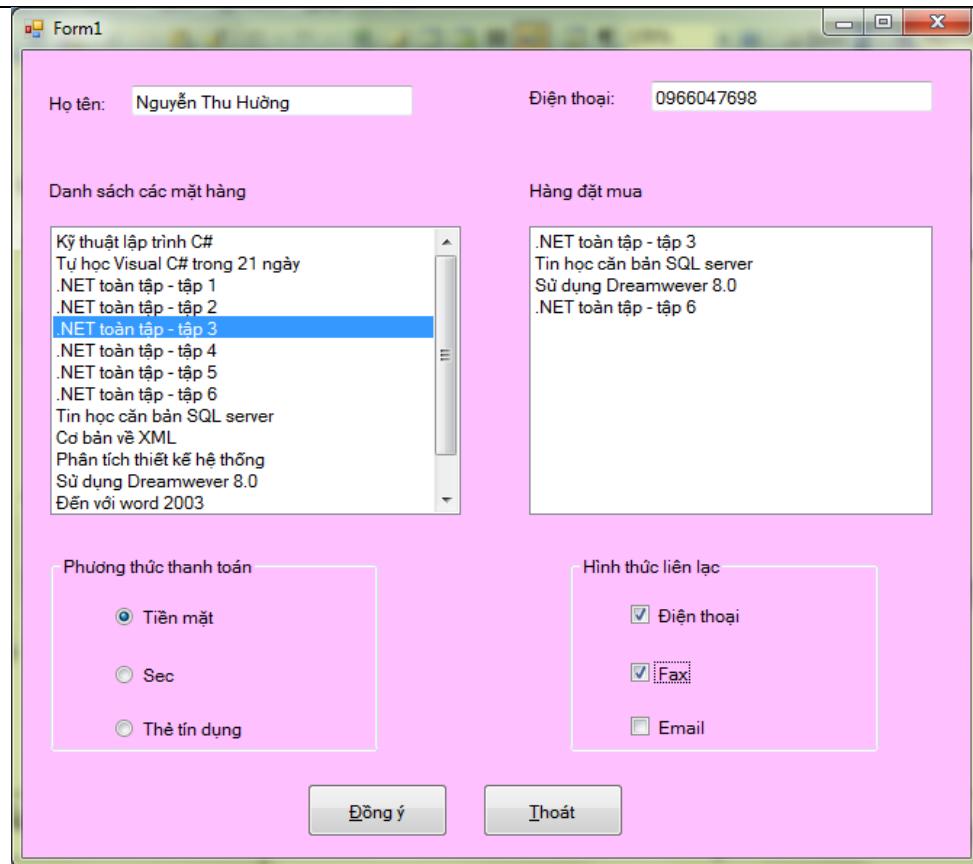
Bài tập 11:

Lập chương trình thực hiện các công việc theo giao diện và yêu cầu dưới đây:

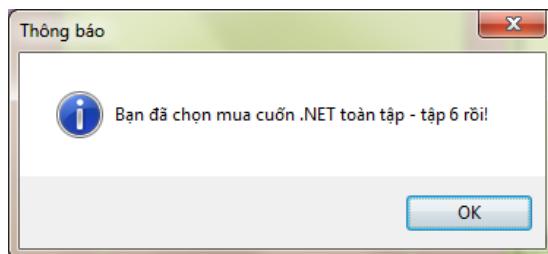
Chương trình có một Form bán hàng trực tuyến, danh sách các mặt hàng được hiển thị sẵn trong hộp Listbox hoặc CheckedListBox “Danh sách các mặt hàng”.

Để mua hàng người dùng kích đúp vào mặt hàng cần mua trong “Danh sách các mặt hàng”, mặt hàng được chọn sẽ được hiển thị vào trong “Hàng đặt mua”.

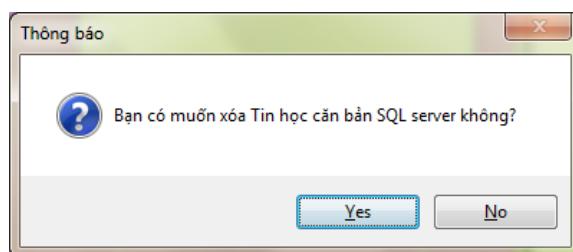
Bài giảng lập trình trực quan



Chú ý: Khi mua hàng phải kiểm tra nếu mặt hàng này đã được mua thì dùng hộp thoại thông báo đã chọn mặt hàng đó và không được mua mặt hàng đó nữa.

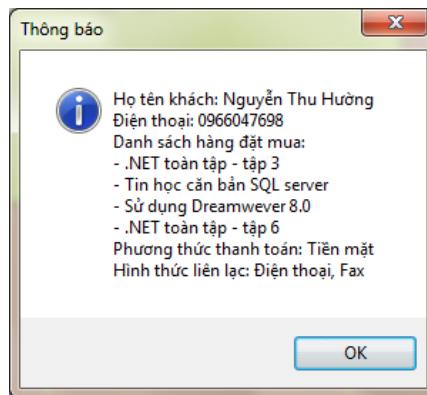


Người dùng có thể xoá mặt hàng trong số các mặt hàng đã chọn bằng cách kích đúp vào mặt hàng cần xoá, trước khi xoá phải hỏi lại người dùng có muốn xoá hay không?



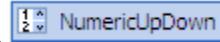
Bài giảng lập trình trực quan

Khi kích chuột vào nút “Đồng ý” kiểm tra người dùng phải nhập đầy đủ thông tin và hiện thông báo gồm các thông tin: Tên khách, Điện thoại, Danh sách các mặt hàng đã mua, Phương thức thanh toán và Hình thức liên lạc.



Chú ý: Khi thực hiện chương trình, để các dòng thông báo hay các kết quả được hiển thị trên nhiều dòng khác nhau thì khi viết mã lệnh ta có thể dùng hằng ký tự điều khiển `\n`

4.3.7. Điều khiển NumericUpDown

Điều khiển NumericUpDown  cho phép người dùng lựa chọn một giá trị số trong một khoảng giá trị với một bước nhảy xác định.

1. Thuộc tính

Name	Tên điều khiển NumericUpDown, bắt đầu bởi tiếp đầu ngữ nud
Increment	Giá trị của bước nhảy.
Maximum	Cận trên của khoảng giá trị.
Minimum	Cận dưới của khoảng giá trị.
Value	Giá trị hiện tại của điều khiển NumericUpDown.

2. Sự kiện

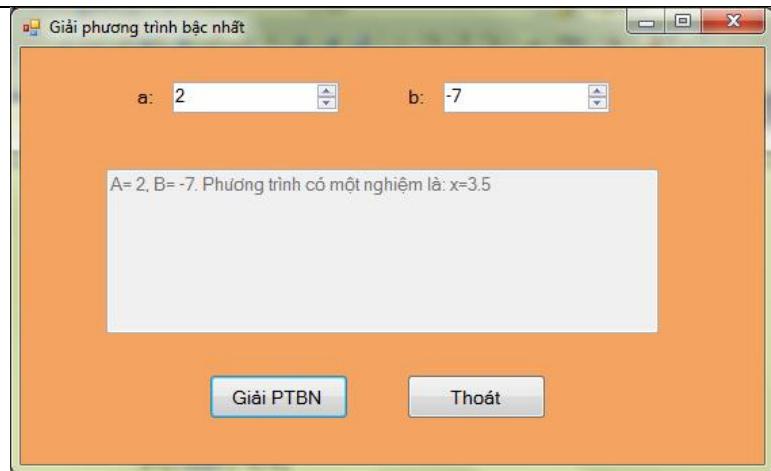
ValueChanged	Được kích hoạt khi người dùng thay đổi giá trị của điều khiển.
---------------------	--

Bài tập 12:

Dùng điều khiển NumericUpDown để giải phương trình bậc nhất:

$$ax + b = 0 \quad (a, b \text{ có giá trị nguyên trong đoạn } [-100, 100])$$

Bài giảng lập trình trực quan



Hướng dẫn:

Vào Microsoft Visual Studio 2010 tạo một dự án mới, đặt tên form là *frmPTBH* và đặt vào form các điều khiển sau:

- Hai điều khiển NumericUpDown có tên *nudA*, *nudB*. Các thuộc tính: *Maximum* = 100, *Minimum* = -100, *Increment* = 1.
- Một hộp Textbox có tên *txtKetQua*. Các thuộc tính: *Enable* = False. Hộp kết quả có thuộc tính *Multiline* = True.

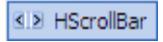
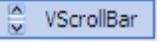
Hai nút lệnh *btnGiai*, *btnThoat* và các nhãn Label...

Viết mã lệnh:

Mở sự kiện Click của nút lệnh *btnGiai* viết mã lệnh như sau:

```
private void btnGiai_Click(object sender, EventArgs e)
{
    int a, b;
    float x;
    string str="A= " + nudA.Value.ToString() + ", B=" +
+nudB.Value.ToString();
    a = Convert.ToInt16(nudA.Value);
    b = Convert.ToInt16(nudB.Value);
    if (a == 0)
    {
        if (b == 0)
            str = str + ". Phương trình vô số nghiệm";
        else
            str = str + ". Phương trình vô nghiệm";
    }
    else
    {
        x=(float)-b/a;
        str = str + ". Phương trình có một nghiệm là: x=" + x.ToString();
    }
    txtKetQua.Text=str;
}
```

4.3.8. Thanh cuộn HscrollBar và VscrollBar

Thanh cuộn ngang HScrollBar  và thanh cuộn dọc VScrollBar  cho phép người dùng lựa chọn một giá trị số trong một khoảng giá trị xác định.

1. Thuộc tính

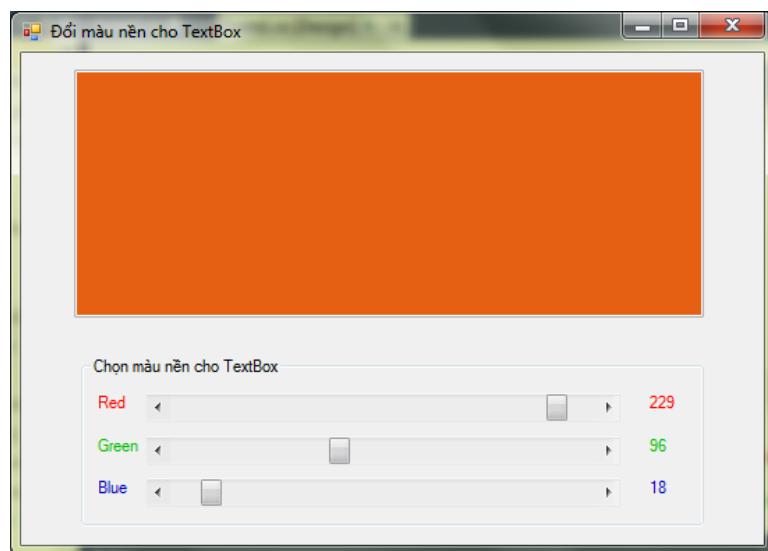
Name	Tên thanh cuộn, bắt đầu bởi tiếp đầu ngữ hsb và vsb .
Minimum	Số nguyên xác định giá trị nhỏ nhất cho thanh cuộn.
Maximum	Số nguyên xác định giá trị lớn nhất cho thanh cuộn.
Value	Cho biết giá trị hiện thời của thanh cuộn.
LargeChange	Chỉ ra mức độ thay đổi của thuộc tính Value khi người dùng nhấn chuột trên thanh cuộn.
SmallChange	Chỉ ra mức độ thay đổi của thuộc tính Value khi người dùng nhấn chuột vào các mũi tên trên thanh cuộn (giá trị mặc định = 1). <=LargeChange

2. Sự kiện

ValueChanged	Được kích hoạt khi người dùng thay đổi giá trị của thanh cuộn.
Scroll	Xảy ra khi người dùng kéo rê chuột hoặc kích chuột vào các mũi tên trên thanh cuộn.

Bài tập 13:

Dùng thanh cuộn HScrollBar để thay đổi màu nền cho TextBox như giao diện sau:



4.3.9. Điều khiển Timer

Điều khiển định thời gian Timer  cho phép thực thi lại một hành động sau một khoảng thời gian xác định.

Khi ta đưa điều khiển Timer vào Form nó không xuất hiện trên Form mà xuất hiện như một biểu tượng ở trên một khay đặt ở cuối cửa sổ thiết kế. Khi chạy chương trình điều khiển Timer cũng không xuất hiện.

1. Thuộc tính

Name	Tên điều khiển Timer, bắt đầu bởi tiếp đầu ngữ tmr
Interval	= n là chu kỳ thực hiện sự kiện Tick của điều khiển Timer. n được tính bằng mili giây và có giá trị >0 (1s=1000ms)
Enabled	Enabled = True: cho phép điều khiển Timer hoạt động, Enabled = False: không cho phép điều khiển Timer hoạt động.

2. Sự kiện

Tick	Sự kiện này được kích hoạt sau mỗi chu kỳ Interval.
-------------	---

3. Phương thức

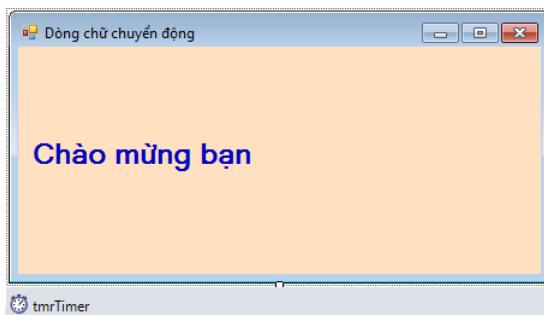
Start: kích hoạt điều khiển Timer, phương thức này tương đương với thuộc tính Enabled = True. Cú pháp của phương thức này là:

TimerName.Start()

Stop: dừng điều khiển Timer, phương thức này tương đương với thuộc tính Enabled = False. Cú pháp của phương thức này là: **TimerName.Stop()**

Bài tập 14:

Viết chương trình tạo dòng chữ “Chào mừng bạn” chuyển động từ trái qua phải màn hình, khi gặp mép màn hình thì chuyển động ngược lại từ phải qua trái. Cứ lặp lại như vậy cho đến khi kết thúc chương trình.



Bài giảng lập trình trực quan

Vào Microsoft Visual Studio 2010 tạo một dự án mới và thiết lập các thuộc tính của các điều khiển như sau:

Điều khiển	Name	Text
Form1	frmVDTimer	Dòng chữ chuyển động
Label1	lblMove	Chào mừng bạn
Timer	tmrTimer	Enabled = True và Interval = 200

Viết Code:

Đặt biến toàn cục:

```
bool hp; // hp=true chuyển động sang phải, hp=false chuyển động sang trái.
```

Mở sự kiện Load của Form frmVDTimer viết mã lệnh sau:

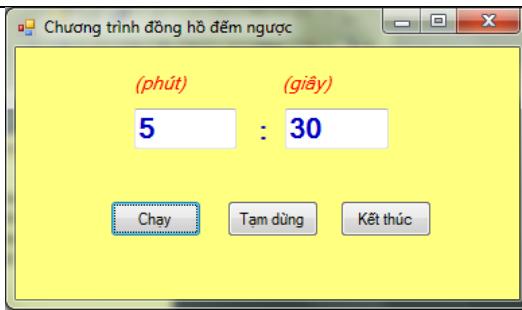
```
private void frmVDTimer_Load(object sender, EventArgs e)
{
    hp = true;
    tmrTimer.Start();
}
```

Mở sự kiện Tick của điều khiển Timer tmrTimer viết mã lệnh sau:

```
private void tmrTimer_Tick(object sender, EventArgs e)
{
    if (hp)
    {
        if ((lblMove.Left + lblMove.Width) < this.Width)
            lblMove.Left = lblMove.Left + 10;
        else
            hp = false;
    }
    else
    {
        if (lblMove.Left > 0)
            lblMove.Left = lblMove.Left - 10;
        else
            hp = true;
    }
}
```

Bài tập 15:

Viết chương trình đồng hồ đếm ngược theo giao diện sau:



Yêu cầu:

- + Nút chạy: sẽ kích hoạt đồng hồ bắt đầu đếm.
- + Nút tạm dừng: sẽ tạm dừng đếm lại. Nếu muốn đếm tiếp kích vào nút chạy.
- + Nút kết thúc: Sẽ làm cho đồng hồ về 00:00.

4.3.10. Điều khiển RichTextBox

Điều khiển RichTextBox  cho phép hiển thị được văn bản có định dạng như một tệp Word, tức là có thể hiển thị màu sắc, loại Font, cỡ chữ... khác nhau trong từng đoạn văn bản.

Điều khiển RichTextBox có thể tạo và lưu trữ các tệp văn bản có phần mở rộng .rtf. Để tạo một tệp có phần mở rộng .rtf ta có thể sử dụng trình soạn thảo Word và lưu tệp – Save As theo định dạng *Rich TextFormat (*.rtf)*

Tên điều khiển RichTextBox có tiếp đầu ngữ **rtb** và nó có 2 phương thức cơ bản sau:

LoadFile: nạp nội dung một tệp .rtf vào RichTextBox. Ví dụ nạp nội dung tệp *Bai1.rtf* trong ổ D vào hộp RichTextBox *rtbBai1* ta thực hiện như sau:

```
rtbBai1.LoadFile("D:\Bai1.rtf", RichTextBoxStreamType.RichText)
```

SaveFile: lưu nội dung trong hộp RichTextBox vào một tệp có phần mở rộng .rtf. Ví dụ lưu nội dung của hộp RichTextBox *rtbBai1* vào tệp *Bai2.rtf* trong ổ D ta thực hiện như sau: (*nếu tệp Bai2.rtf chưa tồn tại thì sẽ được tạo mới*)

```
rtbBai1.SaveFile("D:\Bai2.rtf", RichTextBoxStreamType.RichText)
```

CHƯƠNG 5: CÁC HỘP HỘI THOẠI THÔNG DỤNG

Chúng ta thấy trong hầu hết các ứng dụng của Windows đều có các hộp thoại Open để mở tập tin, Save để lưu các tập tin hoặc các hộp thoại Color để chọn màu, Font để chọn phông chữ... các hộp thoại này gọi là các hộp thoại thông dụng - Common Dialog.

Để sử dụng các hộp thoại thông dụng ta dùng các lớp đối tượng .NET trong thư viện **System.IO** bao gồm 5 lớp tương ứng với 5 hộp hội thoại cơ bản như sau:

Hộp thoại	Lớp đối tượng
Mở tập tin – Open File	OpenFileDialog()
Lưu tập tin – Save File	SaveFileDialog()
Chọn màu – Color	ColorDialog()
Chọn phông – Font	FontDialog()
In ấn – Print	PrintDialog()

5.1. Hộp hội thoại Open File

Các thuộc tính và phương thức quan trọng của hộp hội thoại OpenFileDialog:

Thuộc tính	Chức năng
Name	Tên điều khiển OpenFileDialog. Tiếp đầu ngữ của tên điều khiển là odlg
FileName	Cung cấp tên và đường dẫn của tập tin đã chọn.
Filter	Xác định danh sách các bộ lọc tập tin mà hộp hội thoại sẽ hiển thị, ví dụ: “Text *.txt Icons *.ico>All files *.*” (không được chia dấu cách)
FilterIndex	Chỉ ra bộ lọc tập tin mặc định, giả sử có 3 bộ lọc (*.com), (*.exe) và (*.ico) nếu FilterIndex = 2 thì hộp thoại sẽ hiển thị sẵn bộ lọc (*.exe)
InitialDirectory	Xác định thư mục mặc định cho hộp hội thoại khi vừa được gọi.
Multiselect	Multiselect = True: cho phép người dùng chọn đồng thời nhiều file.
FileNames	Cung cấp tên và đường dẫn của các tập tin đã chọn.
Title	Xác định tiêu đề của hộp hội thoại.
OpenFile	Mở nội dung File đã được chọn (ReadOnly).

Bài tập 16:

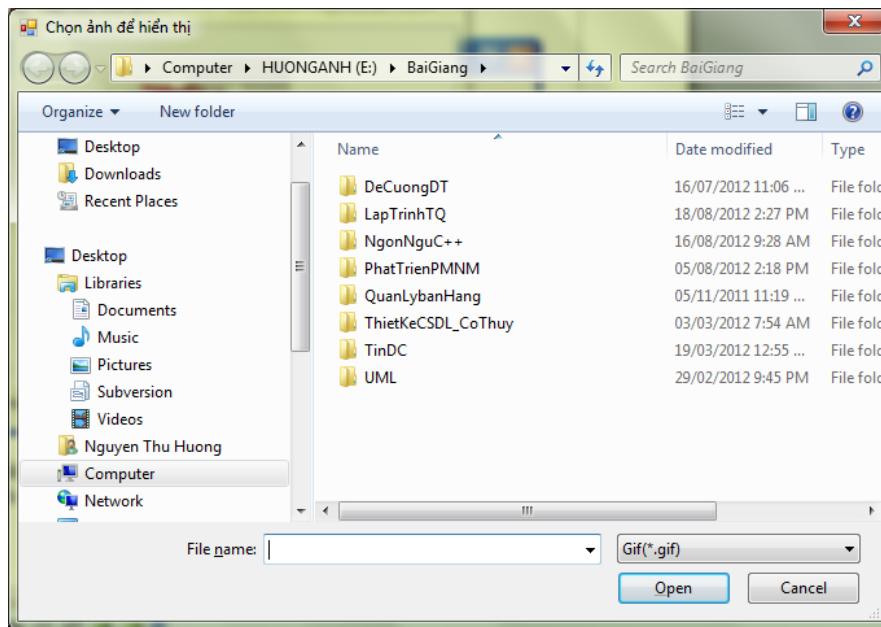
Viết chương trình cho phép dùng hộp thoại OpenFileDialog chọn hình ảnh để hiển thị.

Vào Microsoft Visual Studio 2010 tạo dự án mới có tên là **OpenDialog** và thiết lập thuộc tính của các điều khiển như sau:

Bài giảng lập trình trực quan

Điều khiển	Name	Text
Form1	frmPicture	Hiển thị hình ảnh
PictureBox	picAnh	
Button1	btnOpen	Open

Chạy chương trình, kích chọn nút lệnh Open sẽ xuất hiện hộp thoại sau:



Mở đường dẫn chứa ảnh cần hiển thị, kích chọn ảnh sau đó kích chọn **Open**, ảnh sẽ được hiển thị lên điều khiển PictureBox trên form như sau:



Trong trường hợp người dùng chọn **Cancel** sẽ xuất hiện thông báo: “You clicked Cancel!”.

Hướng dẫn:

Mở sự kiện Click của nút Open rồi viết lệnh sau:

Bài giảng lập trình trực quan

```
private void btnOpen_Click(object sender, EventArgs e)
{
    OpenFileDialog dlgOpen = new OpenFileDialog();
    dlgOpen.Filter = "Bitmap (*.bmp)|*.bmp|Gif (*.gif) | *.gif|All files (*.*)|*.*";
    dlgOpen.InitialDirectory = "E:\BaiGiang";
    dlgOpen.FilterIndex = 2;
    dlgOpen.Title = "Chọn ảnh để hiển thị";
    if (dlgOpen.ShowDialog ()== System.Windows.Forms.DialogResult.OK)
        picAnh.Image = Image.FromFile(dlgOpen.FileName);
    else
        MessageBox.Show("You clicked Cancel" , "Open Dialog",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

5.2. Hộp thoại SaveFile và luồng FileStream

5.2.1. Hộp thoại SaveFile

Các thuộc tính và phương thức quan trọng của hộp hội thoại SaveFile:

Thuộc tính	Chức năng
FileName	Cung cấp tên và đường dẫn của tập tin đã chọn.
Filter	Xác định danh sách các bộ lọc tập tin mà hộp hội thoại sẽ hiển thị, ví dụ: “Text *.txt Icons *.ico All files *.*”
FilterIndex	Chỉ ra bộ lọc tập tin mặc định, giả sử có 3 bộ lọc (*.com), (*.exe) và (*.ico) nếu FilterIndex = 2 thì hộp thoại sẽ hiển thị sẵn bộ lọc (*.exe)
InitialDirectory	Xác định thư mục mặc định cho hộp hội thoại khi vừa được gọi.
AddExtension	= True tự động thêm phần mở rộng hiện hành vào tên tập người dùng chọn nếu người dùng không chỉ rõ phần mở rộng của tên tập.
DefaultExt	Cung cấp phần mở rộng mặc định cho tên tập nếu người dùng không chỉ rõ phần mở rộng của tên tập, ví dụ: “.doc”
Title	Xác định tiêu đề của hộp hội thoại.

5.2.2. Luồng FileStream

Với một file dữ liệu có kiểu bất kỳ đều được coi như là một luồng dữ liệu, ta có thể mở một luồng dữ liệu để đọc thông tin từ file hoặc ghi thông tin vào file sau đó đóng luồng lại.

Để làm việc với luồng dữ liệu ta sẽ làm việc với namespace System.IO bằng cách dùng từ khóa using:

```
using System.IO;
```

Bài giảng lập trình trực quan

1. Luồng ghi dữ liệu - StreamWriter

Mở luồng để ghi file:

```
StreamWriter Tenluong = new StreamWriter(Tenfile)
```

Ghi từng dòng dữ liệu vào file:

```
Tenluong.WriteLine("Noidung")
```

Ghi tất cả dữ liệu vào file:

```
Tenluong.Write("Noidung")
```

Đóng luồng:

```
Tenluong.Close()
```

2. Luồng đọc dữ liệu - StreamReader

Mở luồng để đọc file:

```
StreamReader Tenluong = new StreamReader(Tenfile)
```

Đọc từng dòng dữ liệu của file: ta dùng vòng lặp với số lần lặp không xác định để đọc từng dòng dữ liệu, nếu đọc thành công thì trả về chuỗi chưa dữ liệu đọc được, nếu đến cuối file thì trả về Nothing.

```
Noidung = Tenluong.ReadLine()
```

Đọc tất cả dữ liệu của file lưu vào một biến:

```
Noidung = Tenluong.ReadToEnd()
```

Đóng luồng:

```
Tenluong.Close()
```

Bài tập 17:

Viết chương trình cho phép lưu nội dung của một hộp TextBox vào một file trên máy tính.

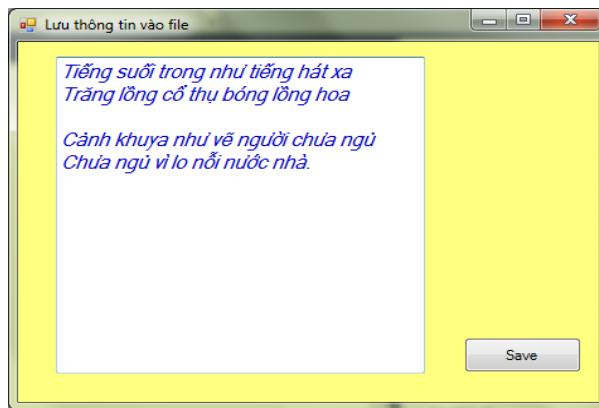
Vào Microsoft Visual Studio 2010 tạo dự án mới có tên là **SaveDialog** và thiết lập thuộc tính của các điều khiển như sau:

Điều khiển	Name	Text
Form1	frmSave	Lưu thông tin vào File

Bài giảng lập trình trực quan

TextBox (MultiLine = True)	txtSave	
Button1	btnSave	Save

Khi chạy chương trình gõ nội dung cần lưu vào TextBox



Kích chọn nút Save xuất hiện hộp thoại *Chọn file để lưu* với đường dẫn mặc định là *E:\Baigiang*. Gõ tên tệp cần lưu dữ liệu vào ô *FileName*, ví dụ: *Canh_Khuya.doc* và chọn *Save*.

Kết quả trong thư mục *E:\Baigiang* xuất hiện tệp *Canh_Khuya.doc* chứa nội dung của hộp Textbox *txtSave*.

Hướng dẫn:

Khai báo thêm namespace System.IO như sau:

```
using System.IO;
```

Mở sự kiện Click của nút btnSave và viết lệnh như sau:

```
private void btnSave_Click(object sender, EventArgs e)
{
    SaveFileDialog dlgSave = new SaveFileDialog();
    dlgSave.Filter = "Text file (*.txt) | *.txt | Word Document (*.doc)
                     | *.doc | All files (*.*) | *.*";
    dlgSave.InitialDirectory = "E:\\Baigiang";
    dlgSave.FilterIndex = 2;
    dlgSave.Title = "Chọn File để lưu";
    dlgSave.AddExtension = true;
    dlgSave.DefaultExt = ".doc";
    if (dlgSave.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
        StreamWriter file = new StreamWriter(dlgSave.FileName);
        try
        {
            file.WriteLine(txtSave.Text);
            MessageBox.Show("Thành công");
        }
        catch
        {
            MessageBox.Show("Lỗi ghi file");
        }
    }
}
```

```
        file.Close();
    }
else
    MessageBox.Show("you clicked Cancel");
}
```

5.3. Hộp thoại Color

Có chức năng hiển thị bảng màu hiện hành cho phép người sử dụng chọn một màu bất kỳ để tô màu chữ, màu nền.

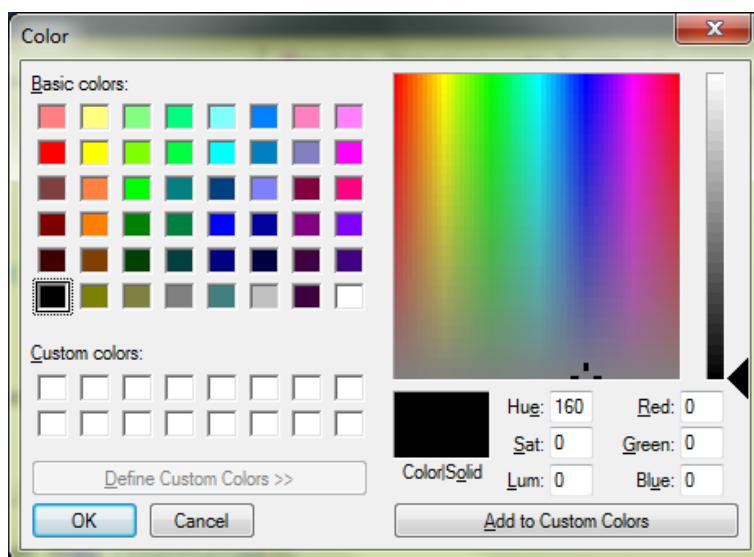
Các thuộc tính quan trọng của hộp thoại Color:

Thuộc tính	Chức năng
Color	Trả về màu được chọn trong hộp thoại Color.
FullOpen	Hiển thị toàn bộ hộp thoại Color.
SolidColorOnly	Không hiển thị phần Define Custom Colors.

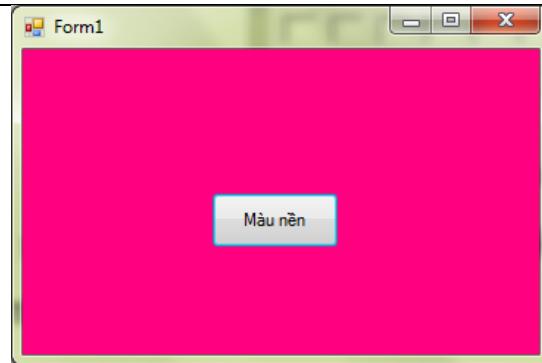
Bài tập 18:

Viết chương trình cho phép chọn màu nền cho form.

Chạy chương trình, kích chọn nút lệnh *Màu nền* sẽ xuất hiện hộp thoại sau:



Chọn một màu bất kỳ rồi bấm OK, khi đó màu được chọn sẽ được gán làm màu nền cho form, ví dụ:



Hướng dẫn:

Vào Microsoft Visual Studio 2010 tạo dự án mới có tên là **ColorDialog** và kéo vào Form một điều khiển button đặt tên là **btnColor**.

Mở cửa sổ code của sự kiện Click của nút **btnColor** viết lệnh sau:

```
private void btnColor_Click(object sender, EventArgs e)
{
    ColorDialog dlgColor = new ColorDialog();
    dlgColor.FullOpen = true;
    if (dlgColor.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        this.BackColor = dlgColor.Color;
    else
        MessageBox.Show("You clicked Cancel", "Color Dialog",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

5.4. Hộp thoại Font

Có chức năng hiển thị hộp thoại Font cho phép người sử dụng chọn font chữ, kiểu chữ, cỡ chữ...

Các thuộc tính quan trọng của hộp thoại Font:

Thuộc tính	Chức năng
Font	Trả về kiểu Font chữ được chọn trong hộp thoại Font.
ShowColor	= True: cho phép hiển thị hộp thoại Color.
Color	Trả về màu được chọn trong hộp thoại Font.

Bài tập 19:

Viết chương trình cho phép chọn kiểu font chữ khác nhau cho TextBox.

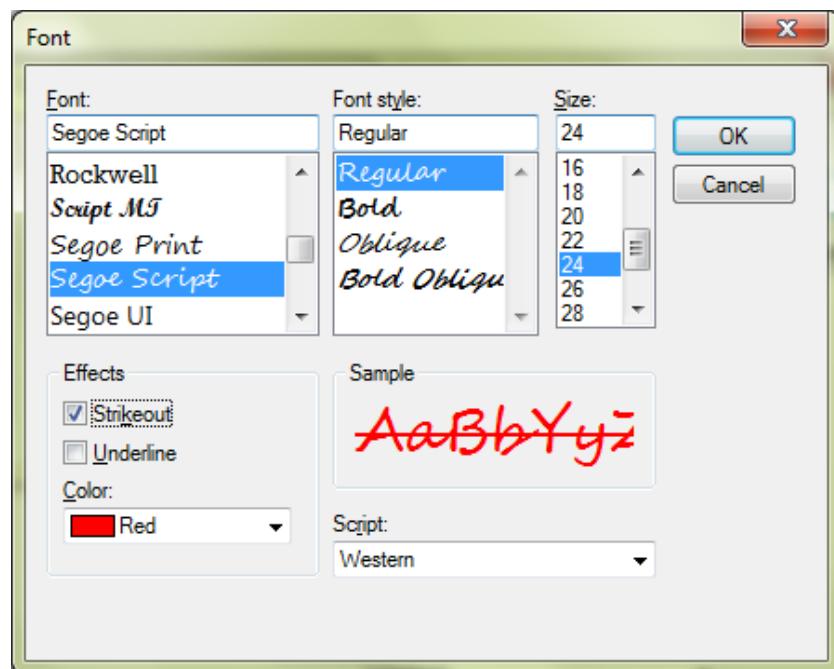
Tạo dự án mới với các điều khiển có thuộc tính Name, Text như sau:
Form1(frmFont, "Hộp thoại chọn Font"), TextBox1(txtFont, "Hà Nội"), Button1(btnFont, "Chọn Font")

Bài giảng lập trình trực quan

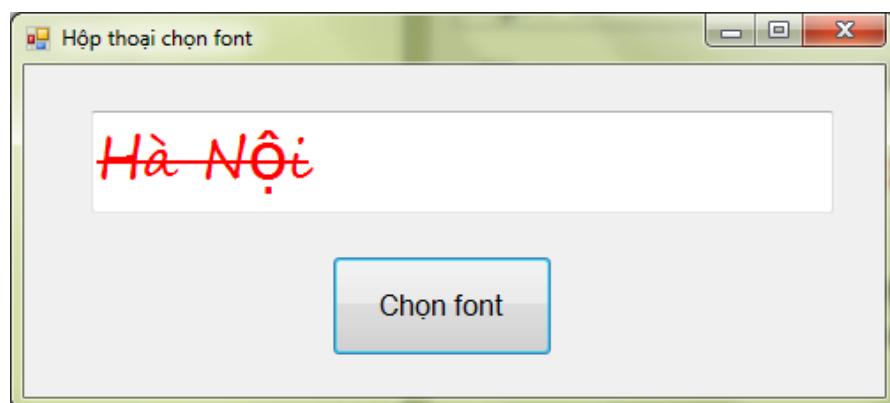
Mở sự kiện Click của nút lệnh btnFont và viết lệnh như sau:

```
private void btnFont_Click(object sender, EventArgs e)
{
    FontDialog dlgFont = new FontDialog();
    dlgFont.ShowColor = true;
    if (dlgFont.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
        txtFont.Font = dlgFont.Font;
        txtFont.ForeColor = dlgFont.Color;
    }
    else
        MessageBox.Show("You clicked Cancel", "Font Dialog",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

Chạy chương trình, kích chọn nút lệnh Chọn Font sẽ xuất hiện hộp thoại Font:



Chọn giá trị cho các thuộc tính rồi bấm OK ta được kết quả như sau:

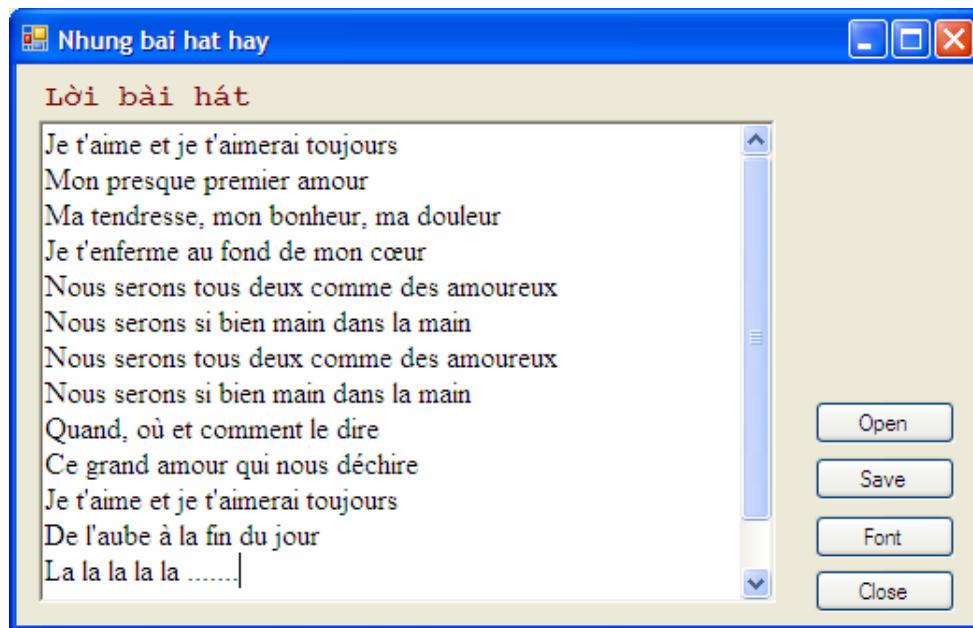


Bài giảng lập trình trực quan

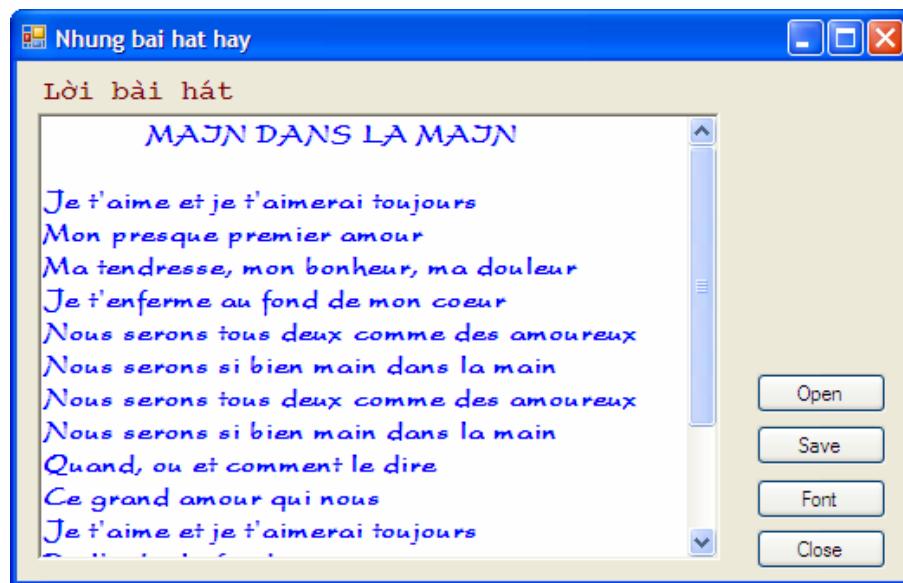
Bài tập 20:

Viết chương trình cho phép đọc nội dung một tệp *.rtf* trong máy lưu vào hộp RichTextBox và có thể sửa nội dung, thay đổi Font chữ, màu chữ... rồi lưu lại.

Kích chọn nút *Open* xuất hiện hộp thoại *OpenFile*, chọn tệp *Main dans la main.rtf* ở thư mục *E:\Baigiang\ Nhac_Loi* lưu vào hộp RichTextBox



Kích chọn nút *Font* xuất hiện hộp thoại *Font*, chọn màu chữ và kiểu chữ như sau:

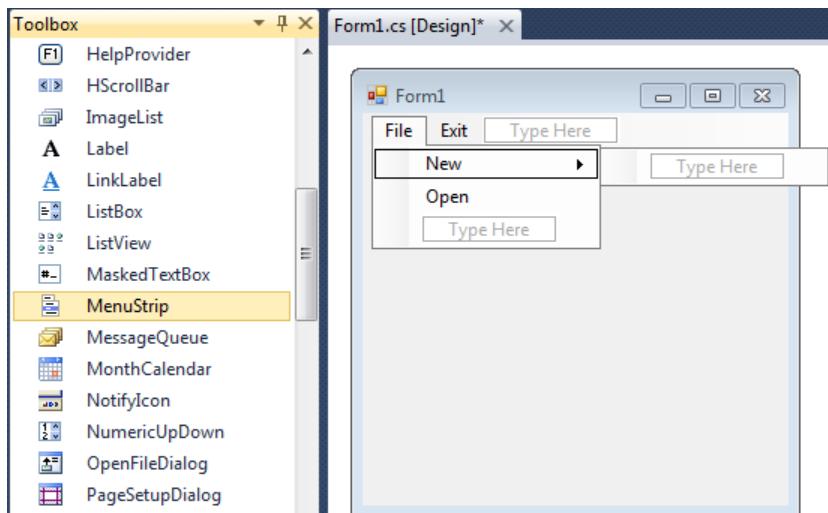


Kích chọn nút *Save* xuất hiện hộp thoại *SaveFile*, lưu nội dung hộp RichTextBox vào tệp *Main dans la main.rtf*

CHƯƠNG 6: MENU VÀ CÁC ỨNG DỤNG CÓ NHIỀU BIỂU MÃU

6.1. Điều khiển ToolStrip

Điều khiển ToolStrip giúp người lập trình có thể thiết kế thanh menu trên Form. Khi đưa điều khiển ToolStrip vào Form, một thanh menu ngang sẽ xuất hiện trên dòng đầu tiên của Form với các ô chứa dòng chữ *Type Here* cho phép tạo các mục menu mới.



1. Thuộc tính

Name	Mỗi mục menu đều phải có tên, menu có tiếp đầu ngữ là mnu
Enabled	= False: mục menu sẽ bị xám, ta không thể chọn mục menu đó.
Image	Thiết lập hình ảnh biểu tượng cho mỗi mục menu.
ShortcutKeys	Cho phép tạo phím tắt để mở các mục menu. A small screenshot of a dialog box titled 'Modifiers:' with three checkboxes: 'Ctrl' (checked), 'Shift', and 'Alt'. Below it is a 'Key:' section with a dropdown menu containing the letter 'N' and a 'Reset' button.

Bài giảng lập trình trực quan

Text	Tạo tiêu đề của các mục menu. Nếu đặt ký tự & trước một chữ cái trong thuộc tính Text thì khi chạy chương trình người dùng có thể bấm tổ hợp phím Alt + Chữ cái đó để kích hoạt menu. Ví dụ : &File sẽ cho phép bấm Alt+F để kích hoạt menu File. Nếu Text được xác lập là một dấu trừ (-) chương trình sẽ hiển thị một đường thẳng ngăn cách giữa các khoản mục menu.
Visible	= False: các mục menu sẽ không được hiển thị.
ToolTipText	Tạo dòng mách nước cho các mục menu.

2. Sự kiện

Click	Được kích hoạt khi người dùng kích chuột để chọn một mục menu.
-------	--

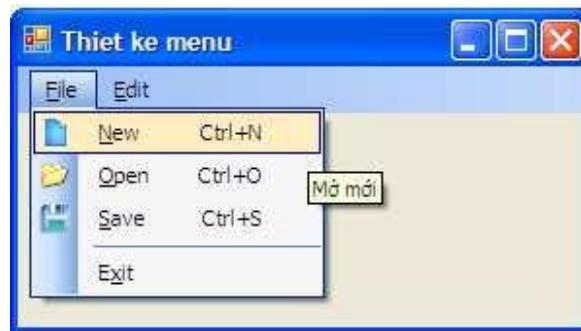
Chú ý:

Chèn mục menu: để chèn thêm một mục menu vào trong danh sách các mục menu đã có, ta kích chuột phải tại mục menu, chọn *Insert/MenuItem*. Kết quả một mục menu mới được chèn vào trên mục menu đang được chọn.

Xóa mục menu: kích chuột phải tại mục menu muốn xóa, chọn *Delete*

Bài tập 21:

Tạo một menu với giao diện như sau:



Khi người dùng kích chuột tại mục menu nào thì xuất hiện hộp thoại thông báo tên mục menu đó.

Hướng dẫn:

Vào Microsoft Visual Studio 2005 tạo dự án mới có tên là **Menu** và thiết lập thuộc tính của các điều khiển như sau:

Điều khiển	Name	Text	Image	ShortcutKeys	ToolTipText
------------	------	------	-------	--------------	-------------

Bài giảng lập trình trực quan

Form1	frmMenu	Thiet ke menu			
MenuStrip1					
mnuFile	&File				
	mnuNew	&New	New.ico	Ctrl + N	Mở mới
	mnuOpen	&Open	Open.ico	Ctrl + O	Mở tệp đã có
	mnuSave	&Save	Save.ico	Ctrl + S	Lưu tệp
	mnu1	-			
	mnuExit	E&xit			
	...				

Viết Code: Lần lượt mở sự kiện click của các menu con và viết mã lệnh. Ví dụ với sự kiện click của mnuNew như sau:

```
private void mnuNew_Click(object sender, EventArgs e)
{
    MessageBox.Show("Bạn đã chọn mục menu New",
                    "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

Các mục menu khác làm tương tự

6.2. Popup menu – ContextMenuStrip

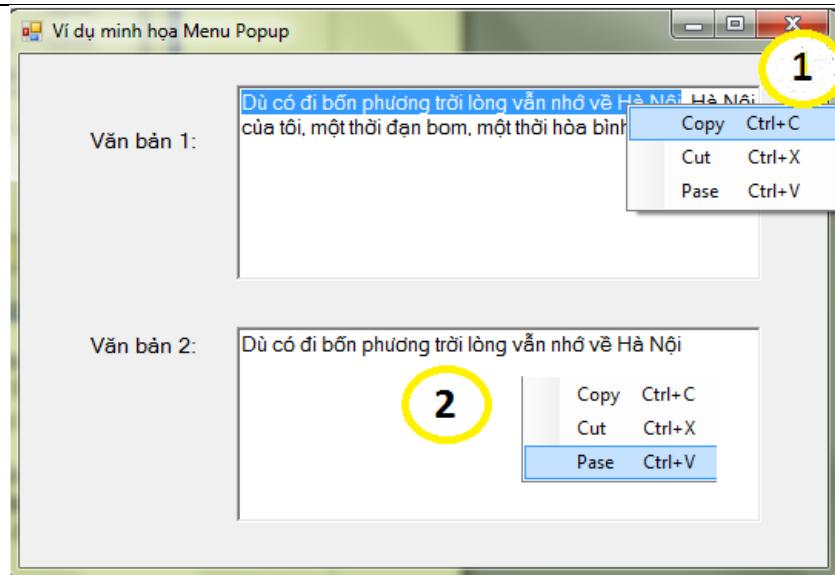
Điều khiển ContextMenuStrip  cho phép tạo các menu dạng Popup được gắn với một điều khiển nào đó trên Form, menu này chỉ xuất hiện khi người dùng dùng kích chuột phải tại điều khiển gắn menu đó.

Điều khiển ContextMenuStrip có tiếp đầu ngữ là **cmnu** và có các thuộc tính và phương thức tương tự như điều khiển MenuStrip.

Bài tập 22:

Viết chương trình tạo menu Popup có giao diện và các chức năng như sau:

Bài giảng lập trình trực quan



Bôi đen các dòng văn bản trong hộp *Văn bản 1* và kích chuột phải tại vị trí bất kỳ trong hộp *Văn bản 1*, xuất hiện menu gồm 3 mục lựa chọn: ***Copy***, ***Cut***, ***Paste***. Kích chọn mục *Copy* để lưu các dòng văn bản được bôi đen vào bộ nhớ đệm ClipBoard. Kích chuột phải tại vị trí bất kỳ trong hộp *Văn bản 2* chọn *Paste* để gán nội dung dòng văn bản trong ClipBoard vào hộp *Văn bản 2*.

Hướng dẫn

Vào Microsoft Visual Studio 2010 tạo dự án mới có tên là **MenuPopup** và thiết lập giá trị cho các thuộc tính của các điều khiển như sau:

Điều khiển	Name	Text	ContextMenuStrip	Shortcut Keys
ContextMenu Strip	cmnuEdit			
	cmnuCopy	Copy		Ctrl + C
	cmnuCut	Cut		Ctrl + X
	cmnuPaste	Paste		Ctrl + V
Label1		Văn bản 1		
Label2		Văn bản 2		
RichTextBox1	rtbVanban1		cmnuEdit	
RichTextBox2	rtbVanban2		cmnuEdit	

Mở sự kiện Click của 2 mnu cmnuCopy và cmnuPase viết mã lệnh như sau:

```
private void cmnuCopy_Click(object sender, EventArgs e)
{
    Clipboard.SetText(rtbVanBan1.SelectedText);
```

```
}

private void cmnuPase_Click(object sender, EventArgs e)
{
    rtbVanBan2.Text = Clipboard.GetText();
}
```

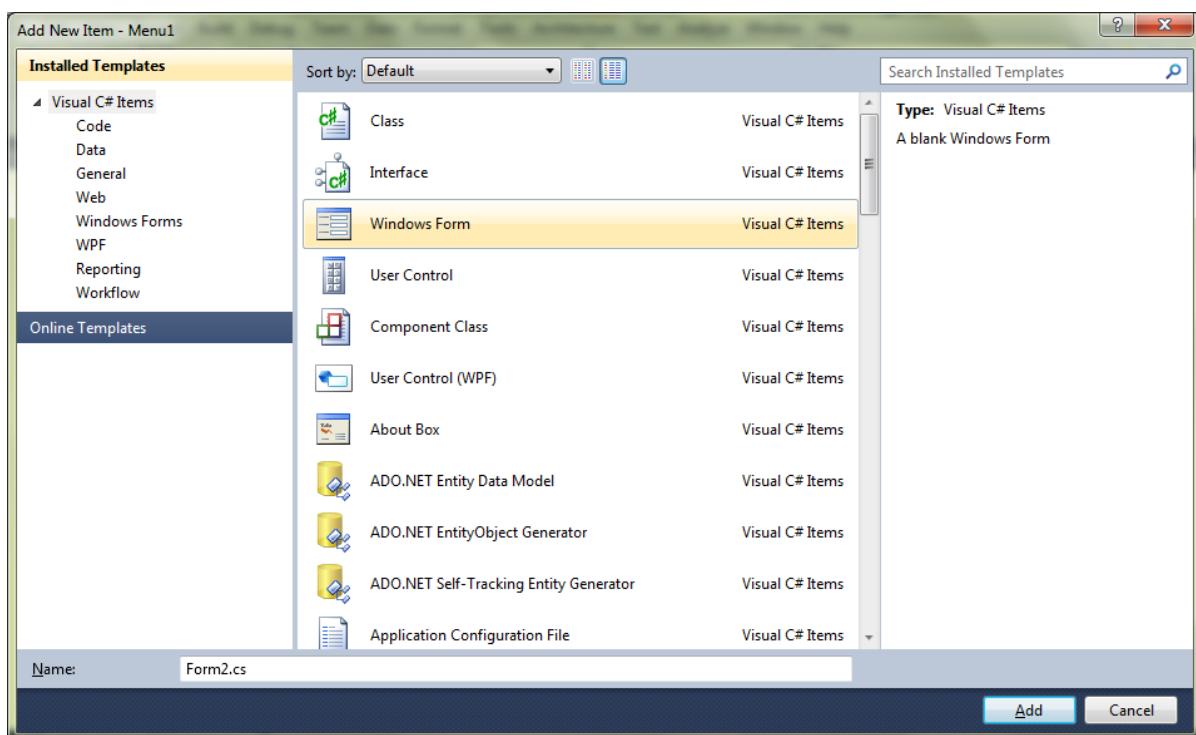
Các mục menu khác viết tương tự.

6.3. Dự án nhiều biểu mẫu

Khi ứng dụng trỏ lên phức tạp, chương trình không thể chỉ chứa trong một biểu mẫu mà phải sử dụng nhiều biểu mẫu để bổ sung tính linh hoạt và năng lực cho ứng dụng.

6.3.1. Bổ sung biểu mẫu

Để bổ sung thêm các biểu mẫu - Form cho ứng dụng đang thiết kế, chọn menu *Project/Add New Item...* hoặc kích chuột vào biểu tượng  trên thanh công cụ hoặc bấm phím tắt *Ctrl+Shift+A* xuất hiện cửa sổ Add New Item:



C# cung cấp vài biểu mẫu được xây dựng sẵn để có thể bổ sung vào dự án, ví dụ biểu mẫu cơ bản ‘Windows Form’ như ta đã biết, hoặc ‘User control’ - dùng để thiết kế các điều khiển riêng cho người dùng...

Chọn loại biểu mẫu cần bổ sung rồi bấm Add, ví dụ chọn Windows Form khi đó một biểu mẫu mới sẽ được chèn vào ứng dụng.

Bài giảng lập trình trực quan

Ngoài ra C# còn cho phép bổ sung các form đã được xây dựng trong các dự án khác bằng cách chọn menu *Project/ Add Existing Item...* hoặc bấm phím tắt *Shift +Alt + A* rồi chọn Form cần bổ sung. Việc chèn Form đã có sẵn tạo điều kiện cho việc phân nhỏ ứng dụng cho nhiều người, mỗi người làm một phần sau đó ghép lại với nhau tạo thành ứng dụng hoàn chỉnh.

Bài tập 23:

Vào Microsoft Visual Studio 2010 tạo dự án mới có tên là ***BaiTapToan***. Đổi tên cho Form1 thành ***frmMain*** và tiêu đề là ***Bài tập toán lớp 2***.

Bổ sung thêm 2 Windows Form mới có tên là ***frmPhepCong1*** và ***frmSoSanh1*** với tiêu đề lần lượt là ***Bài tập về phép cộng 1*** và ***Bài tập so sánh 1***.

Lúc này dự án có 03 Form là: frmMain, frmPhepCong1 và frmSoSanh1.

6.3.2. Biểu mẫu khởi động

Khi chạy một ứng dụng, biểu mẫu được thực hiện đầu tiên gọi là biểu mẫu khởi động. Biểu mẫu Startup này mặc định là biểu mẫu đầu tiên được xây dựng trong ứng dụng khi thiết kế giao diện, ví dụ biểu mẫu frmMain trong ứng dụng *BaiTapToan*.

Ta có thể thay đổi mặc định này để có thể chọn một biểu mẫu bất kỳ làm biểu mẫu khởi động khi chạy chương trình. Ví dụ muốn form *frmPhepCong1* được thực hiện đầu tiên, ta thực hiện như sau:

+ Mở file *Program.cs*

+ Thay đổi tên form bắt đầu chạy trong dòng lệnh sau:

```
Application.Run(new frmMain());
```

Thành như sau:

```
Application.Run(new frmPhepCong());
```

6.3.3. Mở biểu mẫu

Để mở một biểu mẫu ta có phải khởi tạo đối tượng form cho Form cần mở như sau:

+ Ví dụ cần mở form frmPhepCong ta khởi tạo như sau:

```
frmPhepCong phepCong=new frmPhepCong();
```

Sau đó ta thể thực hiện theo 2 cách sau:

Cách 1: cú pháp ***phepCong.Show()***

Bài giảng lập trình trực quan

Tải biểu mẫu vào bộ nhớ sau đó đưa biểu mẫu lên phía trên các biểu mẫu khác (nếu có), phương thức này cho phép người sử dụng có thể tương tác được với các biểu mẫu nằm phía dưới.

Cách 2: cú pháp `phepCong.ShowDialog()`

Tương tự như cách 1 nhưng người sử dụng chỉ có thể tương tác được với biểu mẫu hiện hành mà không thể tương tác với các biểu mẫu nằm phía dưới nó.

6.3.4. Đóng biểu mẫu

Để đóng một biểu mẫu ta có thể thực hiện theo 2 cách sau:

Cách 1: đóng form đang làm việc `this.Close();`

Cách 2: đóng một form bất kỳ, ví dụ form frmPhepCong được khởi tạo như trên:

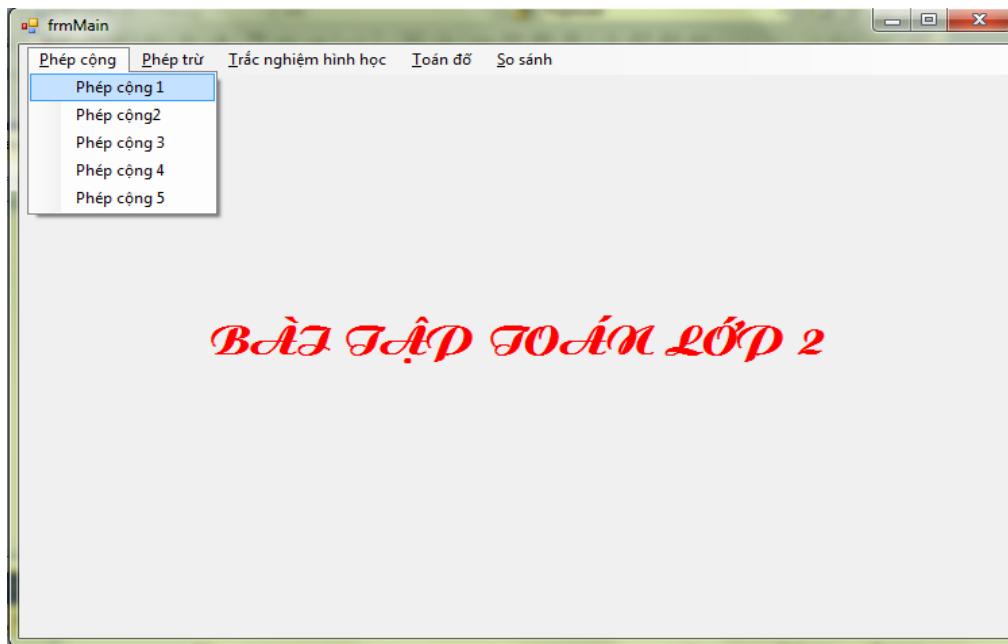
```
phepCong.Close();
```

6.3.5. Xóa biểu mẫu

Kích chuột phải tại Form cần xóa trong cửa sổ *Solution Explorer*, chọn *Delete* kết quả Form sẽ bị xóa ra khỏi dự án.

Bài tập 24:

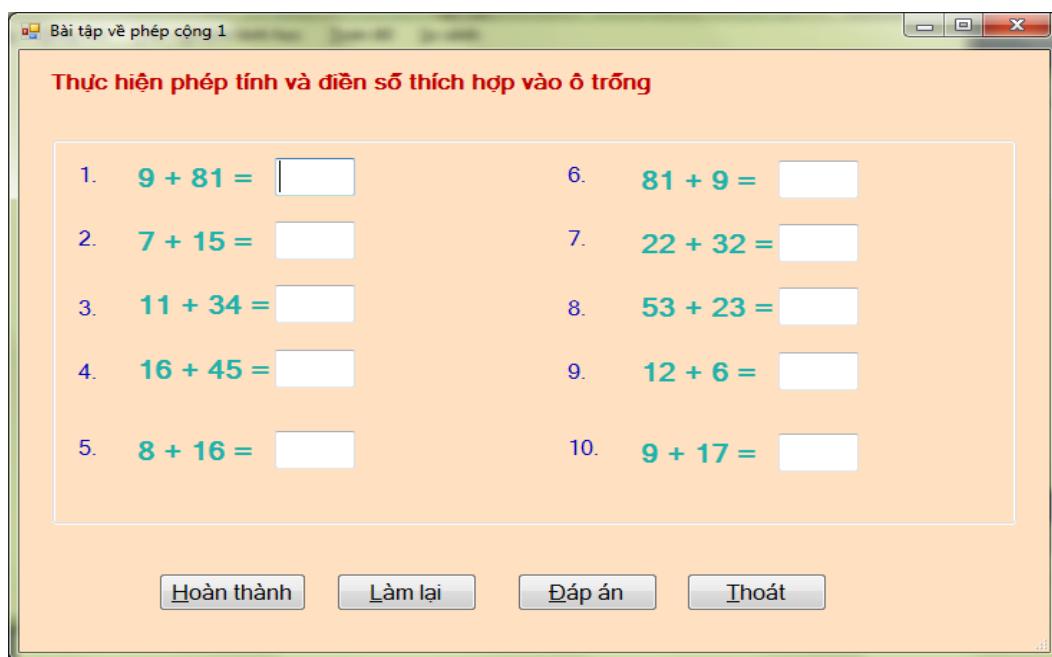
Tiếp theo bài tập 23, tạo ToolStrip cho Form *frmMain* theo giao diện sau:



Bài tập 25:

Bài giảng lập trình trực quan

Tiếp bài tập 24, tạo form frmPhepCong1 với giao diện như sau:



Yêu cầu:

- + Khi người dùng chọn mục *Phép cộng 1* trên menu Form *frmPhepCong1* xuất hiện nội dung bài tập cần điền số thích hợp vào trong hộp Textbox.
- + Chỉ cho phép gõ số nguyên vào các ô TextBox.
- + Người sử dụng viết đáp án cho các câu vào các ô Textbox từ 1 đến 10.
- + Khi chọn nút **Hoàn thành** chương trình kiểm tra kết quả, câu nào đúng thì đổi màu nền ở Textbox tương ứng với câu trả lời sang màu xanh, câu nào sai thì đổi Textbox có nền màu đỏ. Và hiển thị điểm đạt được cho người dùng (mỗi câu trả lời đúng được 1 điểm).
- + Khi chọn nút **Đáp án** thì form sẽ hiển thị đáp án như giao diện sau:

Bài giảng lập trình trực quan



+ Bấm nút **Làm lại**, xuất hiện nội dung đề bài và xoá rỗng các đáp án cũ để người dùng có thể trả lời lại các câu hỏi.

+ Bấm nút **Thoát** để đóng form frmPhepCong1 trở về form chính frmMain.

Bài tập 26:

Tiếp bài tập 25, tạo form frmSoSanh1 với giao diện như sau:

Bài giảng lập trình trực quan

Bài tập so sánh 1

Điền dấu >, < hoặc = vào ô trống

1.	23	<input type="text"/>	$9 + 10$	6.	$81 - 9$	<input type="text"/>	$45 + 2$
2.	$34 + 3$	<input type="text"/>	37	7.	$26 - 3$	<input type="text"/>	26
3.	11	<input type="text"/>	18	8.	92	<input type="text"/>	$11 + 81$
4.	16	<input type="text"/>	$16 + 2$	9.	15	<input type="text"/>	$17 - 2$
5.	8	<input type="text"/>	$11 - 3$	10.	$8 + 9$	<input type="text"/>	16

Hoàn thành Làm lại Đáp án Thoát

Khi chọn mục *So sánh/So sánh 1* thì Form *frmSoSanh1* xuất hiện nội dung các câu cần điền dấu >, < hoặc =.

Các nút *Hoàn thành*, *Làm lại*, *Đáp án* và *Thoát* có chức năng tương tự như trong form *frmPhepCong1*.

CHƯƠNG 7: LẬP TRÌNH CƠ SỞ DỮ LIỆU

7.1. Giới thiệu về bài toán

7.1.1. Lập trình cơ sở dữ liệu và bài toán quản lý

Hiện nay, một trong những mảng phần mềm được ứng dụng rộng rãi nhất đó là phần mềm quản lý, ví dụ như: quản lý bán hàng, quản lý điểm, quản lý đăng ký học, quản lý công văn giấy tờ. Bài toán quản lý đã dần trở thành bài toán phổ biến, quen thuộc và nó có tầm quan trọng ảnh hưởng lớn đến sự phát triển kinh tế của xã hội.

Bất kể một bài toán quản lý nào cũng cần có một cơ sở dữ liệu thích hợp để lưu trữ quản lý thông tin của nó. Và một chương trình quản lý đòi hỏi phải có những yêu cầu về kết nối và các thao tác trên CSDL.

Lập trình CSDL là một trong các thế mạnh của .NET cho phép dễ dàng tạo ra những phần mềm quản lý chuyên nghiệp, có thể kết nối tới CSDL của hầu hết các hệ quản trị dữ liệu như: MS Access, SQL server, MySQL server, Oracle, ... Hơn thế nữa nó còn có thể kết nối đến các dạng CSDL khác như Excel, ... để thêm mới, sửa, xóa, tìm kiếm và kết xuất dữ liệu với độ chính xác cao, chức năng phong phú và giao diện thân thiện.

Trong chương trình này ta sẽ học về cách thao tác và làm việc với hệ quản trị CSDL SQL server cùng các tiện ích khác như xuất dữ liệu ra Excel.

Để học phần này máy tính của bạn học cần cài đặt các thành phần sau:

+ Bộ Visual Studio (bản 2005 hoặc bản 2008 hoặc bản 2010 – Cài bản đầy đủ). Khi cài đầy đủ thì chúng đã có sẵn cả SQL server rồi.

7.1.2. Cách tổ chức các tài nguyên trong một dự án của bài toán quản lý

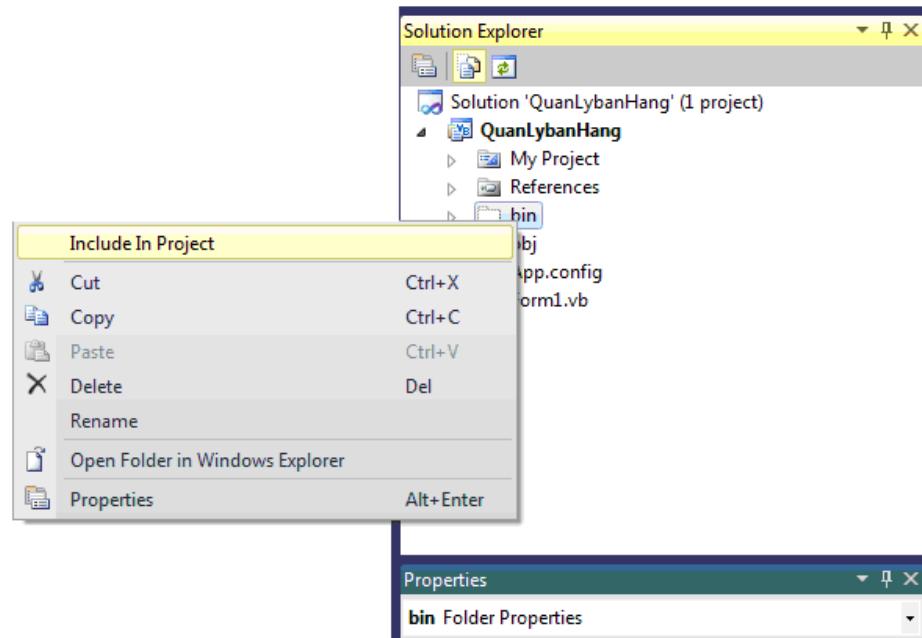
Để dễ dàng quản lý tài nguyên của ứng dụng ta nên phân chia ứng dụng thành các thư mục nhỏ, mỗi một thư mục chứa các file cùng thực hiện một nhóm chức năng. Ví dụ thư mục **Images** dùng để chứa các file ảnh.

Để tổ chức tài nguyên cho một chương trình quản lý, thông thường ta làm như sau:

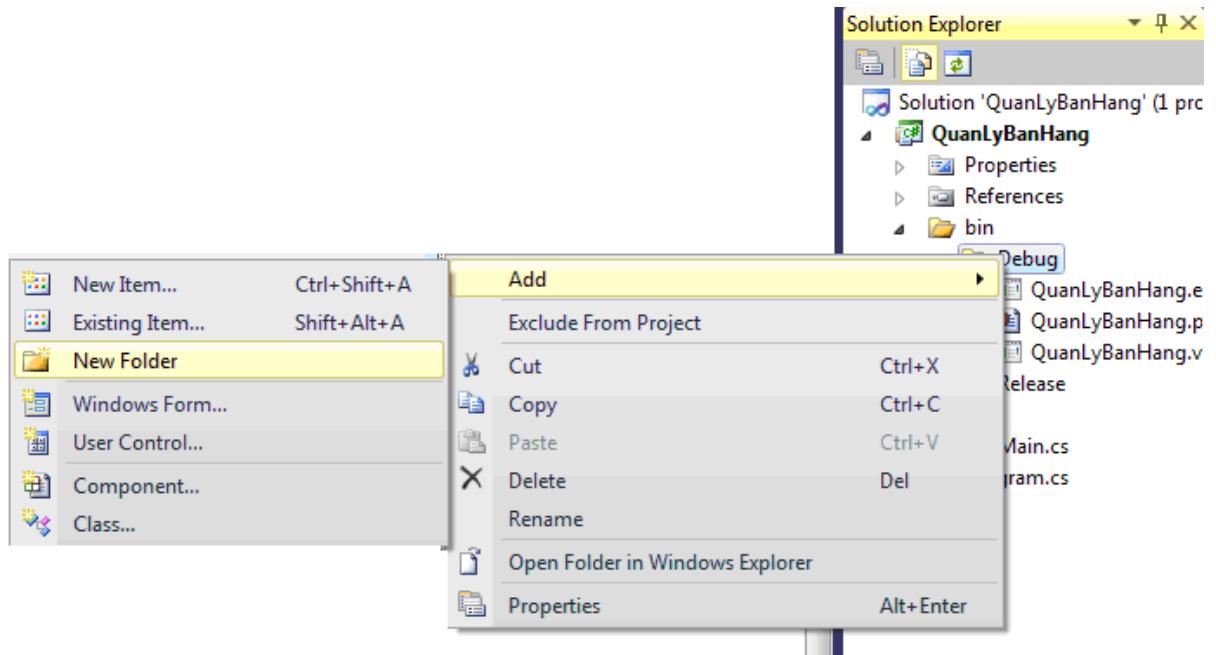
+ Khởi động Microsoft Visual Studio 2010 tạo dự án mới (giả sử dự án mới có tên là **QuanLyBanHang** lưu vào thư mục E:\). Trong dự án ta tạo một số thư mục để lưu các nhóm đối tượng khác nhau.

Bài giảng lập trình trực quan

+ Sau khi tạo xong dự án mới, trên cửa sổ Solution Explorer kích vào biểu tượng show all file để thư mục bin và obj hiện lên. Lúc này thư mục bin bị mờ đi ta kích chuột phải vào nó rồi chọn **Include In Project** như hình sau:



+ Bây giờ ta tạo thêm trong /bin/Debug/ thư mục **DataBase** để chứa cơ sở dữ liệu cho ứng dụng. Bằng cách kích chuột phải vào bin chọn **Add** và chọn **new folder**, sau đó đổi tên thư mục là **DataBase**

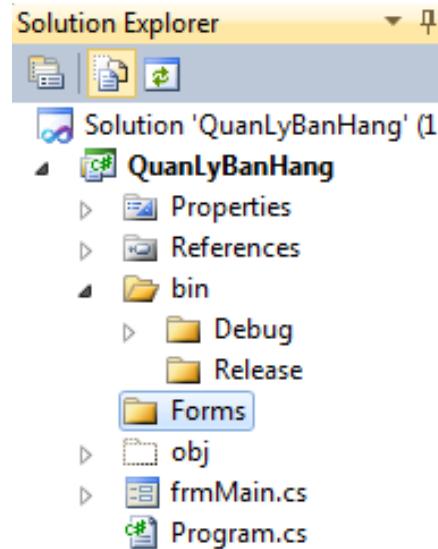


+ Tương tự ta tạo thư mục để chứa ảnh là **Images** trong bin/Debug.

Bài giảng lập trình trực quan

+ Trong cửa sổ Solution Explorer kích chuột phải tại tên dự án chọn **Add**, chọn **new folder**, đổi tên **Folder** thành **Forms**. Thư mục này có tác dụng chứa các form trong chương trình.

Cuối cùng cây thư mục chứa tài nguyên của ứng dụng như sau:



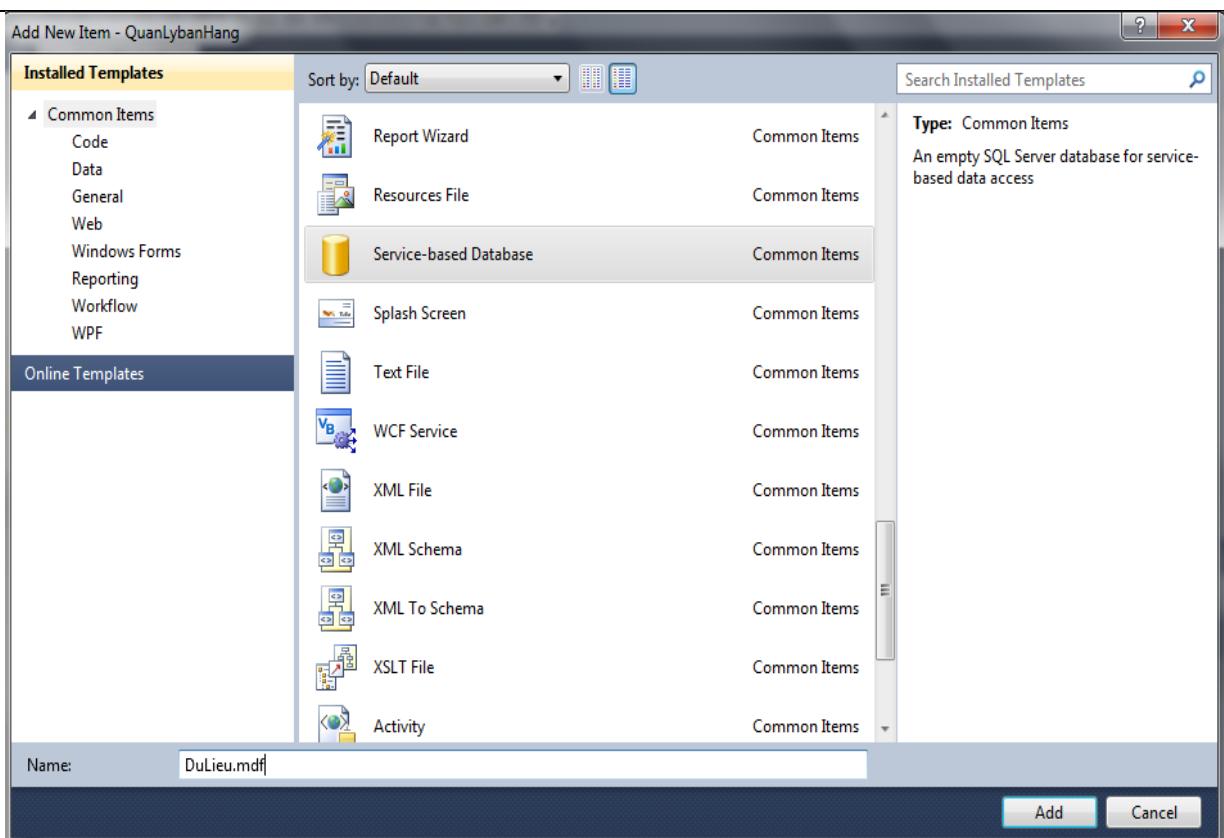
7.2. Cách tạo cơ sở dữ liệu (Database) trong môi trường visual studio 2010

Trong bộ Visual Studio 2010 có hỗ trợ SQL server 2008, nên ta dùng luôn hỗ trợ này để tạo CSDL như hướng dẫn dưới đây:

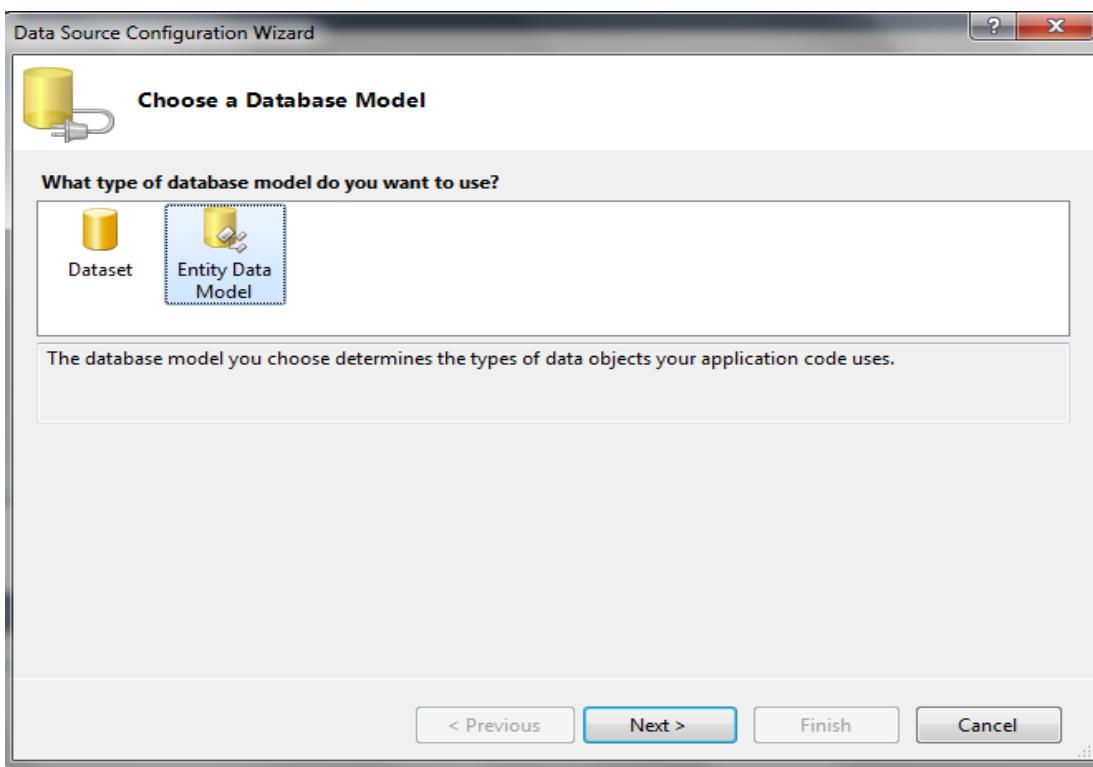
7.2.1. Tạo mới một DataBase

Ta sẽ lưu CSDL vào thư mục **DataBase** đã tạo trong thư mục **bin/Degug/** ở trên. Trong cửa sổ Solution Explorer kích chuột phải tại thư mục DataBase, chọn **Add/New Item...** xuất hiện cửa sổ **Add new item**

Bài giảng lập trình trực quan

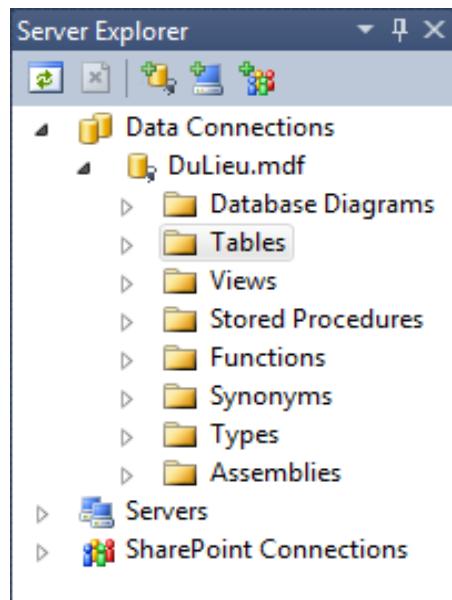


Chọn **Service-based DataBase**, ở hộp **Name** đặt tên CSDL là **DuLieu.mdf** sau đó nhấn nút **Add**. Kết quả xuất hiện cửa sổ Data Source Configuration Wizard:

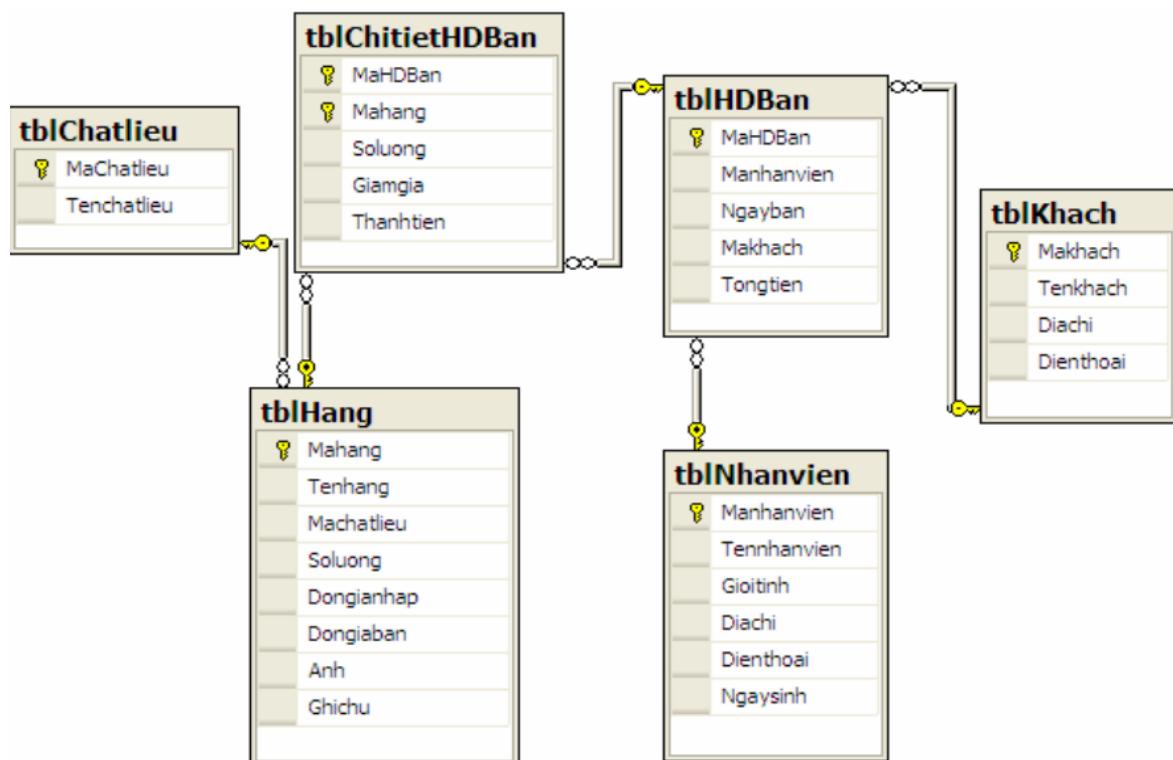


Bài giảng lập trình trực quan

Chọn **Cancel** để tạo một CSDL rỗng. Kết quả trong thư mục **DataBase** xuất hiện vào file **DuLieu.mdf**. Kích đúp chuột vào đối tượng này xuất hiện cửa sổ **Server Explorer** cho phép tạo CSDL cho dự án như sau:



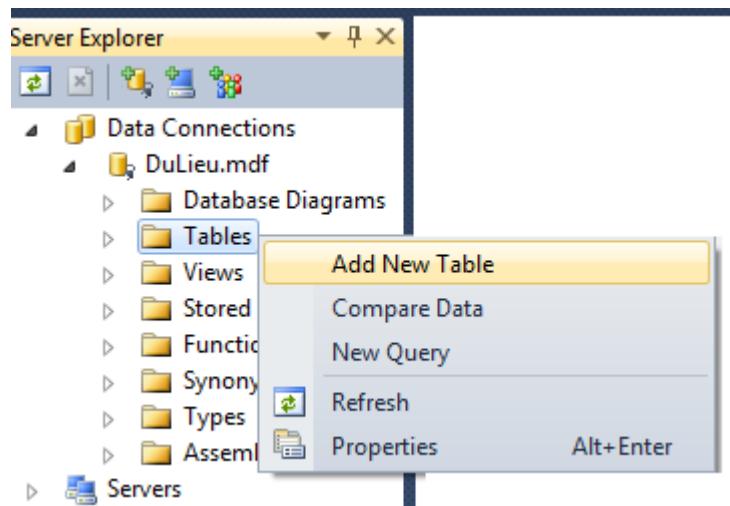
Bây giờ ta bắt đầu tạo ra một CSDL quan hệ như sau:



7.2.2. Tạo các bảng CSDL

Để tạo một bảng, ví dụ như bảng **tblChatlieu** ta làm như sau:

- Kích chuột phải vào **Table** và chọn **Add New Table**



- Tạo bảng chất liệu gồm các trường như sau:

dbo.Table1: T... \DULIEU.MDF)*			Form 1.vb [Design]	Start Page
	Column Name	Data Type	Allow Nulls	
1	MaChatlieu	nvarchar(10)	<input type="checkbox"/>	
2	Tenchatlieu	nvarchar(50)	<input type="checkbox"/>	

- Để tạo khóa chính cho trường *MaChatlieu*, ta kích cuộn phải tại dòng *MaChatlieu* chọn *Set Primary Key*.
- Để lưu bảng nhấn *Ctrl + S* xuất hiện cửa sổ *Choose Name*, đặt tên cho bảng là *tblChatlieu* rồi nhấn *OK*.



Bài giảng lập trình trực quan

Bài tập: Sinh viên tự tạo tương tự với các bảng khác với tên bảng và các trường dữ liệu như sau:

Bảng *tblHang*

Column Name	Data Type	Allow Nulls
MaHang	nvarchar(10)	<input type="checkbox"/>
TenHang	nvarchar(50)	<input type="checkbox"/>
MaChatLieu	nvarchar(10)	<input type="checkbox"/>
SoLuong	int	<input type="checkbox"/>
DonGiaNhap	float	<input type="checkbox"/>
► DonGiaBan	float	<input checked="" type="checkbox"/>
Anh	ntext	<input checked="" type="checkbox"/>
GhiChu	ntext	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Bảng *tblNhanvien*

Column Name	Data Type	Allow Nulls
► MaNhanVien	nvarchar(10)	<input type="checkbox"/>
TenNhanVien	nvarchar(50)	<input type="checkbox"/>
GioiTinh	nvarchar(10)	<input type="checkbox"/>
DiaChi	nvarchar(100)	<input checked="" type="checkbox"/>
DienThoai	nvarchar(15)	<input checked="" type="checkbox"/>
NgaySinh	date	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Bảng *tblKhach*

Column Name	Data Type	Allow Nulls
► MaKhach	nvarchar(10)	<input type="checkbox"/>
TenKhach	nvarchar(50)	<input type="checkbox"/>
DiaChi	ntext	<input checked="" type="checkbox"/>
DienThoai	nvarchar(15)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Bài giảng lập trình trực quan

Bảng *tblHDBan*

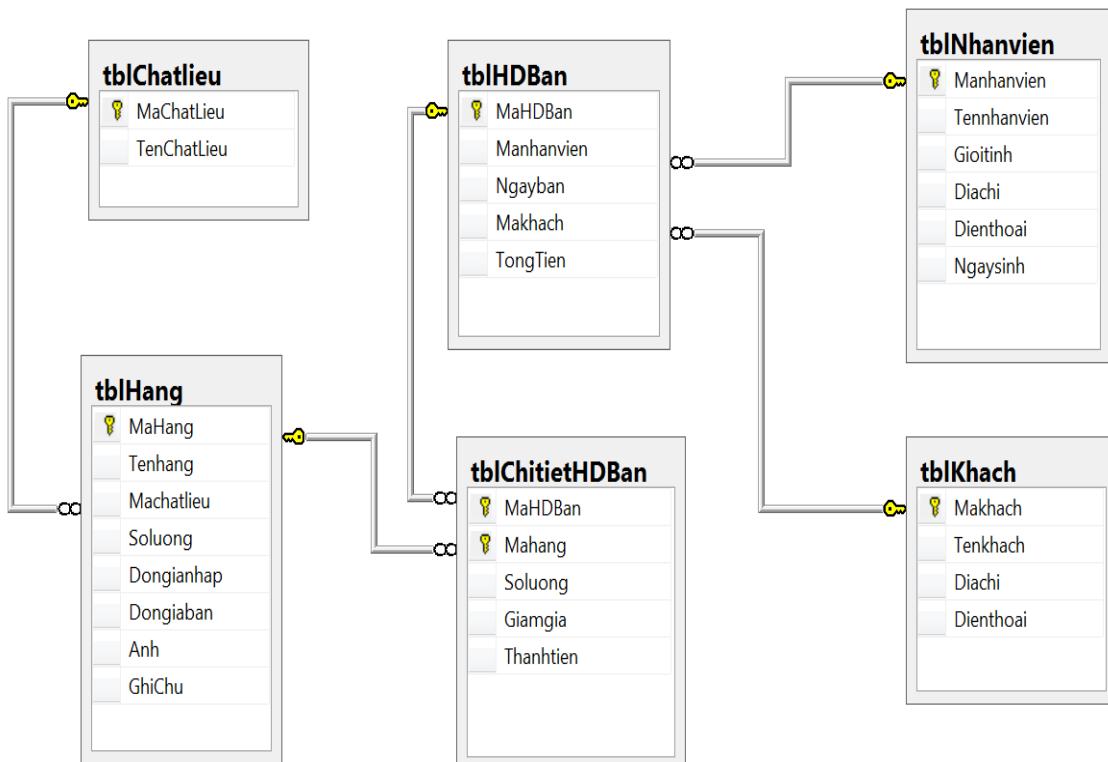
Column Name	Data Type	Allow Nulls
MaHDBan	nvarchar(30)	<input type="checkbox"/>
Manhanvien	nvarchar(10)	<input type="checkbox"/>
NgayBan	date	<input type="checkbox"/>
Makhach	nvarchar(10)	<input type="checkbox"/>
TongTien	float	<input checked="" type="checkbox"/> <input type="checkbox"/>

Bảng *tblChitietHDBan*

Column Name	Data Type	Allow Nulls
MaHDBan	nvarchar(30)	<input type="checkbox"/>
Mahang	nvarchar(10)	<input type="checkbox"/>
Soluong	float	<input type="checkbox"/>
Giamgia	float	<input type="checkbox"/>
Thanhtien	float	<input type="checkbox"/>

7.2.3. Tạo quan hệ Relationship cho CSDL

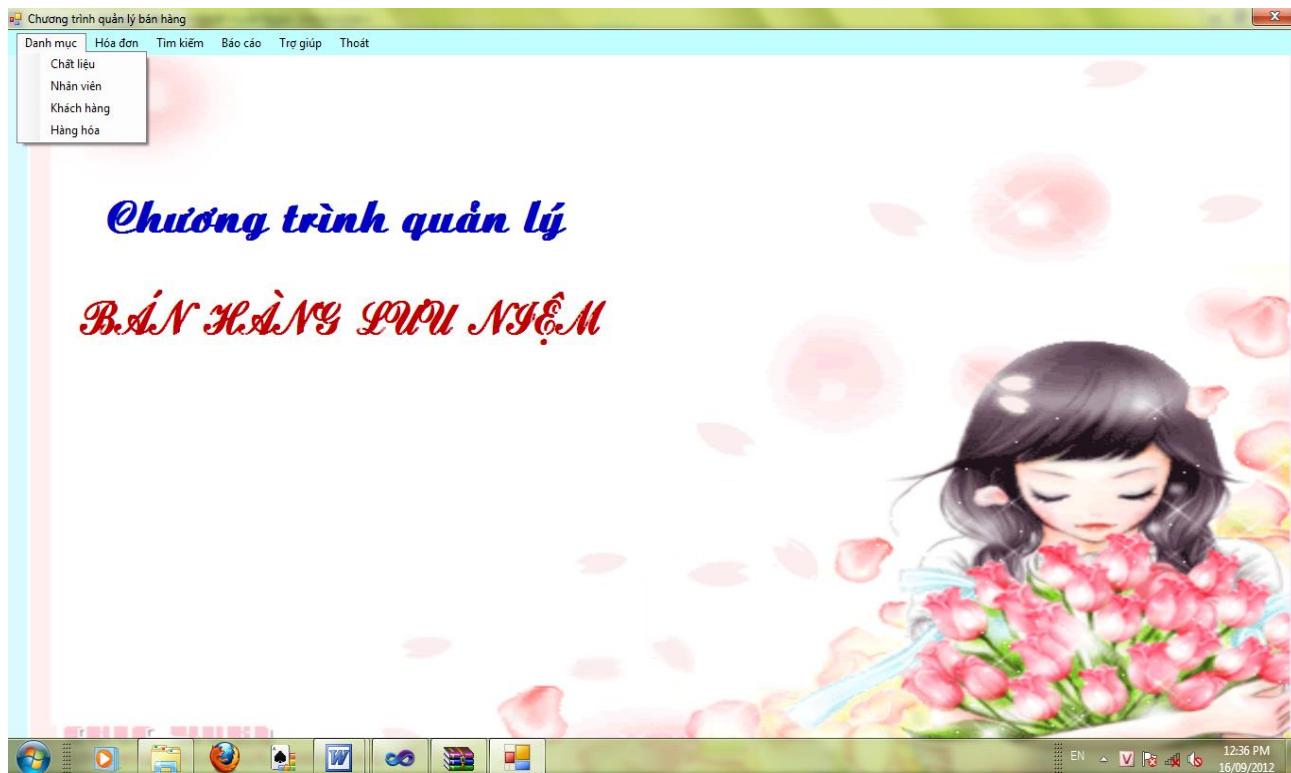
Kích chuột phải tại *DataBase Diagrams* chọn *Add New Diagram* và tạo quan hệ cho CSDL như hình dưới đây:



Bài giảng lập trình trực quan

Bài tập 27:

Trong project QuanLyBanHang tạo form frmMain có giao diện như sau:



Trong form trên có các điều khiển chính:

+ Tạo menu cho chương trình gồm các mục menu với thuộc tính Name và Text như sau:

Danh mục *mnuDanhMuc*

Chất liệu *mnuChatLieu*

Nhân viên *mnuNhanVien*

Khách hàng *mnuKhachHang*

Hàng hóa *mnuHangHoa*

Hóa đơn *mnuHoaDon*

Hóa đơn bán *mnuHDBan*

Tìm kiếm *mnuTimKiem*

Hóa đơn *mnuTKHoaDon*

Hàng *mnuTKHang*

Bài giảng lập trình trực quan

Khách hàng *mnuTKKhach*

Báo cáo *mnuBaoCao*

Hàng tồn *mnuBCHangTon*

Doanh thu *mnuBCDoanhThu*

Trợ giúp *mnuTroGiup*

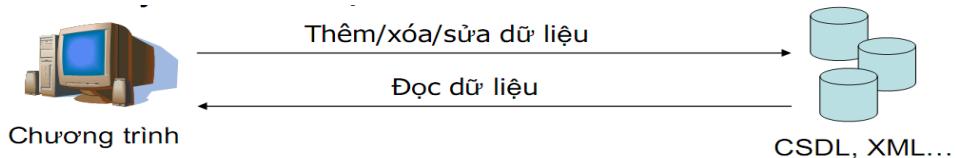
Thoát *mnuThoat*

+ Thêm 2 nhãn Label với Text là: ‘Chương trình quản lý’ và ‘Bán hàng lưu niệm’

7.3. ADO.NET (ActiveX Data Objects for .NET Framework)

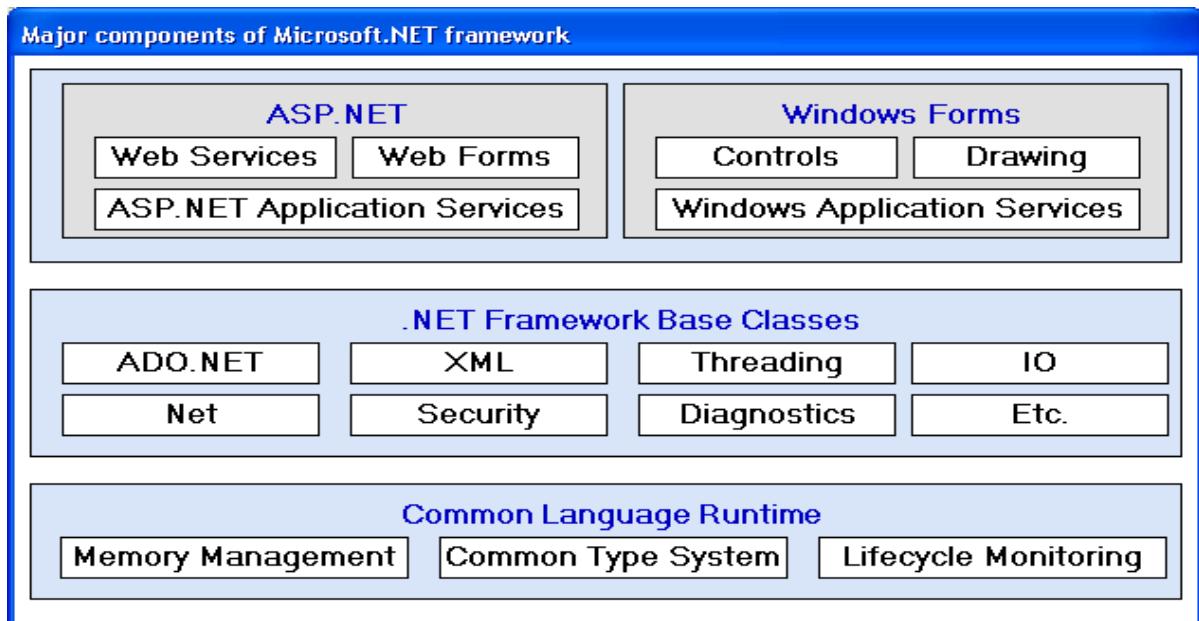
7.3.1. Giới thiệu ADO.NET

ADO.NET là một công nghệ truy cập dữ liệu, nó gồm các lớp nằm trong bộ thư viện lớp cơ sở của .NET Framework để cho phép các ứng dụng Windows (như c#, VB.net) hay các ứng dụng Web (như ASP.Net) thao tác dễ dàng với các nguồn dữ liệu.



Vị trí của ADO.NET trong công nghệ .NET Framework

ADO.NET nằm trong tầng các lớp cơ sở của .NET Framework (.NET Framework Base Classes). Nó nằm giữa tầng Common Language Runtime và tầng ứng dụng.



Bài giảng lập trình trực quan

Từ kiến trúc này ta thấy rằng ADO.NET là một phần nội tại của .NET Framework, do vậy nó có thể được sử dụng trong tất cả các ngôn ngữ hỗ trợ .NET như C#, VB.Net... mà không có sự khác biệt nào (Tức là các chức năng cũng như cách sử dụng hoàn toàn giống nhau).

Đặc điểm chính của ADO.NET

Khả năng làm việc với dữ liệu không kết nối. Dữ liệu được lưu trữ trong bộ nhớ như một CSDL thu nhỏ (DataSet). Nhằm tăng tốc độ xử lý tính toán và hạn chế sử dụng tài nguyên. Trên mô hình ADO.NET, các ứng dụng sẽ kết nối với nguồn dữ liệu khi đọc hoặc cập nhật dữ liệu, sau đó thì cho đóng đường dây kết nối lại. Điểm này rất quan trọng vì trong các ứng dụng Client – server hoặc phân tán, việc để mở liên tục các đường dây kết nối là một tiêu hao nguồn lực vô lối.

Khả năng xử lý dữ liệu chuẩn XML (có thể trao đổi với bất kỳ hệ thống nào). XML là một ngôn ngữ tổng quát định dạng dữ liệu thông qua các thẻ tự định nghĩa, nó là một chuẩn mới mà ngày càng được định dạng rộng rãi.

So sánh ADO.NET với các công nghệ ADO

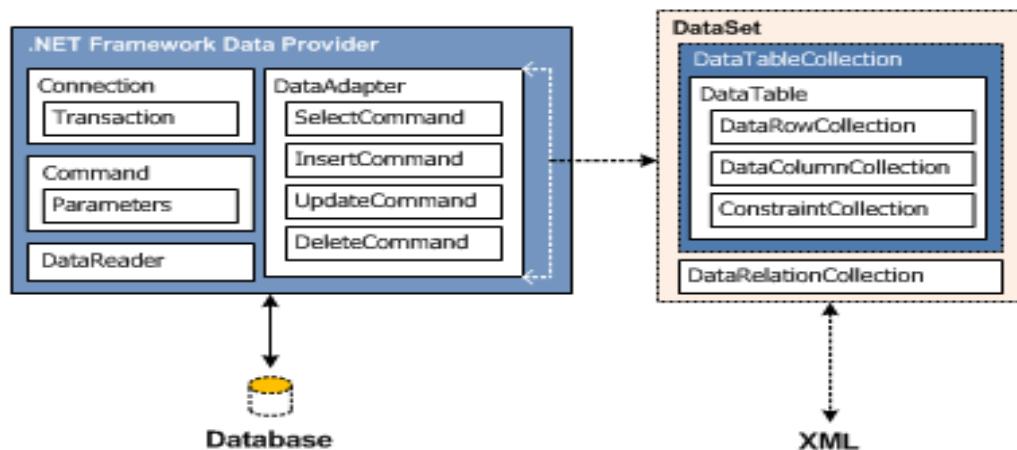
ADO	ADO.NET
Sử dụng RecordSet giống như một bảng dữ liệu	Sử dụng DataSet có thể bao hàm một hay nhiều bảng thực hiện bởi đối tượng DataTable.
Duyệt mẫu tin tuần tự trên Recordset	Cho phép duyệt không tuần tự nhiều mẫu tin trên bảng dữ liệu. Cho phép định hướng một số mẫu tin trên bảng này với một số mẫu tin trên bảng khác.
Sử dụng OLEDB Provide để kết nối cơ sở dữ liệu	Sử dụng đối tượng Connection, DataSet, Command, DataAdapter
Sử dụng kết nối dữ liệu trong một thời gian dài	Ngắt kết nối cơ sở dữ liệu

7.3.2. Kiến trúc của ADO.NET

Ứng dụng thì được kết nối với CSDL (DataBase) thông qua các công nghệ kết nối như ADO, DAO, ADO.NET, ... Ở đây ta quan đến ADO.NET



Kiến trúc của ADO.NET được thể hiện trong hình vẽ sau:



Kiến trúc ADO.NET có thể chia ra làm 2 phần chính là:

1. .NET Framework Data provider

Thành phần này giữ nhiệm vụ làm việc trực tiếp với DataBase. Nó bao gồm các thành phần sau:

- + **Connection**: Đối tượng cho phép bạn kết nối tới CSDL như: SQL server, Oracle, ODBC, OLEDB, MySQL.
- + **Command**: Đối tượng cho phép bạn truy cập cơ sở dữ liệu và thực thi câu lệnh SQL hay các store procedure.
- + **DataReader**: Bộ đọc dùng để đọc nhanh dữ liệu theo một chiều
- + **DataAdapter**: Cầu nối giữa đối tượng DataSet, DataTable với CSDL.

2. DataSet

Đây là một loại dữ liệu tách rời hẳn với DataBase. Nó là bản sao thu nhỏ của CSDL bao gồm các DataTable và các mối quan hệ (DataRelation)

7.3.3. Các bước làm việc với CSDL sử dụng ADO.NET

Bước 1: Tạo kết nối (thiết lập chuỗi kết nối, khai báo và tạo Connection)

Bước 2: Mở kết nối dữ liệu

Bước 3: Tạo lệnh SQL

Bước 4: Thực thi lệnh SQL

Bước 5: Đóng kết nối và hủy các đối tượng (nếu cần)

7.3.4. Các đối tượng trong ADO.NET và một số phương thức

Tùy kiến trúc của ADO.NET, ta có thể thấy rằng ADO.NET có các đối tượng chính là: Connection, Command, DataReader, DataAdapter, DataSet. Sau đây ta sẽ đi tìm hiểu từng đối tượng một.

Các namespaces trong ADO.NET để làm việc với cơ sở dữ liệu là:

- + System.Data
- + System.Data.OleDb: Sử dụng với Access
- + System.Data.SqlClient: Sử dụng với SQLServer
- + System.Data.OracleClient (Sử dụng với Oracle)
- + System.Data.Odbc (Thông qua ODBC của HĐH)

1. Connection

Vai trò của Connection trong ADO.NET là tạo kết nối giữa ứng dụng với cơ sở dữ liệu.

Khi thực hiện kết nối cơ sở dữ liệu ta phải khai báo các thông tin cho Connection trong chuỗi kết nối. Tùy thuộc vào từng loại CSDL khác nhau mà chuỗi kết nối cũng khác nhau.

Tùy từng loại CSDL khác nhau mà ta dùng các Connection khác nhau như: SqlConnection, OleDbConnection, OracleConnection, OdbcConnection.

Khi thực hiện kết nối cần khai báo các thông tin cho Connection thông qua các thuộc tính trong chuỗi kết nối. Tùy từng loại CSDL khác nhau mà chuỗi kết nối cũng khác nhau như:

Nếu kết nối với CSDL Access

Provider: Khai báo Data Provider của Access

(Provider=Microsoft.Jet.OLEDB.4.0)

Bài giảng lập trình trực quan

Data Source: Đường dẫn đến tên tệp tin CSDL (.mdb)

User ID: Tên người dùng

Password : Mật khẩu

Nếu kết nối với CSDL SQL server

Data Source/Server: Tên Server

Initial Catalog/DataBase: Tên CSDL

User ID/UID: Tên người dùng

Password/ PWD: Mật khẩu

Integrated Security: Cơ chế chứng thực đăng nhập

+ true: tài khoản Windows;

+ false: Tài khoản SqlServer (ví dụ: sa)

Khai báo và khởi tạo Connection

<Loại Connection> <Biến Connection>;

<Biến Connection>=new <Loại Connection>(chuỗi kết nối);

Vừa khai báo vừa khởi tạo:

<Loại Connection> <Biến Connection>=new <Loại Connection>(chuỗi kết nối)

Các phương thức:

+ Close: Đóng kết nối

+ Open: Thực hiện kết nối

+ Dispose: Hủy đối tượng

Các thuộc tính của Connection:

+ State: Trạng thái của Connection:

+ Open: Đã kết nối

+ Closed: Đã đóng kết nối

+ Executing: Kết nối đang thực hiện một lệnh

+ Fetching: Kết nối đang truy xuất dữ liệu

+ Connecting: Đang thực hiện kết nối

2. Command

Sau khi tạo kết nối CSDL, mọi thao tác với nguồn dữ liệu (như thêm, sửa, xóa dữ liệu) có thể được thông qua Command.

Cú pháp tạo Command:

```
<Loại Command> <Biến Command>;  
<Biến Command>.Connection=<Biến Connection >;  
<Biến Command>.CommandText=<Lệnh SQL>;
```

Hoặc

```
<Loại Command> <Biến Command> = New <Loại Command>(<Lệnh SQL>)  
<Biến Command>.Connection=<Biến Connection >
```

Mỗi loại CSDL có những loại Command khác nhau, các loại Command thường dùng như: SqlCommand, OleDbCommand, OracleCommand, OdbcCommand.

Thuộc tính hay dùng của Command

- + CommandText: Lệnh SQL hay tên Procedure
- + Connection: Đối tượng Connection

Các phương thức:

+ *ExecuteReader*: Trả về đối tượng DataReader để đọc dữ liệu mỗi lần một dòng với phương thức Read. Cú pháp:

```
<Loại DataReader> <Tên DataReader>;  
<Tên DataReader> = <tên Command>.ExecuteReader;
```

VD:

```
SqlDataReader rd ;  
rd = cmd.ExecuteReader;
```

Tương tự như các đối tượng Connection hay Command thì mỗi loại CSDL sẽ tương ứng với các loại DataReader khác nhau. Nội dung này sẽ được nói đến trong phần DataReader ngay phía dưới.

- + *ExcuteNoneQuery*: Dùng thực thi các phát biểu T-Sql như: Insert, Update, Delete, Create Table,...
- + *ExcuteScalar*: Trả về phát biểu SQL chỉ có một hàng, một cột (1 giá trị đơn)

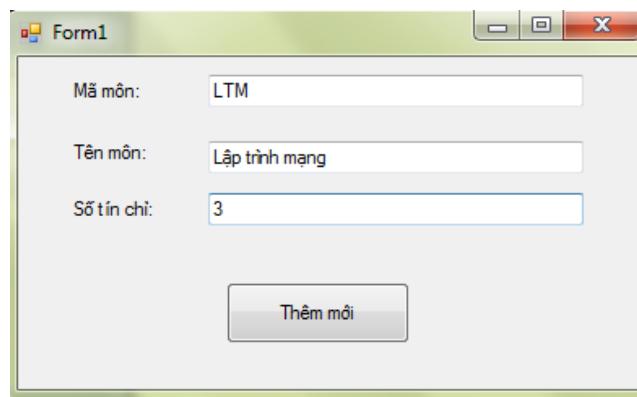
Bài giảng lập trình trực quan

++ Dispose: Hủy đối tượng

Ví dụ : Cho bảng CSDL Access tblMon có cấu trúc như sau :

tblMon : Table		
	Field Name	Data Type
1	MaMon	Text
2	TenMon	Text
3	SoTC	Number

Viết chương trình thêm mới một môn học có giao diện như sau :



Khi ta điền đầy đủ thông tin vào form rồi nhấn nút Thêm mới thì dữ liệu được thêm mới vào CSDL.

Hướng dẫn

Trong ví dụ này file DuLieu.mdb nằm trong thư mục /bin.Debug/DataBase của dự án.

Khai báo thêm 2 namespace sau :

```
using System.Data;
using System.Data.OleDb; //Làm việc với CSDL Access
```

Viết mã lệnh cho sự kiện click của nút thêm mới làm việc theo các bước của ADO.NET

như

sau:

<https://www.youtube.com/watch?v=HQuWC80sLBI>

```
private void btnThemMoi_Click(object sender, EventArgs e)
{
    //Bước 1: Tạo kết nối: Thiết lập chuỗi kết nối, khai báo và tạo connection
    string strConnect = "Provider=Microsoft.Jet.OLEDB.4.0;" +
        "Data Source= DataBase\DuLieu.mdb;" +
        "Persist Security Info=True";
    OleDbConnection conection = new OleDbConnection(strConnect);
    //Bước 2: Mở kết nối
    if (conection.State != ConnectionState.Open)
```

```
        conection.Open();
        //Bước 3: Tạo lệnh SQL
        string sql = "insert into tblMon(MaMon,TenMon,SoTC) "+
        " values ('"+txtMaMon.Text+"','"+txtTenMon.Text+"','"+txtSoTC.Text+"')";
        //Bước 4: Thực thi lệnh SQL sử dụng đối tượng OLEDBCommand
        OleDbCommand command = new OleDbCommand();
        command.Connection = conection; //Chi định đối tượng kết nối
        command.CommandText = sql; //Truyền lệnh sql cho thuộc tính CommandText
        command.ExecuteNonQuery(); //Thực thi lệnh sql
        //Bước 5: Đóng kết nối
        if (conection.State != ConnectionState.Closed)
            conection.Close();
        conection.Dispose(); //Hủy đối tượng
        command.Dispose(); //Hủy đối tượng Command
    }
```

3. 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.

Dùng để đón nhận kết quả trả về từ phương thức ExecuteReader của đối tượng Command. Nó tương tự như một Recordset của ADO, tuy nhiên dữ liệu là Readonly và chỉ đọc theo chiều tiến.

Tùy từng loại CSDL khác nhau mà ta dùng các DataReader khác nhau như: SqlDataReader, OleDbDataReader, OracleDataReader, OdbcDataReader.

```
//Khai báo một DataReader, không có new
<Loại DataReader> <Tên DataReader>;
//Lấy kết quả từ Command
<Tên DataReader> = <tên Command>.ExecuteReader();
```

Các thuộc tính

- + FieldCount: Số cột trên dòng hiện hành của DataReader
- + IsClosed : Cho biết dataReader đã đóng

Các phương thức

- + GetFieldType: Trả về kiểu dữ liệu của tham số truyền vào
- + GetName: Trả về tên của cột truyền vào (cột là số TT cột)
- + GetValue: Trả về trị của cột truyền vào (cột là số TT cột)
- + Read: Di chuyển đến dòng kế tiếp và trả về true nếu còn dòng để di chuyển, ngược lại trả về false.

Bài giảng lập trình trực quan

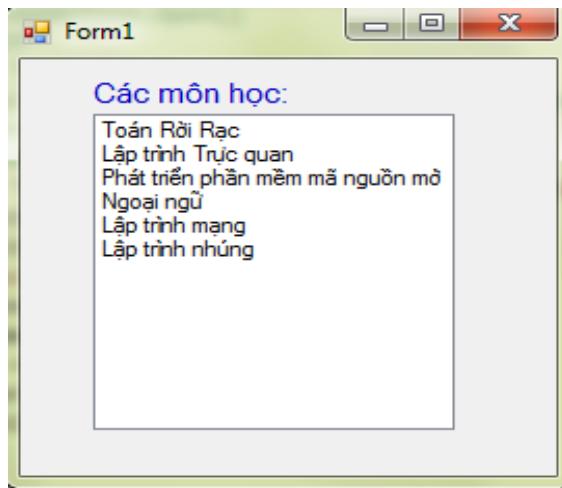
+ Close: đóng DataReader

+ Dispose: Hủy đối tượng

Trong khi dataReader đang mở các thao tác dữ liệu trên nguồn dữ liệu đều không thể cho đến khi dataReader đóng lại bằng lệnh Close.

Ví dụ: Vẫn lấy CSDL Access ở trên. Viết đoạn chương trình sử dụng DataReader đọc dữ liệu trong bảng tblMon đổ ra ListBox như hình minh họa sau:

Tạo giao diện form gồm các điều khiển: Label có Text là: "Các môn học"; điều khiển ListBox có Name là: lstMonHoc.



Hướng dẫn:

Khai báo thêm 2 namespace sau :

```
using System.Data;
using System.Data.OleDb; //Làm việc với CSDL Access
```

Viết mã lệnh cho sự kiện Load của Form làm việc theo các bước của ADO.NET như sau:

```
private void frmDataReader_Load(object sender, EventArgs e)
{
    //Bước 1: Tạo kết nối: Thiết lập chuỗi kết nối, khai báo và tạo connection
    string strConnect = "Provider=Microsoft.Jet.OLEDB.4.0;" +
        "Data Source=DataBase\DuLieu.mdb;" +
        "Persist Security Info=True";
    OleDbConnection conection = new OleDbConnection(strConnect);
    //Bước 2: Mở kết nối
    if (conection.State != ConnectionState.Open)
        conection.Open();
    //Bước 3: Tạo lệnh SQL
    string sql = "Select * from tblMon";
    //Bước 4: đọc kết quả với DataReader từ đối tượng Command
    OleDbCommand command = new OleDbCommand();
```

Bài giảng lập trình trực quan

```
command.Connection = conection; //Chỉ định đối tượng kết nối
command.CommandText = sql; //Truyền lệnh sql cho thuộc tính CommandText
OleDbDataReader dataReader;
dataReader = command.ExecuteReader();
//Duyệt các bản ghi trong DataReader điền vào listBox
while (dataReader.Read() == true)
    lstMonHoc.Items.Add(dataReader.GetValue(1));
//Bước 5: Đóng kết nối
if (conection.State != ConnectionState.Closed)
    conection.Close();
conection.Dispose(); //Hủy đối tượng connection
dataReader.Close(); //đóng dataReader
dataReader.Dispose() //Hủy đối tượng dataReader
}
```

4. DataSet

DataSet là mô hình CSDL quan hệ thu nhỏ tách rời hẳn với phía server đáp ứng các nhu cầu với ứng dụng.

DataSet chứa các bảng (DataTable), các quan hệ (DataRelation) và các ràng buộc (Constraint)

Phương thức hay dùng:

+ Thêm một bảng vào Dataset. Một bảng mới tự động được tạo ra với tên mặc định Table1, Table2 . . .

Tables.Add()

+ Một bảng mới tạo ra theo đúng <tên bảng>. Tên bảng có phân biệt chữ in, thường

Tables.Add(<Tên bảng>)

+ Xóa bảng ra khỏi Dataset

Tables.Remove(<Tên bảng>)

+ Kiểm tra bảng có thuộc Dataset không

Tables.Contains(<Tên bảng>)

+ Lấy chỉ số của bảng

Tables.IndexOf[<tên bảng>]

+ Lấy số bảng trong Dataset

Tables.Count

+ Lấy ra một bảng trong Dataset

Tables[<Chỉ số>]

Bài giảng lập trình trực quan

+ Để xóa bỏ mọi dữ liệu trên dataSet

Clear()

+ Để tạo một bản sao của Dataset

Clone()

+ Xóa bỏ Dataset. Giải phóng mọi tài nguyên trên vùng nhớ Dataset đang sử dụng.

Dispose()

5. DataAdapter

Để lấy dữ liệu từ nguồn dữ liệu về cho ứng dụng, chúng ta sử dụng đối tượng DataAdapter. Đối tượng này cho phép ta lấy cấu trúc và dữ liệu của các bảng.

Mỗi loại CSDL ta sẽ phải dùng tương ứng với một loại DataAdapter khác nhau: SqlDataAdapter, OdbcDataAdapter, OleDbDataAdapter, OracleDataAdapter

Cú pháp tạo DataAdapter:

```
<Loại DataAdapter> <Biến DataAdapter> = New <Loại DataAdapter>(<Lệnh sql>, <Biến Connection>)
```

Chú ý: DataAdapter chỉ thao tác với nguồn dữ qua đối tượng Connection đang kết nối.

Các thuộc tính của DataAdapter

+ DeleteCommand: Xóa dữ liệu, tương ứng với lệnh DELETE

+ InsertCommand: Chèn dữ liệu mới, tương ứng với lệnh INSERT

+ UpdateCommand: Cập nhật dữ liệu, tương ứng với lệnh UPDATE

+ SelectCommand: Truy vấn dữ liệu, tương ứng với lệnh SELECT trong SQL

Phương thức

Đối tượng DataAdapter có 2 phương thức cơ bản:

+ Fill: Thực thi câu lệnh SelectCommand và đổ dữ liệu vào đối tượng DataTable. Cú pháp:

```
Fill(<DataSet>, <Tên dataTable>)
```

+ Update: Thực thi các câu lệnh InsertCommand, UpdateCommand, DeleteCommand, ... Nói chung là các câu lệnh làm thay đổi dữ liệu. Cú pháp:

Update(<mảng dòng>): Cập nhật các dòng (Các đối tượng DataRow) vào nguồn dữ liệu.

Bài giảng lập trình trực quan

Update (<Dataset>) : Cập nhật các thay đổi trên tất cả các bảng của Dataset vào nguồn dữ liệu.

Update (<DataTable>) : Cập nhật tất cả các thay đổi trên DataTable vào nguồn dữ liệu.

Update (<Dataset>, <Tên bảng>) Cập nhật các thay đổi trên bảng trong Dataset vào nguồn dữ liệu.

6. Điều khiển DataGridView

DataGridView là một cấu trúc bảng gồm các cột, các dòng. Nó dùng để hiển thị dữ liệu dưới dạng bảng.

Thuộc tính:

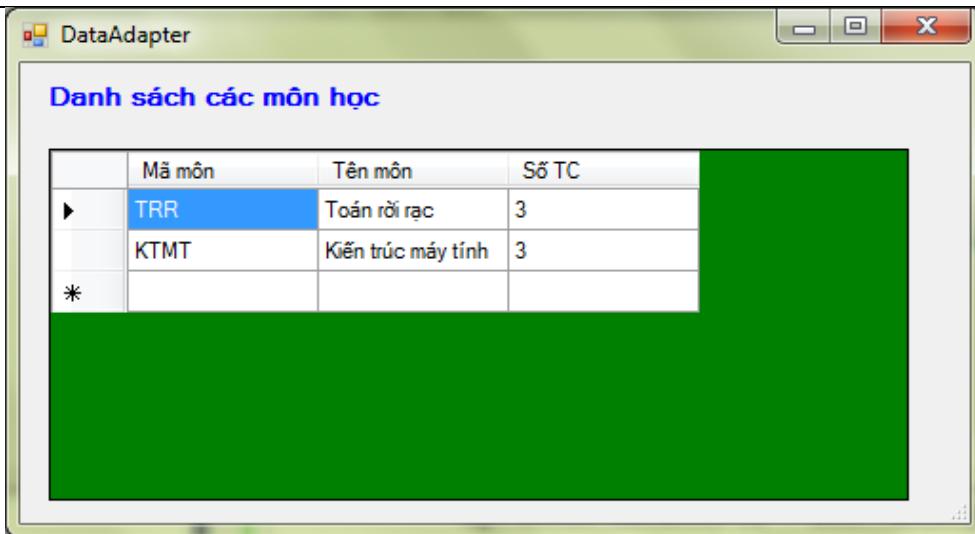
- + Name : Tên của điều khiển, với tiếp đầu ngữ là dgv
- + DataSource: Truyền vào một DataTable chứa dữ liệu
- + Thuộc tính **Columns**: liên quan đến việc thiết lập cho các cột của Grid, thiết lập cột nào thì ta truyền chỉ số của cột đó vào bắt đầu từ 0. Ví dụ: Columns(0) – Thiết lập cho cột thứ 1,... Có các thuộc tính thiết lập cho Columns như sau:
 - *HeaderText*: Thiết lập chuỗi tiêu đề cho cột
 - *DataPropertyName*: Thiết lập tên trường dữ liệu hiển thị cho cột.
 - *Width*: Thiết lập độ rộng cho cột
- + ReadOnly: Thuộc tính thiết lập là true nếu grid chỉ để đọc ngược lại là false.
- + BackGroundColor: Thuộc tính quy định màu nền cho grid (dùng lớp Color để chọn màu cho Grid)
- + ForeColor: Quy định màu chữ cho Grid
- + Font: Quy định về Font chữ cho Grid (như kiểu chữ, màu chữ, ...)

Các sự kiện

- + Click: Được kích hoạt khi người dùng click vào lưới

Ví dụ: Vẫn dùng CSDL Access ở trên. Viết đoạn chương trình sử dụng DataAdapter đọc CSDL và liệt kê danh sách các môn học có trong CSDL như hình sau :

Bài giảng lập trình trực quan



Hướng dẫn :

Tạo giao diện form như trên gồm các điều khiển: Lable có Text là: "Danh sách các môn học"; điều khiển DataGridView có Name là: dgvMonHoc.

Khai báo thêm 2 namespace sau :

```
using System.Data;
using System.Data.OleDb; //Làm việc với CSDL Access
```

Viết mã lệnh cho sự kiện Load của Form làm việc theo các bước của ADO.NET như sau:

```
private void frmDataAdapter_Load(object sender, EventArgs e)
{
    //Bước 1: Tạo kết nối: Thiết lập chuỗi kết nối, khai báo và tạo connection
    string strConnect = "Provider=Microsoft.Jet.OLEDB.4.0;" +
        "Data Source=DataBase\DuLieu.mdb;" +
        "Persist Security Info=True";
    OleDbConnection conection = new OleDbConnection(strConnect);
    //Bước 2: Mở kết nối
    if (conection.State != ConnectionState.Open)
        conection.Open();
    //Bước 3: Tạo lệnh SQL
    string sql = "Select * from tblMon";
    //Bước 4: Sử dụng đối tượng DataAdapter thực thi lệnh Select đổ ra DataTable
    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(sql, conection);
    DataSet dts = new DataSet();
    dataAdapter.Fill(dts, "dtMon"); //Đổ dữ liệu vào dataTable dtMon trong
    dataset dts
    //Bước 5: Đóng kết nối
    if (conection.State != ConnectionState.Closed)
        conection.Close();
    conection.Dispose(); //Hủy đối tượng connection
    //Lấy dữ liệu ra datagridview
    dgvMonHoc.DataSource = dts.Tables[0];
    //Trình bày dataGridView
    dgvMonHoc.Columns[0].HeaderText = "Mã môn";
    dgvMonHoc.Columns[1].HeaderText = "Tên môn";
```

```
dgvMonHoc.Columns[2].HeaderText = "Số TC";
dgvMonHoc.BackgroundColor = Color.Green;
    //Hủy đổi tượng DataSet
dts.Dispose();
}
```

7.4. Làm việc với CSDL SQL qua các đối tượng: SqlConnection, SqlDataAdapter, SqlCommand.

Mỗi loại CSDL khác nhau thì ta dùng các đối tượng Connection, DataAdapter và Command khác nhau. Ở phần này chúng ta chỉ đề cập đến một loại CSDL đó chính là SQL server. Khi dùng Sql server thì các đối tượng tương ứng của nó sẽ có là: SqlConnection, SqlDataAdapter, SqlCommand.

7.4.1. Đối tượng SqlConnection

Đối tượng này có chức năng kết nối với cơ sở dữ liệu Sql server nhờ các thuộc tính và phương thức mà chúng có.

Để kết nối CSDL SQL server, chúng ta làm theo các bước như sau:

Bước 1: Khai báo sử dụng không gian tên:

```
using System.Data;
using System.Data.SqlClient;
```

Bước 2: Khai báo đối tượng SqlConnection

```
SqlConnection <tên đối tượng SqlConnection>;
```

ví dụ:

```
SqlConnection Conn;
```

Bước 3: Khai báo chuỗi kết nối

Ví dụ ta dùng CSDL SQL server được hỗ trợ sẵn trong VS 2010 thì ta sẽ tìm được chuỗi kết nối bằng cách:

+ Kích vào tên CSDL ở Server explorer.

+ Chuỗi kết nối chính là thuộc tính ConnectinString của CSDL. Ở CSDL **DuLieu.mdf** có chuỗi kết nối như sau:

```
Data source =
.\SQLEXPRESS;AttachDbFilename=E:\QuanLybanHang\bin\Debug\ DataBase\DuLieu.mdf;Integrated
Security=True;User Instance=True;
```

Ta thấy với đường dẫn trên là đường dẫn tuyệt đối và liên kết nên sẽ gặp rắc rối nếu ta chuyển ứng dụng sang máy khác. Nên ta sử dụng đường dẫn tương đối bằng cách dùng: Với

Bài giảng lập trình trực quan

`System.IO.Directory.GetCurrentDirectory().ToString()` để lấy ra đường dẫn của ứng dụng nơi chứa file chạy của chương trình. Sau đó dẫn đến `|DataBase|DuLieu.mdf` như sau:

```
string strConnect = "Data Source=.\SQLExpress;AttachDbFilename=" +
    System.IO.Directory.GetCurrentDirectory().ToString() + "\\DataBase\\"
    "DuLieu.mdf;Integrated Security=True";
```

Bước 4: Khởi tạo đối tượng SqlConnection

```
Conn = New SqlConnection(strConnect);
```

Bước 5: Mở kết nối. Trước khi mở cần kiểm tra xem kết nối đã được mở chưa?

```
if (Conn.State != ConnectionState.Open)
    Conn.Open();
```

Bước 6: Thiết lập câu lệnh SQL và thực thi nó thông qua các đối tượng SqlDataAdapter hoặc SqlCommand

Bước 7: Đóng kết nối và hủy các đối tượng (nếu có)

```
if (Conn.State != ConnectionState.Closed)
    Conn.Close();
Conn.Dispose();
//Hủy các đối tượng khác nếu có
```

7.4.2. Đối tượng SqlDataAdapter và SqlCommand

1. Dùng SqlDataAdapter

Ta dùng SqlDataAdapter như một thành phần điều phối dữ liệu, tức nó không lưu trữ dữ liệu mà chỉ phát đi các lệnh SQL và nhận dữ liệu trả về của CSDL sau đó chuyển cho đối tượng khác (thường là DataTable)

Thông thường ta dùng SqlDataAdapter để lấy dữ liệu khi thực hiện câu lệnh Select đổ ra một DataTable.

Các bước thực hiện như sau:

- + Khai báo chuỗi lệnh SQL
- + Khai báo và khởi tạo một SqlDataAdapter với 2 đối truyền vào là câu lệnh Sql và đối tượng SqlConnection

```
SqlDataAdapter DA = new SqlDataAdapter(strsql, Conn); //strsql là chuỗi câu lệnh sql,  
Conn là tên đối tượng SqlConnection
```

- + Khai báo và khởi tạo một DataTable

```
DataTable DT = new DataTable();
```

- + Đổ dữ liệu vào DataTable thông qua phương thức Fill của SqlDataAdapter

```
DA.Fill(DT)
```

Bài giảng lập trình trực quan

Ngoài thực thi câu lệnh Select thi đối tượng này cũng thực thi các câu lệnh Insert, update, delete,.. tuy nhiên ta hay dùng SqlCommand để làm điều đó hơn.

2. Dùng SqlCommand

Cũng như SqlDataAdapter thì sau khi kết nối CSDL xong thì mọi thao tác trên CSDL SQL server có thể được thực hiện thông qua SqlCommand. Các bước tiến hành như sau:

- + Khai báo và khởi tạo đối tượng SqlCommand

```
SqlCommand sqlcommand = new SqlCommand();
```

- + Truyền đối tượng SqlConnection cho thuộc tính Connection.

```
sqlcommand.Connection = Conn;
```

- + Truyền chuỗi lệnh Sql vào cho thuộc tính CommandText

```
sqlcommand.CommandText = strsql; //strsql là một chuỗi sql
```

- + Thực thi lệnh sql bằng phương thức **ExecuteNonQuery()**

```
sqlcommand.ExecuteNonQuery();
```

7.4.3. Các thao tác dữ liệu

Ở đây ta sẽ học các thao tác dữ liệu hay dùng là: truy vấn, thêm mới, sửa, xóa dữ liệu. Ta sẽ học phần này thông qua việc phân tích một ví dụ cụ thể.

Trong Project **QuanLyBanHang**, trong thư mục Forms ta tạo một form mới đặt tên là frmChatLieu:

- + Form này cho phép chúng ta cập nhật các danh mục chất liệu.
- + Bảng CSDL liên quan là: tblChatLieu.

Giao diện của form như sau:

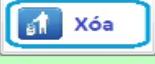
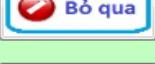
Bài giảng lập trình trực quan

Cập nhật danh mục chất liệu

DANH SÁCH CÁC CHẤT LIỆU

Mã chất liệu:	CLM
Tên chất liệu:	Mây

	Mã chất liệu	Tên chất liệu
	CLB	Bông
	CLG	Giấy
▶	CLM	Mây
	CLN	Nhựa
	CLS	Sắt
	CLV	Vải
*		

 Thêm
 Lưu
 Sửa
 Xóa
 Bỏ qua
 Thoát

Các điều khiển dùng trong form là :

Điều khiển	Name	Text
Form	frmChatLieu	Cập nhật danh mục chất liệu
TextBox	txtMaCL	
	txtTenCL	
Button	btnThemMoi	Thêm mới
	btnSua	Sửa
	btnXoa	Xóa
	btnBoQua	BỎ qua
	btnThoat	Thoát
DataGridView	dgvChatLieu	

Bài giảng lập trình trực quan

Nhận xét: Ta thấy mỗi lần muốn thực thi một câu lệnh SQL dạng select, update, delete hay insert ta đều phải thực hiện trình tự các công việc giống nhau chỉ khác nhau về câu lệnh SQL. Vì vậy để thuận tiện, ta nên xây dựng lớp xử lý dữ liệu đảm nhận việc kết nối CSDL và việc xử lý, cập nhật dữ liệu.

Lớp xử lý dữ liệu gồm các phương thức sau đây:

- + Phương thức **KetNoiCSDL()** nhằm thực hiện kết nối tới CSDL.
- + Phương thức **DongKetNoiCSDL()** nhằm đóng lại kết nối CSDL.
- + Phương thức **Docbang(string strSql)** nhằm thực hiện câu lệnh Select trả về một DataTable.
- + Phương thức **CapNhatDuLieu(string strSql)** nhằm thực hiện các câu lệnh Insert, Update, Delete để cập nhật dữ liệu cho CSDL.

Trong cửa sổ Solution Explorer tạo một thư mục mới đặt tên là Class, kích chuột phải tại Class chọn Add/New Item... chọn Class trong hoopk thoại Add New Item và đặt tên cho Class là *ProcessDataBase.cs* tại ô Name.

Khai báo thêm 2 namespace sau :

```
using System.Data;
using System.Data.SqlClient;
```

Sau đó xây dựng lớp xử lý cơ sở dữ liệu như sau:

```
public class ProcessDataBase
{
    string strConnect = "Data Source=.\SQLExpress;AttachDbFilename=" +
        System.IO.Directory.GetCurrentDirectory().ToString() + "\\Data\\Data\\DuLieu.mdf;Integrated Security=True";
    SqlConnection sqlConnect = null;

    //Hàm mở kết nối CSDL
    private void KetNoiCSDL()
    {
        sqlConnect = new SqlConnection(strConnect);
        if (sqlConnect.State != ConnectionState.Open)
            sqlConnect.Open();
    }

    //Hàm đóng kết nối CSDL
    private void DongKetNoiCSDL()
    {
        if (sqlConnect.State != ConnectionState.Closed)
            sqlConnect.Close();
        sqlConnect.Dispose();
    }
}
```

Bài giảng lập trình trực quan

```
//Hàm thực thi câu lệnh dạng Select trả về một DataTable
public DataTable DocBang(string sql)
{
    DataTable dtBang=new DataTable ();
    KetNoiCSDL();
    SqlDataAdapter sqldataAdapte = new SqlDataAdapter(sql, sqlConnect);
    sqldataAdapte.Fill(dtBang);
    DongKetNoiCSDL();
    return dtBang;
}

//Hàm thực lệnh insert hoặc update hoặc delete
public void CapNhatDuLieu(string sql)
{
    KetNoiCSDL();
    SqlCommand sqlcommand = new SqlCommand();
    sqlcommand.Connection = sqlConnect;
    sqlcommand.CommandText = sql;
    sqlcommand.ExecuteNonQuery();
    DongKetNoiCSDL();
}
}
```

Sau đó trong mỗi lớp sử dụng lớp *ProcessDataBase* ở trên ta phải khai báo khởi tạo đối tượng này như sau:

```
ProcessDataBase dtBase=new ProcessDataBase();
```

7.4.3.1. Viết sự kiện Load của form

Khi bắt đầu load form nên ta sẽ phải hiển thị hết các danh mục chất liệu trong bảng *tblChatlieu* lên *DataGridView*. Để làm điều này ta viết chương trình cho sự kiện Load của form Chất liệu theo các bước sau:

- + Bước 1: Gọi phương thức **DocBang** trong lớp *DataBase* ở trên với câu lệnh SQL truyền vào là: *Select * from tblChatlieu*. Hàm này sẽ trả về một *DataTable* chứa danh sách các Chất liệu.
- + Bước 2: Gán *DataTable* lấy được ở bước 1 gán vào thuộc tính *DataSource* của *DataGridView*.
- + Bước 3: Giải phóng bộ nhớ cho *DataTable*

Đồng thời khi load Form lên thì người dùng chỉ kích được vào nút Thêm hoặc Thoát; các nút còn lại không kích được. Để làm điều này ta sử dụng thuộc tính *Enabled* của các nút

Bài giảng lập trình trực quan

Đoạn chương trình cụ thể như sau:

```
private void frmChatLieu_Load(object sender, EventArgs e)
{
    //Gọi pt DocBang lấy dữ liệu của bảng tblChatLieu đổ vào DataTable
    DataTable dtChatLieu = dtBase.DocBang("select * from tblChatLieu");
    dgvChatLieu.DataSource = dtChatLieu;
    //Định dạng dataGridView
    dgvChatLieu.Columns[0].HeaderText = "Mã chất liệu";
    dgvChatLieu.Columns[1].HeaderText = "Tên chất liệu";
    dgvChatLieu.Columns[0].Width = 150;
    dgvChatLieu.Columns[1].Width = 250;
    dgvChatLieu.BackgroundColor = Color.LightBlue;
    dtChatLieu.Dispose(); //Giải phóng bộ nhớ cho DataTable

    btnXoa.Enabled = false;
    btnSua.Enabled = false;
    btnLuu.Enabled = false;
    btnBoQua.Enabled = false;
    btnThemMoi.Enabled = true;

}
```

7.4.3.2. Viết sự kiện click cho DataGridView

Khi người dùng kích chuột chọn một dòng dữ liệu trên DataGridView các dữ liệu tương ứng sẽ được hiển thị lên trên 2 TextBox mã chất liệu và tên chất liệu. Đồng thời nút Sửa, xóa, Bỏ qua sáng lên để người dùng có thể chọn sửa, xóa hoặc bỏ qua thao tác này. Nút thêm và lưu không kích được.

Khi sửa dữ liệu ta không sửa mã cho nên ô nhập mã không thể kích hoạt được (dùng thuộc tính Enabled)

```
private void dgvChatLieu_Click(object sender, EventArgs e)
{
    txtMaCL.Text = dgvChatLieu.CurrentRow.Cells[0].Value.ToString();
    txtTenCL.Text = dgvChatLieu.CurrentRow.Cells[1].Value.ToString();

    txtMaCL.Enabled= false;
    btnXoa.Enabled = true;
    btnSua.Enabled = true;
    btnLuu.Enabled = false;
    btnBoQua.Enabled = true;
    btnThemMoi.Enabled = false;
}
```

7.4.3.3. Viết sự kiện click cho nút Sửa

Khi người dùng kích chuột vào một dòng bản ghi bất kỳ trên lưới để hiển thị dữ liệu của bản ghi đó lên Form thì người dùng có thể chỉnh sửa các thông tin đó. Phương thức *btnSua_Click* có tác dụng lưu các thông tin người dùng đã sửa vào CSDL.

Bài giảng lập trình trực quan

```
private void btnSua_Click(object sender, EventArgs e)
{
    if (txtTenCL.Text == "")
    {
        MessageBox.Show("Bạn phải nhập tên chất liệu", "Thông báo",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
        txtTenCL.Focus();
    }
    else
    {
        dtBase.CapNhatDuLieu("update tblChatLieu set TenChatLieu=N'"
                            +txtTenCL.Text+ '' where MaChatLieu='"+txtMaCL.Text+"'");
        ResetValue(); //Xóa dữ liệu ở các ô nhập TextBox
        //Sau khi update cần lấy lại dữ liệu để hiển thị lên lưới
        dgvChatLieu.DataSource = dtBase.DocBang("select * from
                                                tblChatLieu");

        btnXoa.Enabled = false;
        btnSua.Enabled = false;
        btnLuu.Enabled = false;
        btnBoQua.Enabled = false;
        btnThemMoi.Enabled = true;
    }
}
```

ResetValue() là một phương thức nằm trong lớp frmChatLieu dùng để xóa rỗng dữ liệu trong các ô nhập dữ liệu:

```
void ResetValue()
{
    txtMaCL.Text = "";
    txtTenCL.Text = "";
}
```

7.4.3.4. Viết sự kiện click cho nút Xóa

Ta có thể chọn 1 chất liệu trên DataGridView sau đó chọn nút xóa để xóa dữ liệu đi. Ta sẽ viết mã lệnh cho sự kiện Click của nút btnXoa như sau:

- + Ta sẽ hỏi xem người dùng có thực sự muốn xóa không? Nếu người dùng chọn “yes” thì là có. Nếu người dùng chọn không thì sẽ không làm gì cả.
- + Sau khi xóa xong thì hiển thị lại dữ liệu lên lưới và gọi phương thức ResetValue() xóa dữ liệu ở 2 ô TextBox.

Bài giảng lập trình trực quan

Mã lệnh được viết như sau:

```
private void btnXoa_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Bạn có muốn xóa chất liệu có mã là:"+
        txtMaCL.Text+" không?", "Thông báo",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
        System.Windows.Forms.DialogResult.Yes)
    {
        dtBase.CapNhatDuLieu("delete tblChatLieu where MaChatLieu=" +
            txtMaCL.Text + "'");
        dgvChatLieu.DataSource = dtBase.DocBang("Select * from
            tblChatLieu");
        ResetValue();
        btnXoa.Enabled = false;
        btnSua.Enabled = false;
        btnLuu.Enabled = false;
        btnBoQua.Enabled = false;
        btnThemMoi.Enabled = true;
    }
}
```

7.4.3.5. Viết sự kiện click cho nút Thêm

Nút này sẽ xóa rỗng các ô nhập liệu để người dùng có thể nhập mới một chất liệu. Sau đó cho phép người dùng có thể nhập dữ liệu vào ô TextBox nhập mã chất liệu. Làm cho nút Lưu, bỏ qua có thể kích được; nút xóa, sửa, Thêm không kích được nữa.

Mã lệnh cụ thể như sau:

```
private void btnThemMoi_Click(object sender, EventArgs e)
{
    ResetValue();
    txtMaCL.Enabled = true;
    txtMaCL.Focus();

    btnXoa.Enabled = false;
    btnSua.Enabled = false;
    btnLuu.Enabled = true;
    btnBoQua.Enabled = true;
    btnThemMoi.Enabled = true;
}
```

7.4.3.6. Viết sự kiện click cho nút Lưu

Người dùng kích nút Thêm rồi điền đầy đủ thông tin vào các ô nhập dữ liệu, sau đó ấn nút Lưu thì dữ liệu vừa nhập sẽ được bổ sung vào bảng tblChatLieu trong CSDL.

Để làm việc này ta viết mã lệnh thực thi cho sự kiện Click của nút btnLuu như sau:

+ Bước 1: Kiểm tra xem người dùng có nhập đủ hết các thông tin cho 2 ô TextBox nhập mã chất liệu và tên chất liệu không. Nếu chưa nhập thì yêu cầu người dùng nhập và đặt con trỏ về ô cần nhập cho người dùng nhập.

Bài giảng lập trình trực quan

+ Bước 2: Kiểm tra xem dữ liệu người dùng nhập có mã trùng với mã đã có trong CSDL không, nếu trùng thì đặt con trỏ vào ô nhập mã chất liệu để nhập lại.

```
DataTable dtChatLieu = dtBase.DocBang("Select * from tblChatLieu where"+
    " MaChatLieu='"+txtMaCL.Text+"'");
if (dtChatLieu.Rows.Count > 0)
{
    MessageBox.Show("Mã chất liệu này đã tồn tại, bạn hãy nhập mã khác!");
    txtMaCL.Focus();
}
```

+ Bước 3: Sau khi mọi dữ liệu đã được nhập thì gọi thủ tục **ThucThiLenh** trong lớp DataBase để thêm mới chất liệu vào. Với tham số truyền vào thủ tục là lệnh SQL như sau:

```
insert into tblChatlieu(MaChatLieu,TenChatLieu) values(N'" + txtMaCL.Text +
"',N'" & txtTenCL.Text & "')
```

+ Bước 4: Đọc lại dữ liệu từ bảng tblChatLieu đặt lại lên lưới.

+ Bước 5: Gọi phương thức **ResetValue()** xóa rỗng các ô nhập dữ liệu. Làm cho nút Lưu, xóa, sửa, bỏ qua không kích được. Nút Thêm kích được.

Đoạn chương trình cụ thể là:

```
private void btnLuu_Click(object sender, EventArgs e)
{
    if (txtMaCL.Text == "")
    {
        MessageBox.Show("Bạn phải nhập mã chất liệu");
        txtMaCL.Focus();
    }
    else
    {
        if (txtTenCL.Text == "")
        {
            MessageBox.Show("Bạn phải nhập tên chất liệu");
            txtTenCL.Focus();
        }
        else
        {
            DataTable dtChatLieu = dtBase.DocBang("Select * from tblChatLieu
                where"+" MaChatLieu='"+(txtMaCL.Text).Trim()+"'");
            if (dtChatLieu.Rows.Count > 0)
            {
                MessageBox.Show("Mã chất liệu này đã có, hãy nhập mã khác!");
                txtMaCL.Focus();
            }
            else
            {
                dtBase.CapNhatDuLieu("insert into tblChatLieu values(N'" +
                    txtMaCL.Text + "',N'" + txtTenCL.Text + "')");
                MessageBox.Show("Bạn đã thêm mới thành công");
                dgvChatLieu.DataSource = dtBase.DocBang("select * from
                    tblChatLieu");
                ResetValue();
                btnXoa.Enabled = false;
            }
        }
    }
}
```

Bài giảng lập trình trực quan

```
        btnSua.Enabled = false;
        btnLuu.Enabled = false;
        btnBoQua.Enabled = false;
        btnThemMoi.Enabled = true;
    }
}
}
```

7.4.3.7. Viết sự kiện click cho nút Bỏ qua

Nút bỏ qua sẽ xóa rỗng tất cả các ô nhập dữ liệu, trạng thái các nút nhấn trở về như khi form mới được Load lên:

```
private void btnBoQua_Click(object sender, EventArgs e)
{
    ResetValue();
    btnXoa.Enabled = false;
    btnSua.Enabled = false;
    btnLuu.Enabled = false;
    btnBoQua.Enabled = false;
    btnThemMoi.Enabled = true;
}
```

7.4.3.8. Viết sự kiện click cho nút Thoát

Trước khi thoát hỏi người dùng có muốn thoát không? Nếu người dùng muốn thoát kiểm tra xem nếu người dùng đang thêm mới thì hỏi người dùng có lưu bản ghi đang thêm không xong mới thoát.

Mã lệnh như sau:

```
private void btnThoat_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Bạn có muốn thoát không?", "Thông báo",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)
    {
        if (btnLuu.Enabled == true)
        {
            if (MessageBox.Show("Bạn có muốn Lưu lại chất liệu
                đang thêm mới không?", "Thông báo",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)
                btnLuu_Click(sender, e);
            else
                this.Close();
        }
        else
            this.Close();
    }
}
```

Bài giảng lập trình trực quan

Bài tập 28:

Tiếp tục bài tập 27, tạo form cập nhật danh mục Nhân viên có giao diện như sau:

The screenshot shows a Windows application window titled "Cập nhật danh sách nhân viên". The main title is "Danh sách các nhân viên". The interface includes input fields for Employee ID, Name, Address, Birth Date, and Phone Number, along with gender selection buttons. Below this is a data grid displaying employee records:

	Mã NV	Tên NV	Giới tính	Ngày sinh	Địa chỉ
▶	NV01	Vương Thị Thanh Bình	Nữ	14/02/1987	Nguyễn Khê - Đông Anh - H
*	NV02	Vương Bảo An	Nam	14/02/1990	Đông Anh - Hà Nội

At the bottom are buttons forThêm (Add), Lưu (Save), Sửa (Edit), Xóa (Delete), Bỏ qua (Cancel), and Thoát (Exit).

Chú ý:

- + Điều khiển dùng để nhập ngày sinh của nhân viên là điều khiển MaskedTextBox. Để quy định nhập ngày tháng năm cho điều khiển này ta quy định thuộc tính **Mask** là *Short date*
- + Trước khi lưu vào CSDL cần kiểm tra xem ngày sinh người dùng nhập và có phải là kiểu ngày tháng không?

Bài giảng lập trình trực quan

Bài tập 29:

Tiếp tục bài tập 28, tạo form cập nhật danh mục Khách hàng có giao diện như sau:

Cập nhật danh mục khách hàng

Danh sách các khách hàng

Mã khách:	KH01			Địa chỉ:	Nguyễn Khê - Đông Anh - Hà Nội	
Tên khách:	Vương Tiến Anh			Điện thoại:	0976967619	
*	Mã Khách	Tên khách	Địa chỉ	Diện thoại		
▶	KH01	Vương Tiến Anh	Nguyễn Khê - Đông Anh - Hà Nội	0976967619		
	KH02	Nguyễn Thị Hồng Hoa	183 Nguyễn Chí Thanh - Hà Nội	0986698119		

Thao tác:

- Thêm
- Lưu
- Sửa
- Xóa
- Bỏ qua
- Thoát

Bài giảng lập trình trực quan

Bài tập 30:

Tiếp tục bài tập 29, tạo form cập nhật danh mục Hàng có giao diện như sau:

Danh mục hàng

DANH MỤC HÀNG KHÓA

Mã hàng:	H01	<input type="button" value="Ảnh"/>	
Tên hàng:	Hoa hồng đỏ		
Chất liệu:	Vải		
Số lượng:	8		
Đơn giá nhập:	120000		
Đơn giá bán:	150000		Ghi chú: 15 bông hoa hồng nhung bằng vải mềm được bó tròn rất đẹp

	Mã hàng	Tên hàng	Mã CL	Số lượng	Giá nhập	Giá bán
▶	H01	Hoa hồng đỏ	CLV	8	120000	150000
	H02	Hoa hình trái tim	CLG	10	50000	80000
	H03	Đồng hồ trái tim	CLN	15	100000	150000
	H04	Gấu bông	CLB	10	200000	250000
*						

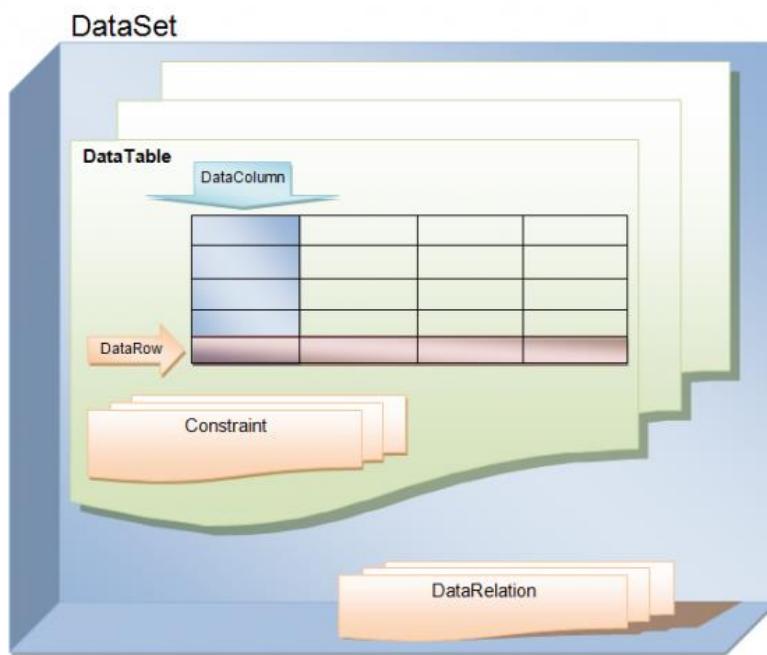
Chú ý:

- + Khi kích vào nút **Ảnh** thì sẽ mở ra hộp thoại OpenFileDialog để người dùng chọn ảnh hiển thị cho sản phẩm.
- + Lưu ảnh vào CSDL dưới dạng đường dẫn của ảnh.
- + Để lấy dữ liệu đổ ra một ComboBox, ta gán một DataTable chứa dữ liệu cho thuộc tính DataSource của ComboBox.

7.5. Làm việc với Dataset và DataTale

7.5.1. Cấu trúc của Dataset và DataTale

Một DataSet tương tự như một tập tin database vật lý hoàn chỉnh nhưng được lưu trong bộ nhớ. DataSet bao gồm các DataTable, DataTable bao gồm các DataColumn, DataRow, các constraint được minh họa như hình dưới:



7.5.2. Các phương thức

Ở phần trên ta đã đề cập đến các phương thức cơ bản của DataSet và DataTable. Ở phần này ta chỉ đề cập thêm một số phương thức làm việc với các DataColumn và DataRow của DataTable để phục vụ cho việc tự tạo một DataTable.

- + Dùng phương thức Add("Tên cột", kiểu dl) để thêm cột mới vào DataTable

```
table.Columns.Add("Name", Type.GetType("Tên_kiểu"))
```

- + Dùng phương thức newRow() để tạo một dòng dữ liệu mới vào DataTable

```
DataRow newR ;  
newR = tabs.NewRow();
```

- + Dùng phương thức Add(newRow) để chèn dòng vừa tạo vào DataTable

```
table.Rows.Add(newR)  
hoặc  
table.Rows.Add("DL_cột 1", "DL_cột 2", ..., "DL_cột n")
```

7.5.3. Tự tạo DataTable

Ở các ví dụ ở trên hầu hết ta nạp dữ liệu cho DataTable là dữ liệu được lấy từ CSDL. Vì DataTable là một loại dữ liệu độc lập với server nên ta có thể tự tạo một DataTable thông qua các DataColumn và DataRow của DataTable.

Kiểu dữ liệu DataColumn chứa đầy đủ các property cần thiết để bạn tạo ra một mô hình dữ liệu hoàn chỉnh cho DataTable.

Ví dụ: ta cần tạo một bảng DataTable có cấu trúc gồm 2 cột là: STT (có kiểu dữ liệu là số nguyên và tự động tăng) và cột thứ 2 là HoTen (có kiểu dữ liệu là String).

```
//Khai báo và khởi tạo một DataTable
DataTable tables = new DataTable();

//Khai báo và khởi tạo một cột với tên là STT kiểu là số nguyên
DataColumn col1 = new DataColumn("STT", Type.GetType("System.Int32"));

col1.AllowDBNull = false; // Cột không được phép Null
col1.AutoIncrement = true ; //Giá trị tăng tự động
col1.AutoIncrementSeed = 1; //Bắt đầu đánh số từ 1
col1.Unique = true ;//Giá trị là duy nhất
tables.Columns.Add(col1) ; //Thêm cột STT vào bảng tables
//Chèn thêm cột HoTen
tables.Columns.Add("HoTen", Type.GetType("System.String"));
```

Sau khi tạo xong cấu trúc của bảng, Ta sẽ chèn dữ liệu vào bảng thông qua kiểu dữ liệu DataRow như sau:

```
//Khai báo một DataRow
DataRow newR;

//Chèn dòng vào bảng
newR = tables.NewRow();

//Chèn dữ liệu vào bảng. Ta không phải thêm vào cột STT vì nó tự động tăng bắt đầu từ 1
newR["HoTen"]= "Nguyễn Thu Hường"; //Thêm dữ liệu cho cột Hoten
tables.Rows.Add(newR); //Chèn hàng vào bảng

//Chèn các hàng vào bảng
tables.Rows.Add(null, "Vương Tiến Anh");
tables.Rows.Add(null, "Vương Lê Thành Đạt");
tables.Rows.Add(null, "Vương Lê Thành Công");
```

Bài giảng lập trình trực quan

Thế là ta đã có một tables. Dùng DataGridView để hiển thị Table này lên ta sẽ có kết quả như sau :

STT	HoTen
1	Nguyễn Thu Hường
2	Vương Tiến Anh
3	Vương Lê Thành Đạt
4	Vương Lê Thành Công
*	

Bảng dữ liệu mà ta tạo hoàn toàn không liên quan đến một dữ liệu nguồn nào cả.

7.6. Xuất dữ liệu ra Excel

7.6.1. Làm việc với đối tượng Excel

Trong quá trình xây dựng ứng dụng, đôi khi ta muốn xuất dữ liệu ra một file excel, ví dụ như in danh sách nhân viên, danh sách khách hàng, hóa đơn, bảng điểm, bảng lương, ...

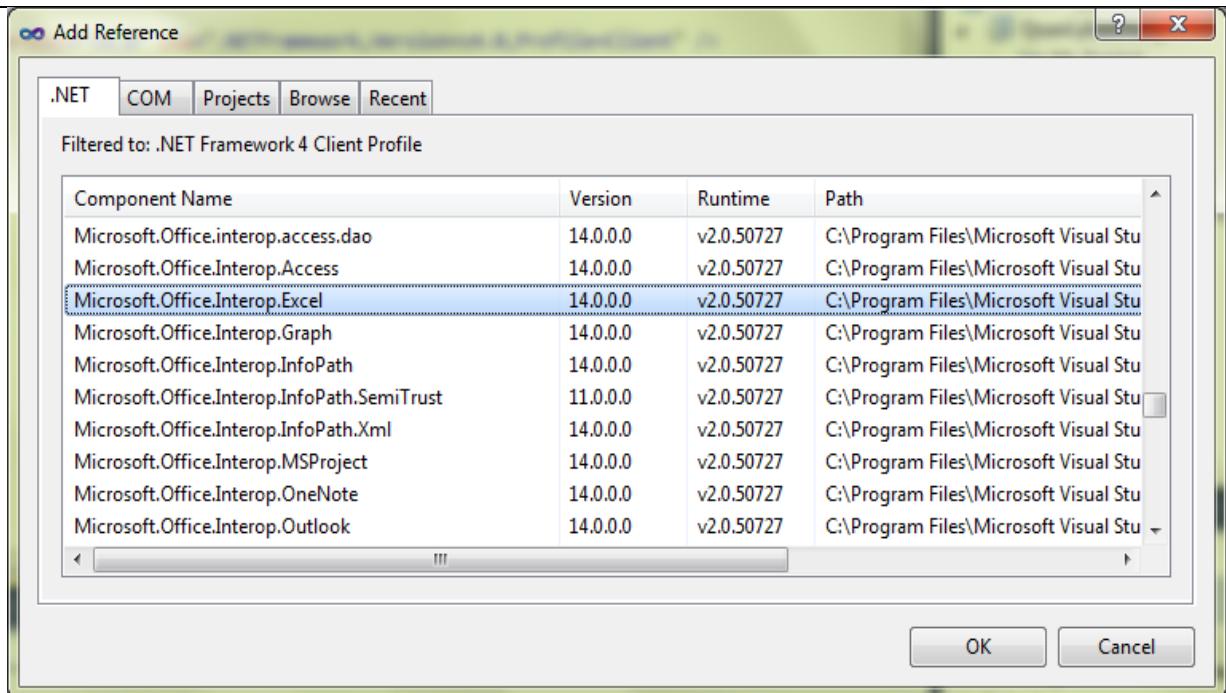
Để làm điều này ta thực hiện các bước như sau:

Bước 1. Tham chiếu đến file thư viện

Để xuất được dữ liệu ra file excel, ta phải tham chiếu (reference) đến đối tượng **Microsoft.Office.Interop.Excel.dll** bằng thao tác:

- + Vào menu **Project/Add Reference...** xuất hiện hộp hội thoại Add Reference
- + Trong tab .NET chọn **Microsoft.Office.Interop.Excel** rồi nhấn OK như hình vẽ:

Bài giảng lập trình trực quan



Bước 2. Khai báo sử dụng đối tượng Microsoft.Office.Interop.Excel

Phần khai báo này đặt ở đầu file .cs và được khai báo theo cú pháp sau:

```
using Excel = Microsoft.Office.Interop.Excel;
```

Với **Excel** là tên của đối tượng, ta có thể đặt một tên khác ở đây.

Bước 3. Khai báo và khởi tạo các thành phần của đối tượng Microsoft.Office.Interop.Excel

Đối tượng **Microsoft.Office.Interop.Excel** bao gồm các thành phần chính sau:

- Application (chương trình excel)
- Workbook (file .xls, có nhiều workbook trong app)
- Worksheet (các trang bảng tính trong file .xls, có nhiều worksheet trong workbook)
- Range (Là một khối ô làm việc của Excel – nó cũng có thể chỉ là một ô)

Ta khai báo và khởi tạo các đối tượng này theo cú pháp sau:

```
//Khai báo và khởi tạo các đối tượng  
Excel.Application exApp = new Excel.Application();  
Excel.Workbook exBook = exApp.Workbooks.Add(Excel.XlWBATemplate.xlWBATWorksheet);  
Excel.Worksheet exSheet = (Excel.Worksheet)exBook.Worksheets[1];  
Excel.Range tenTruong = (Excel.Range)exSheet.Cells[1, 1]; //Đưa con trỏ vào ô A1
```

Với **Excel** là tên của đối tượng đã khai báo ở bước 2.

Bước 4: Định dạng cho file Excel

Ta có một vài các thao tác định dạng cơ bản như sau:

Các định dạng liên quan đến font chữ

Ở đây ta làm việc với các thuộc tính của đối tượng Range. Ví dụ ta định dạng từ ô A1 đến B3 như sau:

```
tenTruong.Font.Size = 10; //Đặt cỡ chữ là 10  
tenTruong.Font.Name = "Times new roman"; //Chọn font Times new roman  
tenTruong.Font.Bold = true; //Định dạng kiểu font chữ là in đậm  
tenTruong.Font.Color = Color.Blue; //Màu xanh da trời  
hoặc  
exSheet.get_Range("A1").Font.Color=Color.Blue;  
hoặc dùng cú pháp  
  
exSheet.get_Range("A7:G7").Font.Bold = true;
```

Các định dạng khác

+ Chính độ rộng của ô

```
tenTruong.ColumnWidth = 7
```

+ Trộn các ô lại với nhau

```
exSheet.get_Range ("C8:E8") . Merge(true);
```

+ Căn lề cho các ô

```
exRange.Range ("C2:E2") .HorizontalAlignment =  
Microsoft.Office.Interop.Excel.XlHAlign.xlHAlignCenter
```

Ở trên là ta căn giữa, ta có thể căn kiểu khác như:

- Microsoft.Office.Interop.Excel.XlHAlign.xlHAlignLeft: Căn trái
- Microsoft.Office.Interop.Excel.XlHAlign.xlHAlignRight: Căn phải

+ Viết nội dung cho các ô

```
exSheet.get_Range ("A7") .Value = "STT";
```

hoặc

```
tenTruong.Value="Nội dung"
```

+ Đặt tên cho sheet đang làm việc

```
exSheet.Name="Name"
```

Bài giảng lập trình trực quan

Bước 5: Kích hoạt cho file excel hoạt động

```
exBookActivate();
```

Bước 6: Lưu file excel lại

Để lưu file ta dùng lệnh:

```
exBookSaveAs(đường dẫn file excel);
```

Ta dùng kèm theo hộp thoại *SaveFileDialog* để cho phép người dùng chọn nơi để lưu file Excel.

Khi ấy đường dẫn được lấy bằng thuộc tính *FileName* của hộp thoại lưu file

```
exBookSaveAs (dlgSaveFile.FileName.ToString());
```

Bước 7: Thoát khỏi ứng dụng

```
exApp.Quit()
```

7.6.2. Ví dụ

Tạo form tìm kiếm hàng với giao diện như sau:

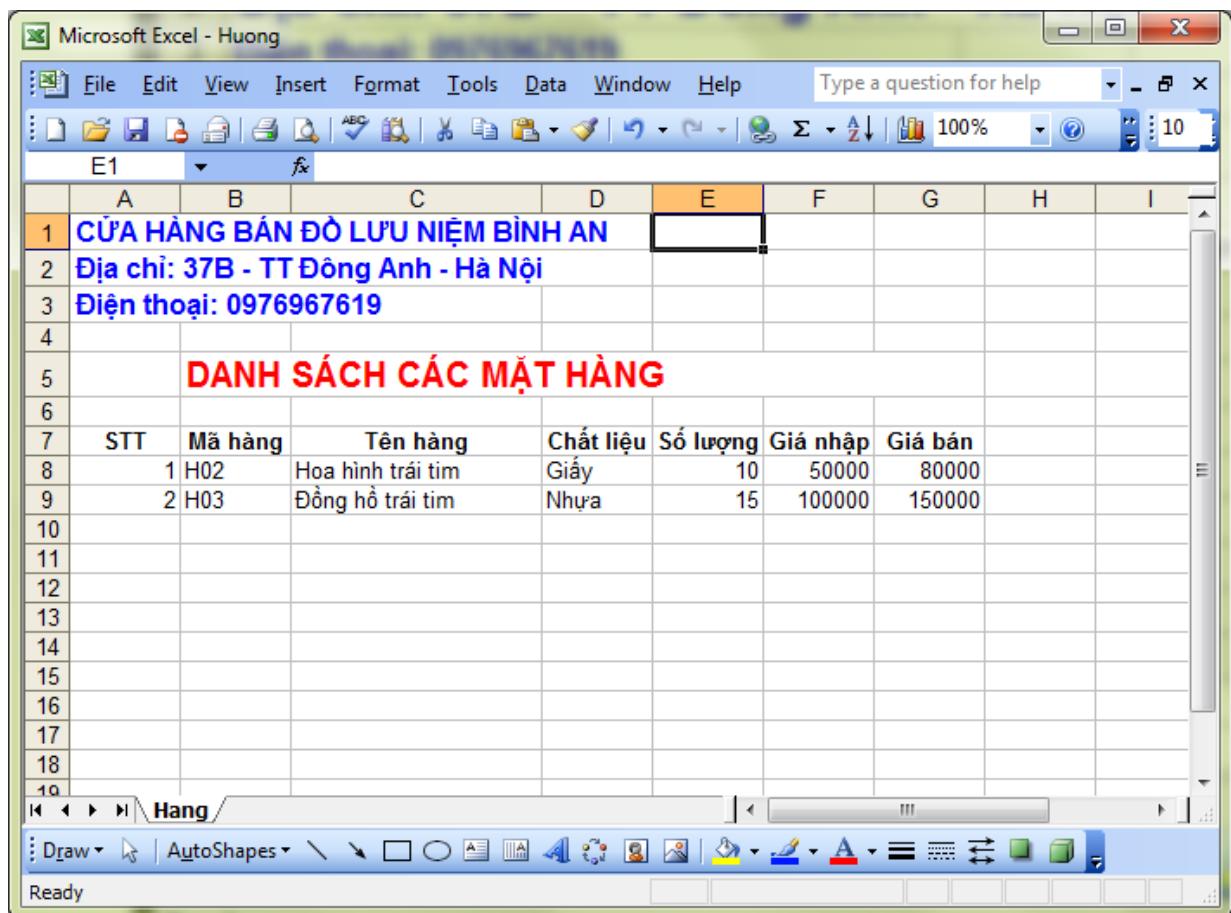


Người dùng điền thông tin các tiêu chí cần tìm, việc tìm kiếm là tìm kiếm gần đúng (với từ khóa like), điều kiện tìm kiếm là điều kiện and.

Khi người dùng nhấn nút *Tìm kiếm* thì kết quả tìm kiếm hiện lên trên lưới.

Bài giảng lập trình trực quan

Khi người dùng chọn nút *In ra excel* thì kết quả tìm kiếm được in ra file Excel như sau:



Hướng dẫn

Tạo form frmTKHang với các điều khiển cơ bản sau:

Loại	Tên điều khiển	Text
TextBox	txtMaHang	
	txtTenHang	
	txtDonGiaBan1	
	txtDonGiaBan2	
ComboBox	cboChatLieu	
Button	btnTim	Tìm kiếm

Bài giảng lập trình trực quan

	btnIn	In ra Excel
DataGridView	dgvHang	
SaveFileDialog	dlgSave	

Mở sự kiện Click của ComboBox cboChatLieu viết vào đoạn mã lệnh lấy ra danh sách chất liệu như sau:

```
private void cboChatLieu_Click(object sender, EventArgs e)
{
    cboChatLieu.DataSource = dtBase.DocBang("Select * from tblChatLieu");
    cboChatLieu.ValueMember = "MaChatLieu"; //Giá trị của phần tử
    cboChatLieu.DisplayMember = "TenChatLieu"; //Chuỗi hiển thị của phần tử
}
```

Mở sự kiện Click nút Tìm kiếm viết vào đoạn mã thực hiện chức năng tìm kiếm như sau:

```
private void btnTim_Click(object sender, EventArgs e)
{
    //Trường hợp không chọn tiêu chí tìm sẽ lấy ra tất cả các mặt hàng
    sql = "select MaHang,TenHang,TenChatLieu,SoLuong,DonGiaNhap,DonGiaBan," +
        "tblHang.MaChatLieu from tblHang inner join tblChatLieu on " +
        "tblHang.MaChatLieu=tblChatLieu.MaChatLieu where MaHang is not null";

    //Khi chọn tiêu chí nào sẽ ghép với tiêu chí đó bằng từ and
    //Tìm kiếm gần đúng với từ khóa like
    if (txtMaHang.Text != "")
        sql = sql + " and MaHang like N'" + txtMaHang.Text.Trim() + "%'";
    if (txtTenHang.Text != "")
        sql = sql + " and TenHang like N'" + txtTenHang.Text.Trim() + "%'";
    if (txtDonGiaBan1.Text != "")
        sql = sql + " and DonGiaBan >= " + txtDonGiaBan1.Text;
    if (txtDonGiaBan2.Text != "")
        sql = sql + " and DonGiaBan <= " + txtDonGiaBan2.Text;
    if (cboChatLieu.Text != "")
        sql = sql + " and tblHang.MaChatLieu=" +
            cboChatLieu.SelectedValue.ToString() + "'";

    //Trình bày gridView
    dtHang = dtBase.DocBang(sql);
    dgvHang.DataSource = null;
    dgvHang.DataSource = dtHang;

    dgvHang.Columns[0].HeaderText = "Mã hàng";
    dgvHang.Columns[1].HeaderText = "Tên hàng";
    dgvHang.Columns[2].HeaderText = "Chất liệu";
    dgvHang.Columns[3].HeaderText = "Giá nhập";
    dgvHang.Columns[4].HeaderText = "Giá bán";
    dgvHang.Columns[5].HeaderText = "Số lượng";
    dgvHang.Columns[0].DataPropertyName = "MaHang";
    dgvHang.Columns[1].DataPropertyName = "TenHang";
    dgvHang.Columns[2].DataPropertyName = "TenChatLieu";
    dgvHang.Columns[3].DataPropertyName = "DonGiaNhap";
    dgvHang.Columns[4].DataPropertyName = "DonGiaBan";
    dgvHang.Columns[5].DataPropertyName = "SoLuong";
```

Bài giảng lập trình trực quan

}

Mở sự kiện Click nút *In ra Excel* viết vào đoạn mã thực hiện chức năng in ra Excel như sau:

```
private void btnIn_Click(object sender, EventArgs e)
{
    if (dtHang.Rows.Count > 0) //TH có dữ liệu để ghi
    {
        //Khai báo và khởi tạo các đối tượng
        Excel.Application exApp = new Excel.Application();
        Excel.Workbook exBook =
            exApp.Workbooks.Add(Excel.XlWBATemplate.xlWBATWorksheet);
        Excel.Worksheet exSheet = (Excel.Worksheet)exBook.Worksheets[1];

        //Định dạng chung
        Excel.Range tenCuaHang = (Excel.Range)exSheet.Cells[1, 1];
        tenCuaHang.Font.Size = 12;
        tenCuaHang.Font.Bold = true;
        tenCuaHang.Font.Color = Color.Blue;
        tenCuaHang.Value = "CỬA HÀNG BÁN ĐỒ LƯU NIỆM BÌNH AN";

        Excel.Range dcCuaHang = (Excel.Range)exSheet.Cells[2, 1];
        dcCuaHang.Font.Size = 12;
        dcCuaHang.Font.Bold = true;
        dcCuaHang.Font.Color = Color.Blue;
        dcCuaHang.Value = "Địa chỉ: 37B - TT Đông Anh - Hà Nội";

        Excel.Range dtCuaHang = (Excel.Range)exSheet.Cells[3, 1];
        dtCuaHang.Font.Size = 12;
        dtCuaHang.Font.Bold = true;
        dtCuaHang.Font.Color = Color.Blue;
        dtCuaHang.Value = "Điện thoại: 0976967619";

        Excel.Range header = (Excel.Range)exSheet.Cells[5, 2];
        exSheet.get_Range("B5:G5").Merge(true);
        header.Font.Size = 13;
        header.Font.Bold = true;
        header.Font.Color = Color.Red;
        header.Value = "DANH SÁCH CÁC MẶT HÀNG";

        //Định dạng tiêu đề bảng

        exSheet.get_Range("A7:G7").Font.Bold = true;
        exSheet.get_Range("A7:G7").HorizontalAlignment =
            Microsoft.Office.Interop.Excel.XlHAlign.xlHAlignCenter;
        exSheet.get_Range("A7").Value = "STT";
        exSheet.get_Range("B7").Value = "Mã hàng";
        exSheet.get_Range("C7").Value = "Tên hàng";
        exSheet.get_Range("C7").ColumnWidth = 20;
        exSheet.get_Range("D7").Value = "Chất liệu";
        exSheet.get_Range("E7").Value = "Số lượng";
        exSheet.get_Range("F7").Value = "Giá nhập";
        exSheet.get_Range("G7").Value = "Giá bán";

        //In dữ liệu
        for (int i = 0; i < dtHang.Rows.Count; i++)
        {
            exSheet.get_Range("A"+(i + 8).ToString()+":G"+(i + 8).ToString())
                .Font.Bold = false;
            exSheet.get_Range("A" + (i + 8).ToString()).Value = (i + 1)
                .ToString();
        }
    }
}
```

Bài giảng lập trình trực quan

```
        exSheet.get_Range("B" + (i + 8).ToString()).Value =
            dtHang.Rows[i]["MaHang"].ToString();
        exSheet.get_Range("C" + (i + 8).ToString()).Value =
            dtHang.Rows[i]["TenHang"].ToString();
        exSheet.get_Range("D" + (i + 8).ToString()).Value =
            dtHang.Rows[i]["TenChatLieu"].ToString();
        exSheet.get_Range("E" + (i + 8).ToString()).Value =
            dtHang.Rows[i]["SoLuong"].ToString();
        exSheet.get_Range("F" + (i + 8).ToString()).Value =
            dtHang.Rows[i]["DonGiaNhap"].ToString();
        exSheet.get_Range("G" + (i + 8).ToString()).Value =
            dtHang.Rows[i]["DonGiaBan"].ToString();
    }

    exSheet.Name = "Hang";
    exBook.Activate(); //Kích hoạt file Excel
    //Thiết lập các thuộc tính của SaveFileDialog
    dlgSave.Filter = "Excel Document (*.xls) | *.xls | Word Document (*.doc)
        | *.doc | All files (*.*) | *.*";
    dlgSave.FilterIndex = 1;
    dlgSave.AddExtension = true;
    dlgSave.DefaultExt = ".xls";
    if (dlgSave.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        exBook.SaveAs(dlgSave.FileName.ToString()); //Lưu file Excel
    exApp.Quit(); //Thoát khỏi ứng dụng
}
else
    MessageBox.Show("Không có danh sách hàng để in");
}
```

Bài tập 31:

Xây dựng form *Hóa đơn bán* với các chức năng như giao diện sau:

Bài giảng lập trình trực quan

Mã hàng	Tên hàng	Số lượng	Đơn giá	Giảm giá %	Thành tiền
H01	Hoa hồng vàng	5	55000	10	247500
H02	Hoa bồ	4	75000	0	300000
H03	Gấu Pooh	5	90000	0	450000

Yêu cầu:

- + Khi chọn Mã nhân viên, mã khách, mã hàng thì các thông tin liên quan của nhân viên, khách hàng và hàng tương ứng hiện ra.
- + Khi nhấn nút Thêm mới thì mã hóa đơn tự động sinh ra.
- + Khi viết số lượng, giảm giá thành tiền được tự động cập nhật.
- + Khi lưu một mặt hàng vào hóa đơn (mua hàng), số lượng được cập nhật tại danh mục hàng. Trước khi mua cần kiểm tra số lượng có đủ không.

7.7. In báo cáo với CrystalReport

Để có thể in dữ liệu, ta có thể dùng in ra excel như trên. Tuy nhiên ta có thể dùng CrystalReport. CrystalReport là một công cụ tạo báo cáo đơn giản nhưng cũng khá hiệu quả.

- ▲ Crystal Report là công cụ để tạo các báo cáo cho các ứng dụng Windows application hoặc Web application. Có thể tích hợp file .rpt vào ứng dụng Windows application hoặc Web application để hiển thị report

Bài giảng lập trình trực quan

- ▲ Crystal Report có thể hiển thị thông tin dưới dạng bảng, đồ họa, biểu đồ, ... có khả năng tính toán như tính tổng, trung bình, ...
- ▲ Crystal Reports .NET đã tích hợp sẵn Report Designer trong bộ Visual Studio .NET để có thể thiết kế report (file .rpt).
- ▲ Việc chạy các application có tích hợp Crystal Report đòi hỏi phải có Crystal Report Engine được cài đặt trên máy.

7.7.1. Cài đặt Crystal report cho visual studio 2010

Download bộ cài tại địa chỉ: http://www.businessobjects.com/jump/...s2_default.asp

Choose and download an installation package:

[SAP Crystal Reports, version for Visual Studio 2010 - Standard](#)

Standard EXE installation package which installs the software into the Visual Studio IDE.

 Chọn Standard

[SAP Crystal Reports, version for Visual Studio 2010 - Click Once](#)

Click once installation package used to create self-updating Windows-based applications which can be installed and run with minimal user interaction.

[SAP Crystal Reports, version for Visual Studio 2010 - Click Once \(32 Bit\)](#)

[SAP Crystal Reports, version for Visual Studio 2010 - Click Once \(64 Bit\)](#)

[SAP Crystal Reports, version for Visual Studio 2010 - Merge Modules](#)

Merge modules installation package used to install components which are shared by multiple applications.

[SAP Crystal Reports runtime engine for .NET Framework 4 \(32-bit\)](#)

[SAP Crystal Reports runtime engine for .NET Framework 4 \(64-bit\)](#)

Sau khi download về, tiến hành cài đặt.

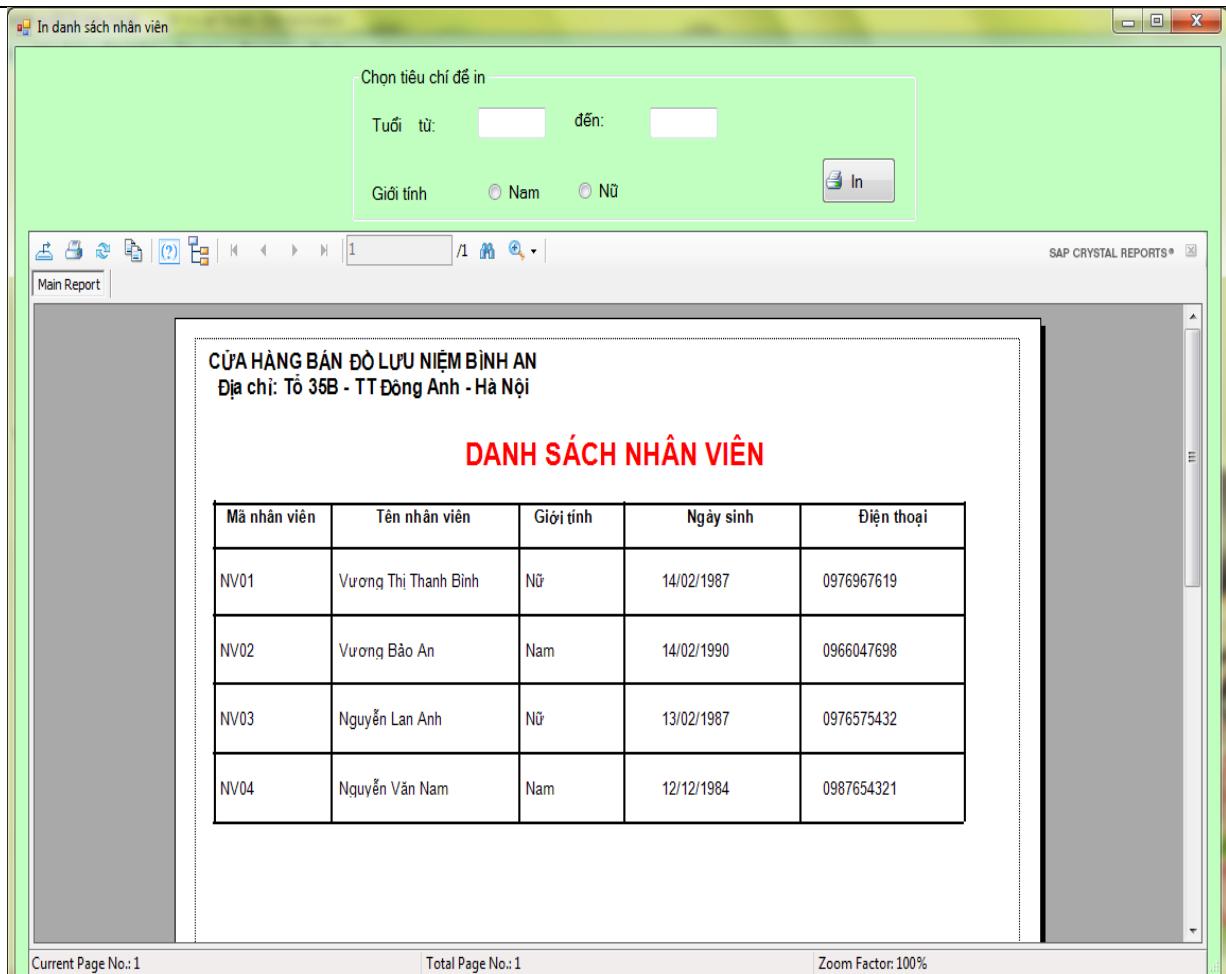
7.7.2. Làm việc với CrystalReport thông qua DataSet

Có nhiều cách để tạo report, tuy nhiên trong bài giảng này tôi hướng dẫn các bạn cách tạo report từ dataset.

Trước khi tạo Crystal Report ta cần xác định xem hình dạng và dữ liệu cần in ra báo cáo gồm những cái gì. Ví dụ ta cần tạo một báo cáo để in danh sách nhân viên như hình vẽ sau:

Đầu tiên ta tạo giao diện cho form gồm các điều khiển sau:

Bài giảng lập trình trực quan

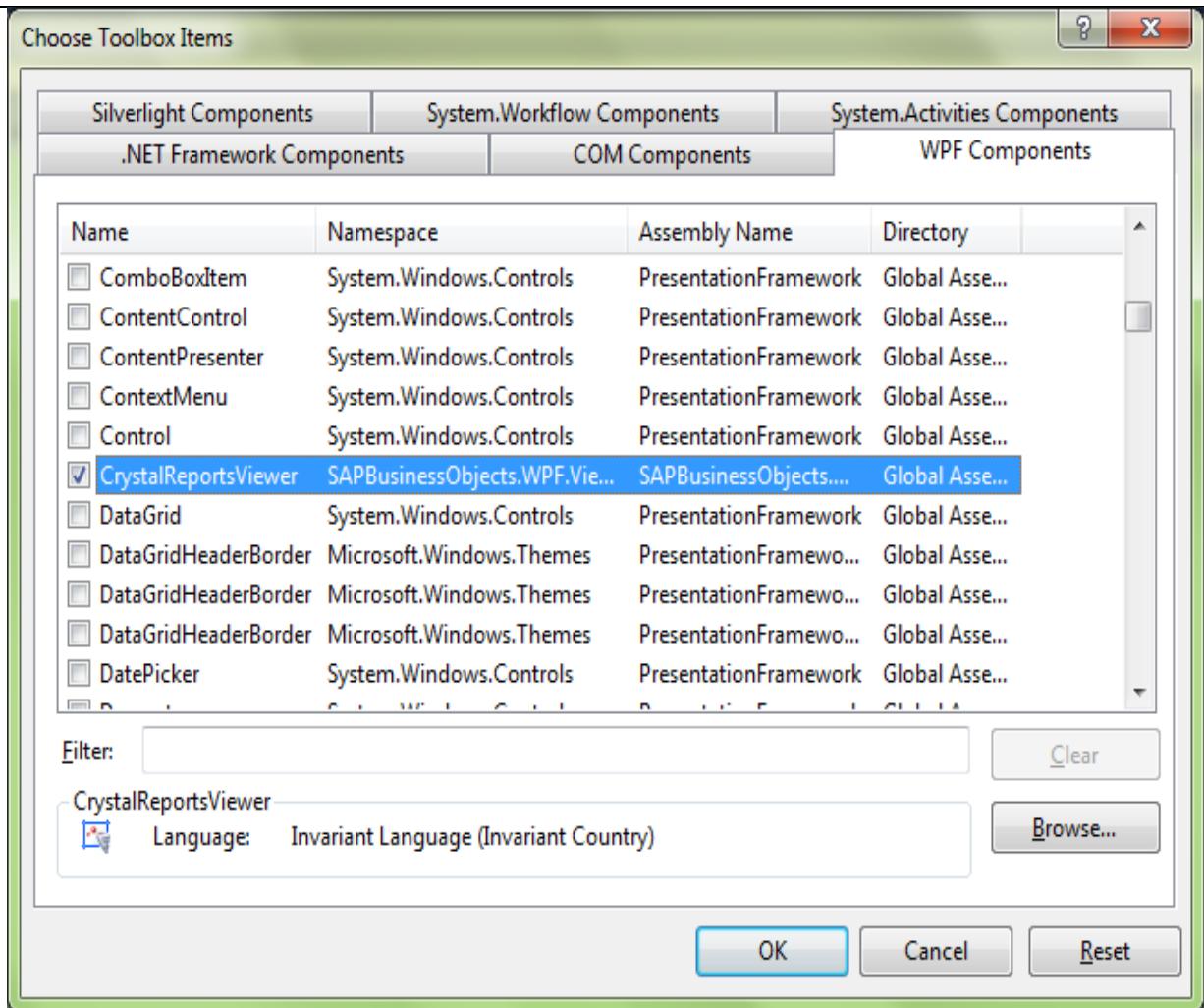


Ta tiếp tục tạo thêm form mới để in danh sách nhân viên trong project QuanLyBanHang và có giao diện như trên. Form gồm có các điều khiển chính như sau:

- + 2 TextBox: để nhập số tuổi có tên là txtTuoi1, txtTuoi2.
- + 2 RadioButton: để chọn giới tính là rdoNam và rdoNu.
- + 1 Button: btnIn
- + 1 CrystalReportViewer: dùng để hiển thị văn bản khi in có tên là rptvNhanVien.

Muốn thêm điều khiển *CrystalReportViewer* ta kích chuột phải vào hộp ToolBox chọn *Choose Item..* sau đó chọn đến thành phần CrystalreportViewer như sau:

Bài giảng lập trình trực quan



Chú ý: Khi thêm điều khiển làm việc với Crystal Report vào chạy chương trình sẽ báo lỗi về Namespace CrystalReportViewer... Ta sẽ sửa lại như sau: Kích chuột phải vào tên dự án chọn properties đổi Target Framework từ .NET Framework 4 Client Profile thành .NET Framework 4.

Để làm việc với Crystal Report ta sẽ khai báo sử dụng namespace sau:

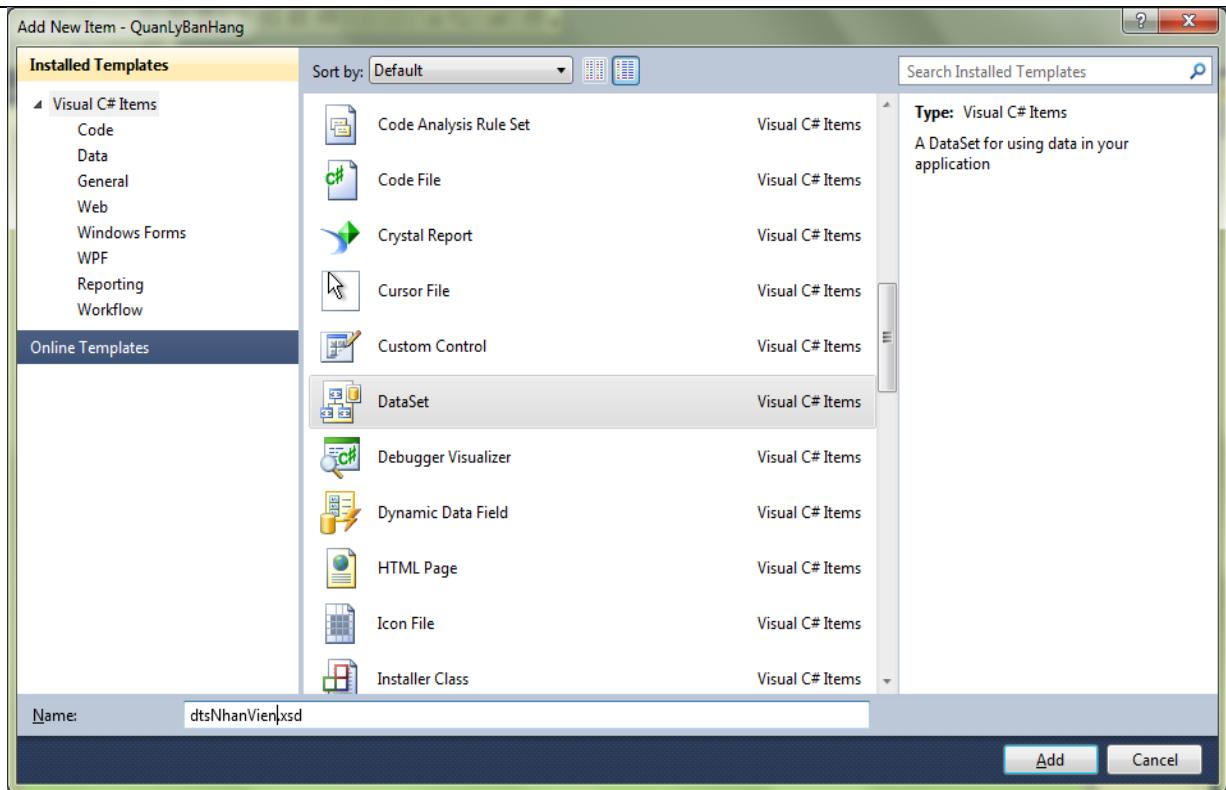
```
using CrystalDecisions.CrystalReports.Engine;
```

Để tạo một báo cáo từ DataSet ta thực hiện theo các bước sau:

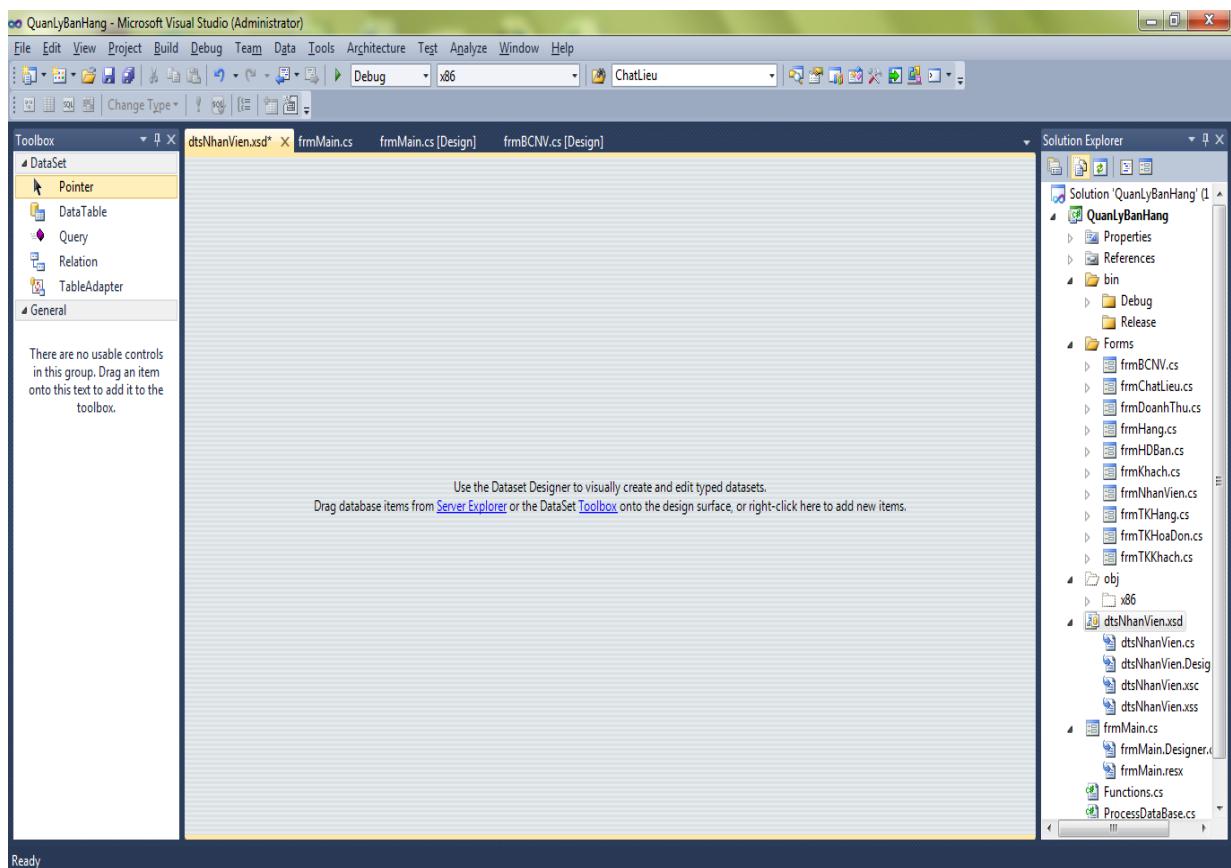
Bước 1: Tạo DataSet

Kích chuột phải vào tên dự án chọn Add/New Item... sau đó chọn Dataset như hình sau và nhấn Add để thêm vào một dataset.

Bài giảng lập trình trực quan

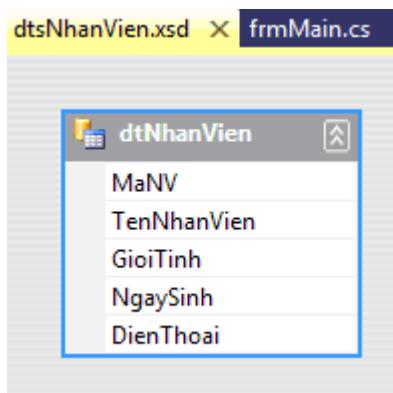


Sau khi chọn Add, xuất hiện giao diện như sau:



Bài giảng lập trình trực quan

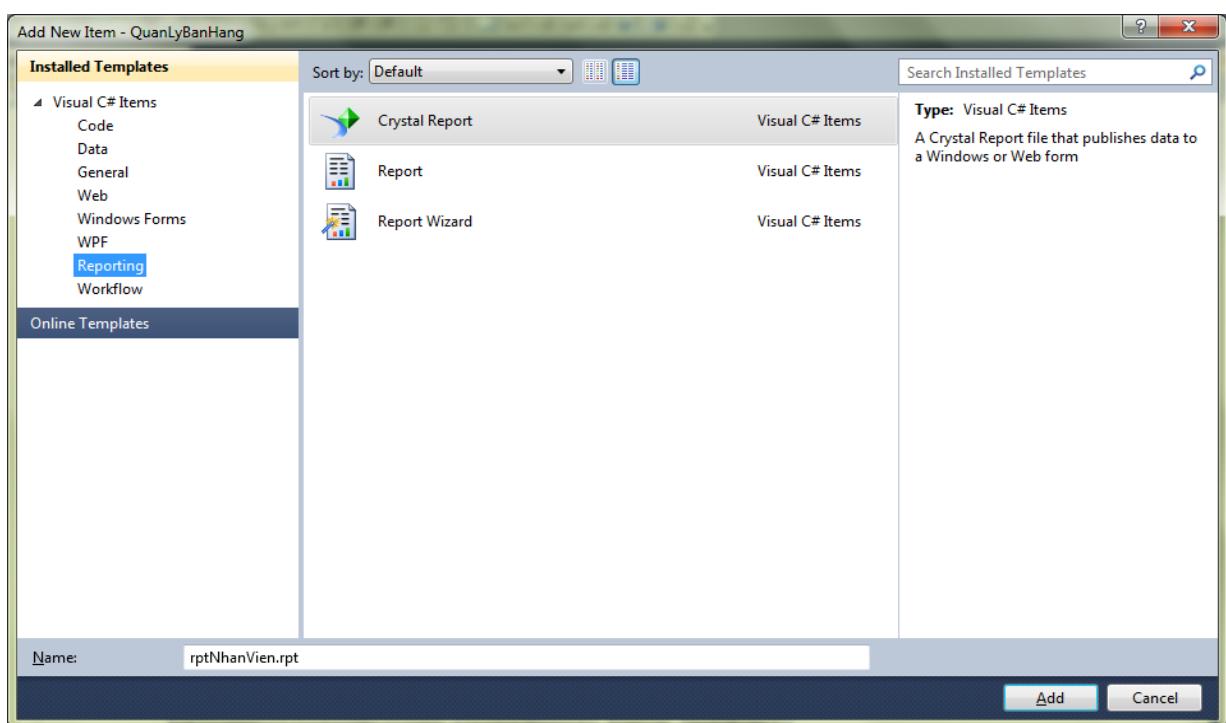
Bước tiếp theo là tạo các datatable cho dataset, bằng cách kéo điều khiển DataTable trên thanh ToolBox vào. Ví dụ tạo một DataTable có cấu trúc như sau:



- + Để đổi tên của datatable, kích chuột phải vào tiêu đề của dataTable chọn Rename.
- + Để thêm một trường (cột) dữ liệu mới, kích chuột phải chọn Add/Column.

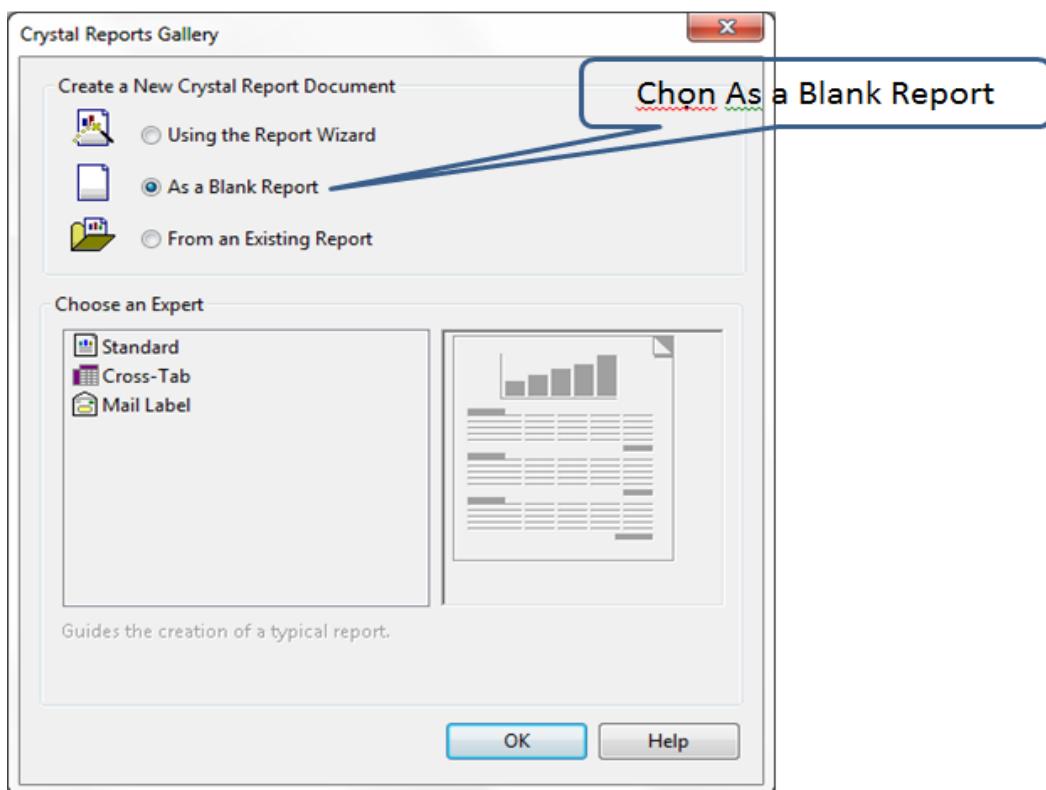
Bước 2: Tạo report.

- + Tạo thư mục chứa các report của ứng dụng, đường dẫn thư mục là: /bin/debug/Report/
 - + Kích chuột phải vào thư mục chứa report, chọn **Add/ New Item...** Sau đó trong nhánh **Reporting** chọn **Crystal report**, đặt tên cho report là **rptNhanVien.rpt** rồi chọn **Add** như hình sau:

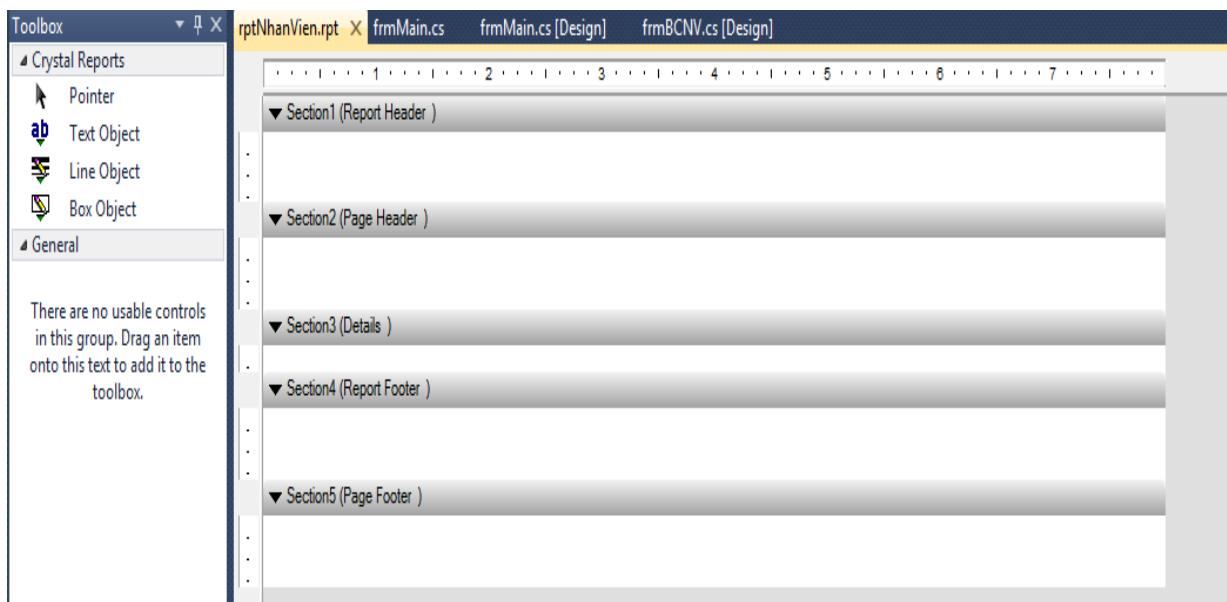


Bài giảng lập trình trực quan

Xuất hiện cửa sổ sau, ta chọn như hình vẽ



Nhấn OK sẽ xuất hiện trên file *rptNhanVien.rpt* cùng với giao diện để thiết kế Crystal Report như sau:



Cấu trúc của một Crystal report:

Ở hình vẽ trên ta thấy một crystal report sẽ có 5 section (từ section 1 đến section 5) với các ý nghĩa của từng section như sau:

- ▲ Section 1 (Report Header): Được hiển thị ở phần đầu báo cáo (đầu trang 1, không hiển thị ở các trang 2, 3, 4, ...)
- ▲ Section 2 (Page Header): Được hiển thị ở đầu tất cả các trang báo cáo.
- ▲ Section 3 (Detail): Hiển thị phần thông tin dữ liệu, từng hàng dữ liệu sẽ được lặp lại.
- ▲ Section 4 (Report Footer): Được hiển thị ở phần cuối báo cáo (ở cuối trang cuối cùng, không hiển thị ở các trang khác)
- ▲ Section 5 (Page Footer): Được hiển thị ở cuối tất cả các trang báo cáo.

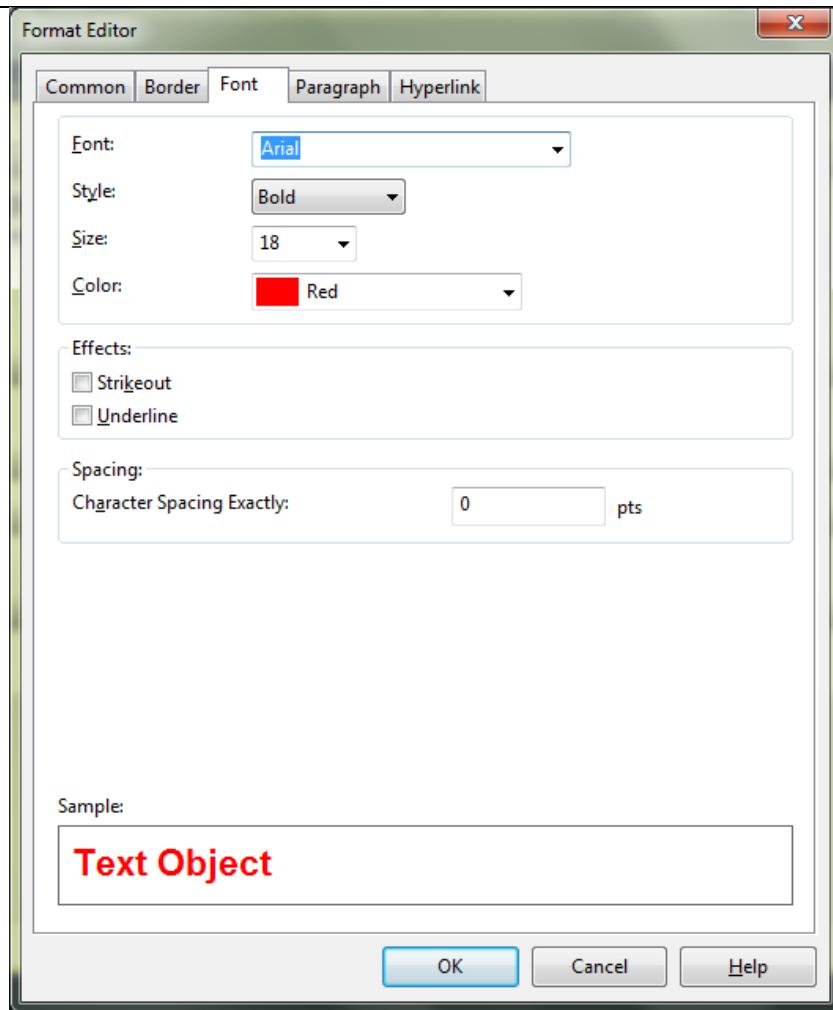
Bước 3: Bắt đầu thiết kế Crystal Report.

+ Sử dụng TextObject để tạo dòng chữ CỬA HÀNG BÁN ĐỒ LUU NIỆM BÌNH AN, dòng chữ Địa chỉ: Tô 35B – TT Đông Anh – Hà Nội và dòng chữ DANH SÁCH NHÂN VIÊN trên phần Section 1 (Report Header) như hình sau:

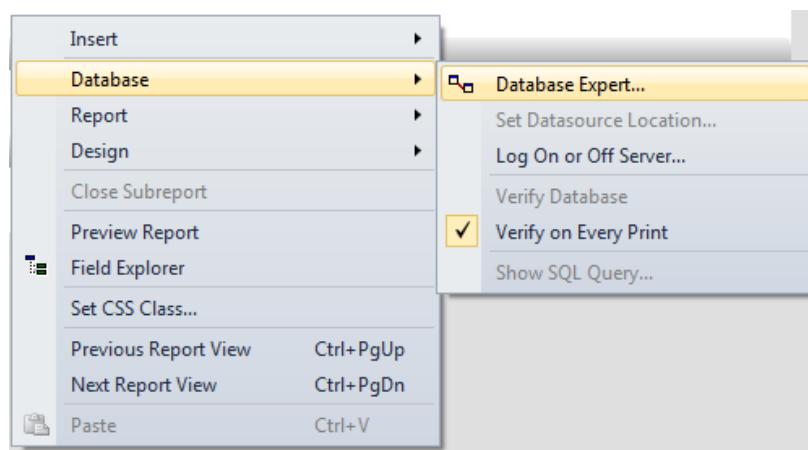


Text Object là đối tượng dùng để hiển thị một chuỗi chữ giống như Label. Để định dạng chữ cho Text Object ta kích chuột phải vào đối tượng Text Object cần định dạng chọn Format Object xuất hiện cửa sổ sau để ta định dạng:

Bài giảng lập trình trực quan

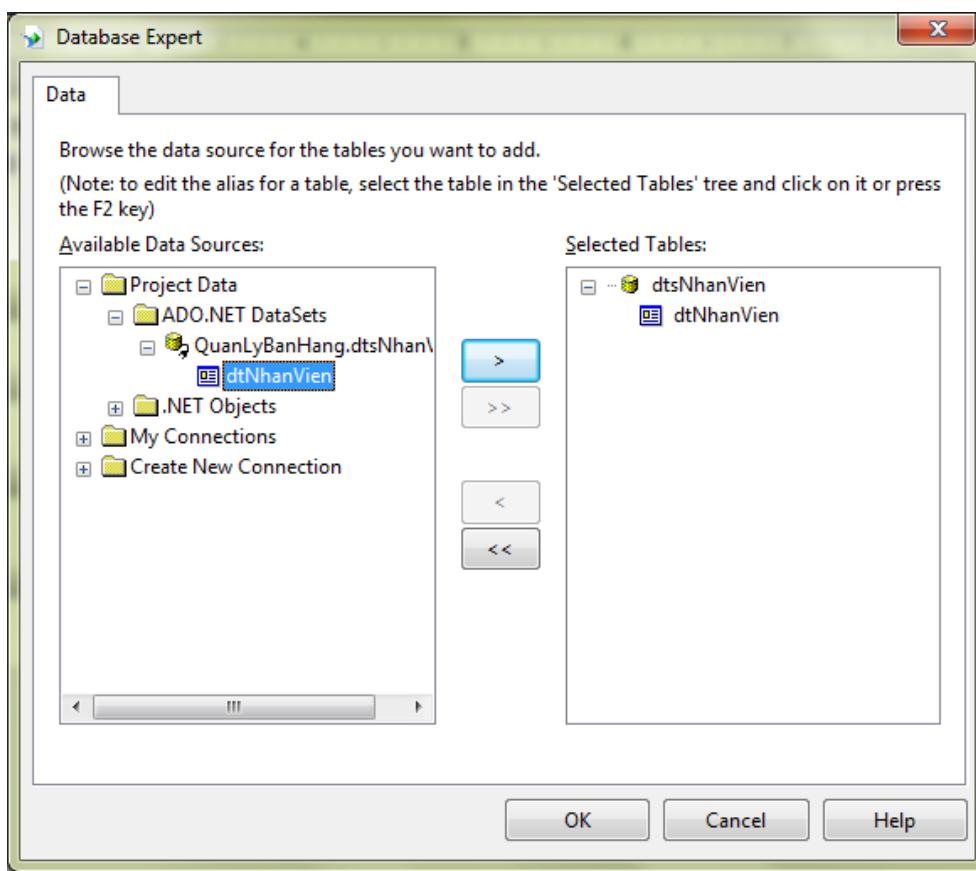


+ Bước tiếp theo, ta bắt đầu chọn dữ liệu cho Crystal Report từ DataSet mà ta đã tạo ở bước 1. Kích chuột phải vào màn hình thiết kế Report chọn Database/Database Expert...

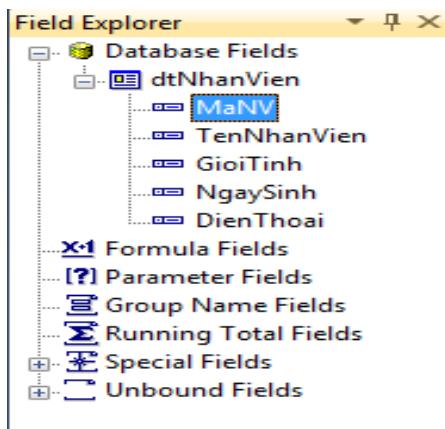


Bài giảng lập trình trực quan

Xuất hiện cửa sổ chọn nguồn dữ liệu, ta chọn dataset vừa tạo ở bước 1, như hình sau:

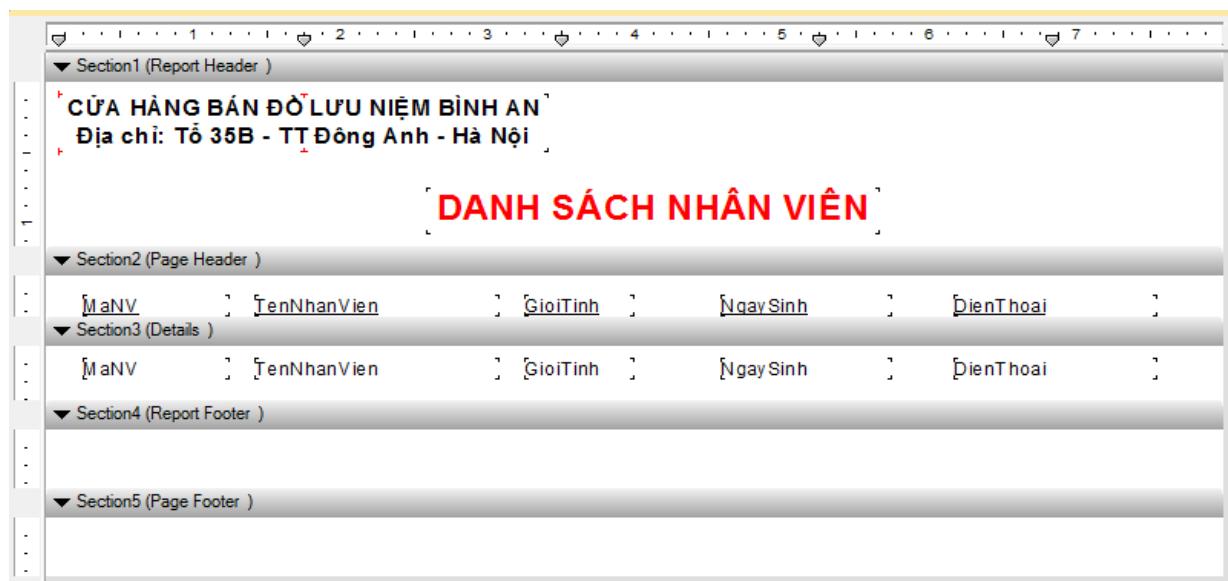


Nhấn OK, trên cửa sổ Field Explorer xuất hiện các trường trong DataTable dtNhanVien đã chọn ở trên. Để mở ra cửa sổ Field Explorer ta kích chuột phải vào giao diện thiết kế report chọn Field Explorer.



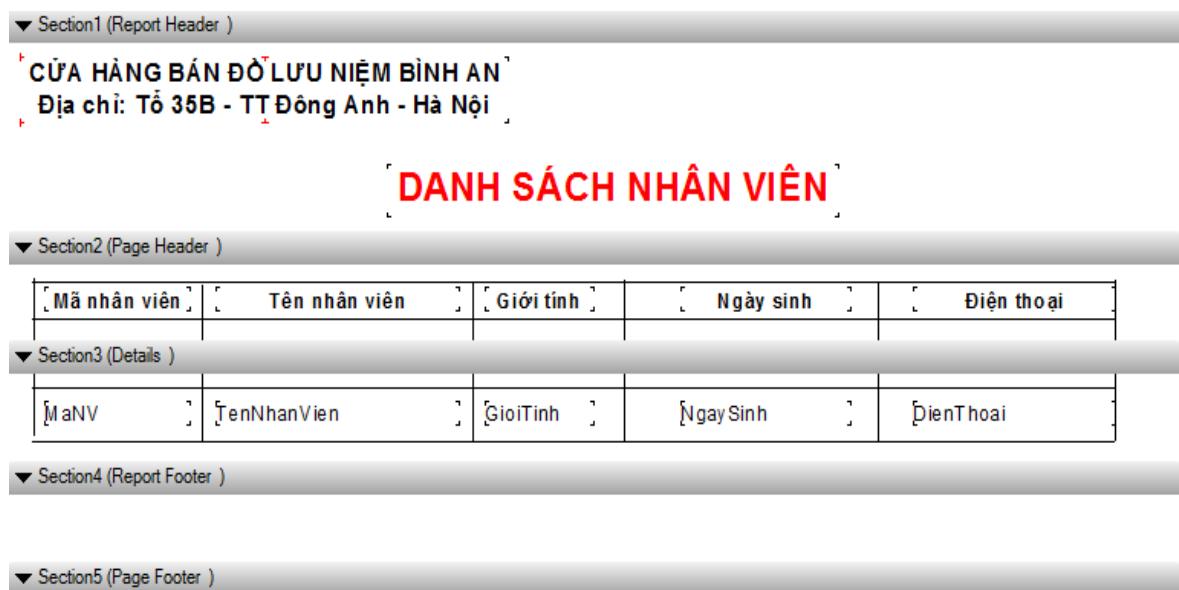
Bài giảng lập trình trực quan

- + Lần lượt kéo các trường dữ liệu cần hiển thị vào phần *Detail* của report.



Khi đó trên phần Page Header cũng xuất hiện ra các tiêu đề cột tương ứng, để sửa lại các tiêu đề này ta kích chuột phải vào từng tiêu đề chọn Edit Text Object.

Để kẻ các đường kẻ ngăn cách giữa các cột ta dùng đối tượng Line Object.



Bước 4: Lấy dữ liệu đổ vào DataTable dữ liệu nguồn của report, gắn Crystal report đã thiết kế ở bước 2 vào CrystalReportViewer.

Bài giảng lập trình trực quan

Với ví dụ trên, ta sẽ viết lệnh cho sự kiện click nút btnIn như sau:

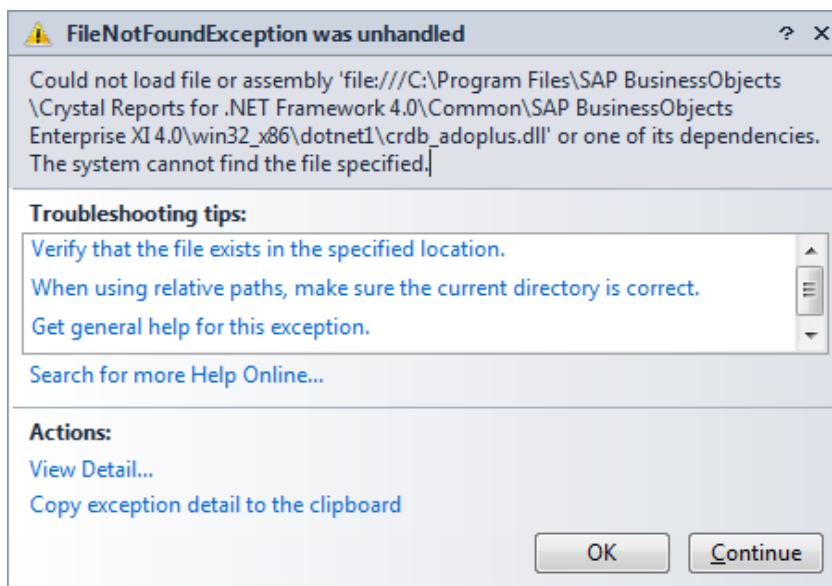
```
private void btnIn_Click(object sender, EventArgs e)
{
    int NamSinh = DateTime.Today.Year; //Lấy ra năm hiện tại
    //Xác định câu lệnh truy vấn
    string sql = "select * from tblNhanVien where MaNhanVien is not null ";
    if (txtTuoil.Text != "")
    {
        int tuoil = Convert.ToInt16(txtTuoil.Text);
        sql = sql + " and Year(NgaySinh)<=" + (NamSinh - tuoil).ToString();
    }
    if (txtTuoil2.Text != "")
    {
        int tuoil2 = Convert.ToInt16(txtTuoil2.Text);
        sql = sql + " and Year(NgaySinh)>=" + (NamSinh - tuoil2).ToString();
    }
    if (rdoNam.Checked == true)
        sql = sql + " and GioiTinh='Nam'";
    if (rdoNu.Checked == true)
        sql = sql + " and GioiTinh='Nữ'";

    //Lấy dữ liệu từ câu lệnh SQL
    DataTable dtNV = dtBase.DocBang(sql);

    //Khởi tạo đối tượng DataSet dtsNhanVien
    dtsNhanVien datasetNV = new dtsNhanVien();
    //Đỗ dữ liệu vào DataTable dtNhanVien đã tạo
    //Dữ liệu được lấy từ DataTable chứa dữ liệu của lệnh sql trên
    DataRow dtrNew; //Khai báo đối tượng DataRow
    for (int i = 0; i < dtNV.Rows.Count; i++)
    {//Lần lượt thêm các dòng dữ liệu vào dtNhanVien của Dataset datasetNV
        dtrNew = datasetNV.Tables["dtNhanVien"].NewRow();
        dtrNew["MaNV"] = dtNV.Rows[i]["MaNhanVien"].ToString();
        dtrNew["TenNhanVien"] = dtNV.Rows[i]["TenNhanVien"].ToString();
        dtrNew["GioiTinh"] = dtNV.Rows[i]["GioiTinh"].ToString();
        dtrNew["NgaySinh"] = string.Format("{0:dd/MM/yyyy}",
            Convert.ToDateTime(dtNV.Rows[i]["NgaySinh"].ToString()));
        dtrNew["DienThoai"] = dtNV.Rows[i]["DienThoai"].ToString();
        datasetNV.Tables["dtNhanVien"].Rows.Add(dtrNew);
    }
    //Gọi đối tượng Report đã thiết kế là file rptNhanVien.rpt
    string fileBaoCao = System.IO.Directory.GetCurrentDirectory() +
        "//Report//rptNhanVien.rpt";
    //Khai báo và khởi tạo đối tượng ReportDocument
    ReportDocument myReportDSNV = new ReportDocument();
    //Load file rptNhanVien.rpt thông qua đối tượng ReportDocument
    myReportDSNV.Load(fileBaoCao);
    //Đặt dữ liệu cho báo cáo chính là dataset chứa dữ liệu nhân viên
    // datasetNV ở trên
    myReportDSNV.SetDataSource(datasetNV);
    //Đặt nguồn cho điều khiển CrystalReportViewer là đối tượng ReportDocument
    //ReportDocument đã chứa report đã tạo
    rptvNhanVien.ReportSource = myReportDSNV;
}
```

Bài giảng lập trình trực quan

Chú ý: Khi chạy chương trình ta sẽ thấy lỗi sau:



Ta xử lý như sau:

+ Mở file *app.config* thay nội dung của nó từ:

```
<?xml version="1.0"?>
<configuration>
<startup><supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/></startup></configuration>
```

Thành:

```
<?xml version="1.0"?>
<configuration>
  <startup useLegacyV2RuntimeActivationPolicy="true">
    <supportedRuntime version="v4.0"/>
    <requiredRuntime version="v4.0.20506"/>
  </startup>
</configuration>
```