

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HCM

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN CÔNG NGHỆ PHẦN MỀM



BÀI GIẢNG

LẬP TRÌNH WINDOWS



TP.HCM, tháng 5 năm 2013

LỜI NÓI ĐẦU

Lập trình Windows là môn học rất quan trọng cho sinh viên đại hoặc, cao đẳng ngành Công nghệ thông tin. Đối với chuyên ngành Công nghệ phần mềm và Hệ thống thông tin thì đây là môn học bắt buộc. Môn học này đóng vai trò quan trọng, làm nền tảng cho sinh viên làm đồ án, thực tập tốt nghiệp cuối khóa.

Mặc khác, giáo trình này không chỉ phù hợp cho sinh viên chuyên ngành Công nghệ phần mềm cũng như Hệ thống thông tin mà còn là tài liệu bổ ích cho sinh viên thuộc các chuyên ngành khác của ngành Công nghệ thông tin cũng như các đối tượng yêu thích lập trình ứng dụng.

Khi biên soạn, chúng tôi đã tham khảo các sách, giáo trình và tài liệu giảng dạy môn học này của một số trường đại học trong và ngoài nước để giáo trình vừa đạt yêu cầu cao về nội dung vừa thích hợp với đối tượng là sinh viên của trường Đại học Công nghiệp Thực phẩm TP.HCM. Chúng tôi cũng đã nhận được sự đóng góp rất quý báu các đồng nghiệp trong và ngoài trường cũng như các lập trình viên đang làm việc tại các công ty phần mềm trong nước.

Mặc dù đã rất cố gắng song không thể tránh khỏi những thiếu sót nhất định, nhóm tác giả mong nhận được những góp ý cả về nội dung lẫn hình thức của các bạn sinh viên, đồng nghiệp, bạn đọc, ... để giáo trình ngày càng hoàn thiện hơn.

Nhóm biên soạn
Trần Thanh Phước, Bùi Công Danh

MỤC LỤC

LỜI NÓI ĐẦU	1
MỤC LỤC.....	3
DANH MỤC HÌNH.....	7
DANH MỤC BẢNG	10
CHƯƠNG 1. TỔNG QUAN NGÔN NGỮ LẬP TRÌNH C#	11
1.1. .Net Framework.....	11
1.1.1. Giới thiệu	11
1.1.2. .Net Framework.....	12
1.1.3. Thư viện Common Language Runtime (CLR)	12
1.1.4. Mã IL	14
1.1.5. Tìm hiểu .Net Base class.....	14
1.2. Ngôn ngữ lập trình C#	15
1.2.1. Giới thiệu	15
1.2.2. Xây dựng ứng dụng trên C#.....	15
1.2.3. Ứng dụng Console.....	16
1.2.4. Ứng dụng Window	17
1.2.5. Ứng dụng Web	18
1.3. Một số cấu trúc cơ bản của ngôn ngữ lập trình C#	19
1.3.1. Biến và kiểu dữ liệu trong C#	19
1.3.2. Cấu trúc điều kiện.....	22
1.3.3. Vòng lặp.....	24
1.3.4. Phương thức	28
}.....	28
1.3.5. Cấu trúc Mảng (Array)	28
1.3.6. Cấu trúc chuỗi	30
1.4. Xử lý lỗi ngoại lệ.....	31
1.4.1. Giới thiệu	31
1.4.2. Đón bắt lỗi ngoại lệ	31
1.4.3. Xử lý ngoại lệ phát sinh	32
CHƯƠNG 2. WINDOWS FORM	37
2.1. Giới thiệu	37
2.2. Form.....	37
2.2.1. Tạo Form	37

2.2.3.	Thêm control vào form	39
2.2.4.	Xử lý các sự kiện cho các control	40
2.2.5.	Xử lý sự kiện của form	40
2.3.	Các control thông dụng.....	41
2.3.1.	Button.....	41
2.3.2.	Label	42
2.3.3.	Textbox	43
2.3.4.	Richtextbox	44
2.3.5.	Menu	45
2.3.6.	Toolbar.....	46
2.3.7.	ContextMenu.....	47
2.3.8.	Combobox.....	48
2.3.9.	Listbox	49
	ListBox.ObjectCollection.....	50
2.3.10.	Radiobutton	51
2.3.11.	Checkbox	51
2.3.12.	Tab	52
2.3.13.	GroupBox.....	53
2.3.14.	Treeview.....	53
2.3.15.	Listview.....	54
2.3.16.	Timer.....	56
2.4.	Form và sự thể hiện của các control	57
CHƯƠNG 3.	COMMON DIALOG.....	74
3.1.	Giới thiệu	74
3.2.	Mở tập tin.....	74
3.2.1.	Giới thiệu OpenFileDialog.....	74
3.2.2.	Một số thuộc tính và phương thức cơ bản	74
3.3.	Lưu tập tin.....	76
3.3.1.	Giới thiệu SaveFileDialog	76
3.3.2.	Một số thuộc tính và phương thức cơ bản	76
3.4.	Chọn Font chữ.....	78
3.4.1.	Giới thiệu FontDialog.....	78
3.4.2.	Một số thuộc tính và phương thức cơ bản	78
3.5.	Chọn màu	79

3.5.1.	Giới thiệu ColorDialog	79
3.5.2.	Một số thuộc tính và phương thức cơ bản	80
3.6.	In ấn	81
3.6.1.	PrintDialog	81
3.6.2.	Pagesetupdialog.....	83
3.6.3.	PrintPreviewDialog	84
CHƯƠNG 4.	TẬP TIN VÀ THƯ MỤC	89
4.1.	Giới thiệu	89
4.2.	Lớp Directory.....	89
4.2.1.	Giới thiệu	89
4.2.2.	Một số thuộc tính và phương thức cơ bản	89
4.3.	Lớp DirectoryInfo.....	91
4.3.1.	Giới thiệu	91
4.3.2.	Một số thuộc tính và phương thức cơ bản	91
4.4.	Lớp File (tập tin)	92
4.4.1.	Giới thiệu	92
4.4.2.	Một số phương thức cơ bản	92
4.5.	Lớp Path (Đường dẫn).....	93
4.5.1.	Giới thiệu	93
4.5.2.	Một số phương thức cơ bản	93
4.6.	Sử dụng luồng với FileStream	94
4.6.1.	Giới thiệu	94
4.6.2.	StreamWriter: Ghi dữ liệu vào File	94
4.6.3.	StreamReader: Đọc file từ nguồn	94
CHƯƠNG 5.	LẬP TRÌNH CƠ SỞ DỮ LIỆU VỚI ADO.NET	99
5.1.	Giới thiệu ADO.Net	99
5.1.1.	Giới thiệu	99
5.1.2.	Sự khác nhau giữa ADO.NET và ADO.....	100
5.1.3.	Các trình điều khiển cơ sở dữ liệu trong ADO.NET.....	101
5.2.	Đối tượng Connection	104
5.2.1.	SqlConnection	105
5.2.2.	OleDbConnection	107
5.3.	Đối tượng Command	108
5.3.1.	Giới thiệu	108

5.3.2.	Khai báo và khởi tạo đối tượng Command.....	108
5.3.3.	Một số thuộc tính và phương thức cơ bản	110
5.4.	Đối tượng DataReader.....	111
5.4.1.	Giới thiệu	111
5.4.2.	Một số thuộc tính và phương thức cơ bản	112
5.5.	Đối tượng DataAdapter	113
5.5.1.	Giới thiệu	113
5.5.2.	Một số thuộc tính và phương thức cơ bản	114
5.5.3.	SqlDataAdapter control và đối tượng SqlDataAdapter	115
5.6.	Dataset	120
5.6.1.	Giới thiệu	120
5.6.2.	Một số thuộc tính và phương thức cơ bản	121
5.7.	Hiển thị dữ liệu bằng DataGridView control.....	122
5.8.	DataTable.....	123
5.8.1.	Giới thiệu	123
5.8.2.	Một số thuộc tính và phương thức cơ bản	124
CHƯƠNG 6.	CRYSTAL REPORT.....	132
6.1.	Gíói thiệu	132
6.2.	Thiết kế Report.....	132
6.3.	Tương tác Report từ C#	137
6.4.	Cung cấp thông tin đăng nhập.....	138
6.5.	Điền dữ liệu vào Report từ đối tượng Dataset	139

DANH MỤC HÌNH

Hình 1.1. Các thành phần của Microsoft .NET Framework-----	11
Hình 1.2. Giao diện Visual Studio.Net 2008 -----	15
Hình 1.3. Tạo giao diện Console-----	16
Hình 1.4. Tạo giao diện cửa sổ đồ họa -----	17
Hình 1.5. Giao diện cửa sổ đồ họa cơ bản-----	17
Hình 1.6 Tạo giao diện Web Application-----	18
Hình 1.7 Kết quả ví dụ Biến-----	19
Hình 1.8 Kết quả ví dụ phép toán ba ngôi-----	21
Hình 1.9 Kết quả ví dụ If-----	22
Hình 1.10 Kết quả ví dụ switch case -----	23
Hình 1.11 Kết quả ví dụ vòng lặp For-----	24
Hình 1.12 Kết quả ví dụ vòng lặp do while -----	25
Hình 1.13 Kết quả ví dụ câu lệnh Go to -----	25
Hình 1.14 Kết quả ví dụ câu lệnh break -----	26
Hình 1.15 Kết quả ví dụ câu lệnh continue-----	26
Hình 1.16 Kết quả ví dụ Mảng -----	28
Hình 1.17 Kết quả ví dụ Chuỗi-----	29
Hình 1.18 Kết quả ví dụ try catch-----	31
Hình 2.1. Tạo giao diện Window Form -----	36
Hình 2.2. Giao diện cửa sổ đồ họa cơ bản-----	36
Hình 2.3. Form đăng nhập -----	37
Hình 2.4. Thêm control vào form-----	38
Hình 2.5. Form Load -----	39
Hình 2.6.Thuộc tính của Button-----	39
Hình 2.7. Form đăng nhập -----	40
Hình 2.8. Biểu diễn đối tượng Label -----	41
Hình 2.9.Hiển thị ảnh trong Label -----	42
Hình 2.10. Minh họa Textbox -----	43
Hình 2.11. Minh họa RichTextBox-----	44
Hình 2.12. Minh họa Menu-----	44
Hình 2.13. Minh họa ToolBar-----	46
Hình 2.14 Sử dụng ContextMenu-----	46
Hình 2.15. Minh họa ComboBox-----	48

Hình 2.16. Minh họa ListBox -----	49
Hình 2.17. Minh họa RadioButton-----	50
Hình 2.18. Minh họa CheckBox -----	51
Hình 2.19. Minh họa CheckBox -----	51
Hình 2.20. Minh họa GroupBox -----	52
Hình 2.21. Minh họa TreeView -----	53
Hình 2.22. Minh họa TreeView -----	54
Hình 2.23 Minh họa Timer -----	56
Hình 2.24. Minh họa Splitter -----	56
Hình 2.25. Minh họa Anchor của các control -----	57
Hình 2.26. Minh họa Anchor cho Button -----	57
Hình 2.27. Minh họa Dock của các control -----	58
Hình 2.28. Minh họa thuộc tính Dock cho Label -----	58
Hình 2.29. Minh họa Dock cho Labe-----	59
Hình 2.30. Minh họa thuộc tính Spliter -----	59
Hình 3.1. Minh họa OpenFileDialog -----	74
Hình 3.2. Minh họa SaveFileDialog-----	75
Hình 3.3. Hộp thoại chọn Font -----	77
Hình 3.4. Thông tin Font đã chọn-----	77
Hình 3.5. Hộp thoại ColorDialog -----	78
Hình 3.6. Hộp thoại PrintDialog -----	79
Hình 3.7. Thông tin hộp thoại PrintDialog -----	80
Hình 3.8. Hộp thoại PageSetup -----	82
Hình 3.9. PrintPreviewDialog -----	84
Hình 3.10. Hộp thoại kết nối với máy In -----	84
Hình 4.1 Minh họa Directory-----	88
Hình 4.2 Minh họa phương thức Delete của lớp File -----	91
Hình 4.3 Minh họa phương thức GetExtension-----	92
Hình 4.4 Giao diện ví dụ lưu – mở File-----	93
Hình 5.1 Vị trí của ADO.NET trong .NET Framework -----	98
Hình 5.2. Connection. -----	102
Hình 5.3. Vị trí của SqlConnection. -----	103
Hình 5.4. Đối tượng SqlCommand -----	106
Hình 5.5. SqlDataReader-----	109
Hình 5.6. Đối tượng DataAdapter -----	112

Hình 5.7. Data Connection-----	114
Hình 5.8. Add Connection -----	114
Hình 5.9. Data Connection-----	115
Hình 5.10. Data Adapter -----	115
Hình 5.11. Data Adapter Configuration-----	116
Hình 5.12. Form SqlDataAdapter và SqlConnection-----	116
Hình 5.13. Thuộc tính SqlDataAdapter-----	117
Hình 5.14. Query Builder -----	117
Hình 5.15. Minh họa Query Builder-----	118
Hình 5.16 DataSet-----	118
Hình 5.17. Hiển thị dữ liệu trong DataGridView control-----	120
Hình 5.18. DataTable-----	121
Hình 6.1. Khởi tạo Crystal Report-----	130
Hình 6.2. Report wizard-----	131
Hình 6.3. Kết nối cơ sở dữ liệu với Report wizard-----	131
Hình 6.4. Chọn trình kết nối SQL server -----	132
Hình 6.5. Thực hiện các thông số kết nối-----	132
Hình 6.6. Chọn bảng hoặc view cần xuất báo cáo-----	133
Hình 6.7. Chọn cột cần xuất báo cáo-----	133
Hình 6.8. Gom nhóm các cột-----	134
Hình 6.9. Lọc dữ liệu các cột-----	134
Hình 6.10. Chọn biểu đồ cho Report-----	135
Hình 6.11 Hiển thị Report trên Form-----	136
Hình 6.12 Minh họa form tìm kiếm -----	137
Hình 6.13 Form Report hiển thị các sinh viên tương ứng với lớp được chọn -----	137

DANH MỤC BẢNG

Bảng 1.1. Các thành phần quan trọng của CLR-----	12
Bảng 1.2. Kiểu dữ liệu số nguyên -----	18
Bảng 1.3. Kiểu dữ liệu số dấu chấm di động (Floating Point Types)-----	19
Bảng 1.4. Kiểu Boolean-----	19
Bảng 1.5. Kiểu Character Type -----	19
Bảng 1.6. Kiểu tham khảo định nghĩa trước-----	19
Bảng 1.7. Các ký tự escape thông dụng-----	19
Bảng 1.8. Các loại toán tử trong C# -----	20
Bảng 1.9. Danh sách các toán tử trong C#-----	21
Bảng 1.10. Một số thuộc tính và phương thức cơ bản -----	29
Bảng 5.1 So sánh đặc điểm ADO và ADO.NET -----	99
Bảng 5.2 Các loại trình điều khiển của SQLServer và OLEDB -----	101

CHƯƠNG 1. TỔNG QUAN NGÔN NGỮ LẬP TRÌNH C#

Mục tiêu:

Sau khi học xong chương này, sinh viên có khả năng:

- Nhận thức rõ những tính năng vượt trội của .Framework so với các ngôn ngữ khác trong việc triển khai các ứng dụng.
- Xây dựng được các ứng dụng trên C#.Net: Console, Winform, Webform.
- Biết cách sử dụng các cấu trúc cơ bản của ngôn ngữ lập trình lập trình C#.

1.1. .Net Framework

1.1.1. Giới thiệu

Visual Studio.NET cung cấp một môi trường phát triển mức cao để xây dựng các ứng dụng trên .NET Framework. Với bộ Visual Studio.NET chúng ta có thể đơn giản hoá việc tạo, triển khai và tiếp tục phát triển các ứng dụng Window, Web và các dịch vụ Window, Web có sẵn một cách an toàn, bảo mật và khả năng biến đổi được. Visual Studio.NET là một bộ đa ngôn ngữ các công cụ lập trình. Ngoài C# (Visual C#.NET), Visual Studio.NET còn hỗ trợ Visual Basic.Net , Visual C++.Net, Visual J#.NET và các ngôn ngữ script như VBScript và JScript. Tất cả các ngôn ngữ này đều cho phép truy cập vào .NET Framework.

Visual Basic.NET (VB.NET) là ngôn ngữ lập trình hướng đối tượng do Microsoft thiết kế lại từ đầu. VB.NET không kế thừa VB6 hay bổ sung, phát triển từ VB6 mà là một ngôn ngữ lập trình hoàn toàn mới trên nền .NET Framework. VB.NET cũng không phải là VB phiên bản 7. Thực sự, đây là ngôn ngữ lập trình mới và rất mạnh, không những lập nền tảng vững chắc theo kiểu hướng đối tượng như các ngôn ngữ lập trình khác như: C++, Java mà còn dễ học, dễ phát triển theo kiểu trực quan của VB6.

Visual C++ .NET là phiên bản kế tiếp của Microsoft Visual C++ 6.0. Như chúng ta thấy Microsoft Visual C++ là công cụ C++ hiệu quả nhất để tạo ra những ứng dụng hiệu năng cao cho Windows và cho World Wide Web. Hầu như tất cả các phần mềm tốt nhất từ những trình duyệt Web cho đến các ứng dụng đều được xây dựng bằng hệ thống phát triển Microsoft Visual C++. Visual C++ .NET mang đến một cấp độ mới về hiệu năng so với Visual C++ mà không làm ảnh hưởng đến tính mềm dẻo, hiệu suất thực hiện cũng như điều khiển.

Visual J# .NET là một công cụ phát triển cho các nhà phát triển ngôn ngữ Java để xây dựng các ứng dụng và các dịch vụ trên nền Microsoft .NET Framework. Visual J#.NET cho phép những người phát triển ngôn ngữ Java có thể chuyển tiếp vào thế giới

của các dịch vụ Web XML và cải thiện đáng kể khả năng vận hành của các chương trình viết bằng ngôn ngữ Java với những phần mềm hiện tại được viết bằng nhiều ngôn ngữ lập trình khác nhau. Việc tích hợp dễ dàng, khả năng thao tác vận hành với nhau và sự chuyển giao các kỹ năng hiện tại và những đầu tư mà Visual J# .NET cho phép có thể tạo ra một cơ hội lớn cho khách hàng muốn phát triển các ứng dụng và các dịch vụ Web XML với ngôn ngữ Java trên nền .NET Framework.

JScript .NET là bộ thực hiện của Microsoft cho JavaScript. Jscript.NET thêm rất nhiều đặc tính mới vào Jscript, bao gồm cả việc hỗ trợ trực tiếp các kỹ thuật lập trình hướng đối tượng.

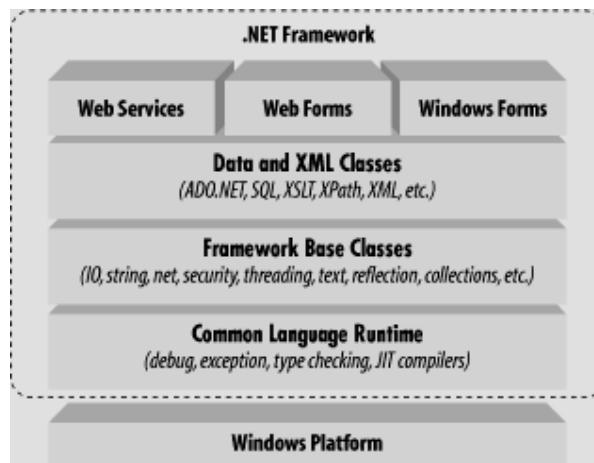
1.1.2. .Net Framework.

.NET gồm có hai phần: **Framework** và **Integrated Development Environment (IDE)**.

Framework cung cấp tất cả những gì cần thiết, căn bản. Chữ Framework có nghĩa là cái khung hay khung cảnh trong đó ta dùng những hạ tầng cơ sở theo một qui ước nhất định để công việc trôi chảy.

IDE cung cấp một môi trường giúp ta triển khai dễ dàng, nhanh chóng hơn. Nếu không có IDE ta cũng có thể dùng Notepad và command line để triển khai nhưng nó chậm hơn và đòi hỏi lập trình viên phải thuộc nhiều thư viện lớp.

Framework của .NET bao bọc hệ điều hành (OS) lại, khiến lập trình viên không phải quan tâm đến những việc liên hệ đến OS như thao tác với tập tin (file handling) và memory allocation (phân phát bộ nhớ). Nó cho ta mọi tầng lớp triển khai phần mềm từ việc trình bày (presentation) cho đến các bộ phận (components) và dữ liệu (data) (hình 1).



Hình 1.1. Các thành phần của Microsoft .NET Framework

1.1.3. Thư viện Common Language Runtime (CLR)

Đã nhiều năm qua, các lập trình viên thường sử dụng các hàm thư viện runtime (gọi lúc thực thi), để hỗ trợ những thao tác xử lý của chương trình, như truy xuất file, thư mục, hoặc đặt ngày tháng hệ thống, thực hiện những thao tác tính toán số học, như

tính giá trị sin của một góc vuông hoặc lấy căn bậc hai của một giá trị.

Thư viện Runtime (bảng 1.1) gồm có hàng trăm tới hàng nghìn hàm giúp người lập trình xây dựng được rất nhiều loại chương trình. Trước đây, đa số các thư viện runtime phụ thuộc vào đặc tính ngôn ngữ, có nghĩa là những hàm thư viện thực hiện viết cho người lập trình Visual Basic thì không thể sử dụng được cho người lập trình C++.

Bảng 1.1. Các thành phần quan trọng của CLR.

Common type system
Hệ thống kiểu chung.
Memory management
Quản lý bộ nhớ
Security machinisms
Cơ chế an toàn.
IL Compilers
Trình biên dịch ngôn ngữ chung IL
Metadata
Hỗ trợ định nghĩa dữ liệu bằng siêu dữ kiện
Lớp libraries
Thư viện lớp.

Môi trường .NET được xây dựng độc lập về ngôn ngữ đối với thư viện runtime, đó là thư viện thực thi ngôn ngữ chung Common Langugue Runtime (CLR) và thư viện lớp BCL (hỗ trợ hàng nghìn định nghĩa lớp).

CLR là tâm điểm của .NET Framework. Đây là một *hàm máy* để chạy các tính năng của .NET. Trong .NET tất cả mọi ngôn ngữ lập trình đều được biên dịch ra Microsoft Intermediate Language (IL). Do bắt buộc mọi ngôn ngữ đều phải dùng cùng các loại kiểu dữ liệu (gọi là Common Type System hay hệ thống kiểu chung) nên CLR có thể kiểm soát mọi giao diện, gọi giữa các thành phần và cho phép các ngôn ngữ có thể tích hợp với nhau một cách thông suốt.

Khi chạy một ứng dụng .NET, nó sẽ được biên dịch bằng một bộ biên dịch JIT (Just-In-Time có nghĩa là chỉ phần mã cần xử lý mới được biên dịch) rất hiệu năng ra mã máy để chạy. Điểm này giúp ứng dụng .NET chạy nhanh hơn mã thông dịch của Java trong Java Virtual Machine (máy ảo Java). Just-In-Time cũng có nghĩa là chỉ phần mã nào cần xử lý trong lúc ấy mới được biên dịch.

Ngoài việc cung cấp và quản lý bộ nhớ, CLR còn xử lý công việc “*garbage collection*” (garbage collection). Trước đây mỗi khi một DLL (thư viện liên kết động) được nạp vào bộ nhớ, hệ thống sẽ nhận có bao nhiêu tác vụ dùng nó để khi tác vụ cuối cùng

chấm dứt thì hệ thống giải phóng DLL này và trả lại phần bộ nhớ nó dùng trước đây cho hệ thống để dùng vào việc khác. Nếu chương trình cung cấp (allocate) bộ nhớ để sử dụng mà không giải phóng (dispose) thì đến một lúc nào đó bộ nhớ sẽ bị cạn và chúng ta sẽ phải khởi động lại hệ điều hành. Và bây giờ, .NET sử dụng một quá trình độc lập để xử lý việc *gom rác*. Tác động phụ ở đây là khi ta đã hủy (dispose) một đối tượng rồi, ta vẫn không biết chắc chắn chừng nào nó mới thực sự biến mất. Vì bộ phận *gom rác* là một quá trình ưu tiên mức thấp, chỉ khi nào bộ nhớ hệ thống gần cạn nó mới nâng cao độ ưu tiên lên. Ngoài *gom rác*, CLR còn thực hiện các chức năng khác như bảo mật. Các dịch vụ chung này đều được quản lý một cách tự động.

Tóm lại, CLR cho phép việc phát triển các ứng dụng một cách dễ dàng hơn, cung cấp một môi trường thực thi an toàn và hiệu năng, hỗ trợ đa ngôn ngữ và đơn giản hóa việc triển khai và quản lý các ứng dụng.

1.1.4. Mã IL

.NET cho phép các ngôn ngữ lập trình khác nhau có thể được biên dịch ra một ngôn ngữ trung gian, gọi là **Microsoft Intermediate Language (MSIL)** hay gọi tắt là **Intermediate Language (IL)**, giống như p-code hay Java Byte-Code. Nếu trong Java ta cần Java Virtual Machine (JVM) thì ở đây ta cần CLR để chạy chương trình. Độc lập với phần cứng CPU, IL code chạy trong CLR được nói là **managed code**. Tức là CLR lãnh trách nhiệm quan sát, không cho code thực hiện sai như nhảy đến một chỗ không tưởng, viết chồng lên bộ nhớ của người khác hay đi ngoài giới hạn của một mảng, ...

Khi biên dịch chương trình, trình biên dịch sẽ chuyển mã nguồn thành mã máy. Các chỉ thị này phụ thuộc vào từ loại CPU như Intel, Macintosh, Motorola... mã này được gọi là mã native.

.NET không biên dịch chương trình ra mã máy như các file .exe thông thường; thay vào đó chương trình được biên dịch ra mã IL.

Chương trình .exe của .NET phải chạy trên bộ khung của .NET Framework, có thể coi đây là máy ảo dùng diễn dịch mã IL thành mã native để các chương trình .NET thực thi.

1.1.5. Tìm hiểu .Net Base class

Thư viện các lớp cơ sở .NET Framework cung cấp một tập các lớp (“APIs”), hướng đối tượng, có thứ bậc và có thể mở rộng và chúng được sử dụng bởi bất cứ ngôn ngữ lập trình nào. Như vậy, tất cả các ngôn ngữ từ Jscript cho tới C++ trở nên bình đẳng, và các nhà phát triển có thể tự do lựa chọn ngôn ngữ mà họ vẫn quen dùng.

Tập các lớp, các kiểu giá trị và giao diện này được tổ chức bằng một hệ thống các Namespace. Một điều rất quan trọng là chúng ta không chỉ giới hạn ở các Namespace này. Ta có thể tự tạo ra Namespace và sử dụng chúng trong ứng dụng của mình hay cũng có thể sử dụng các Namespace của đối tác thứ ba đang có trên thị trường. Một ví dụ cho trường hợp này là Namespace *System.Data.Oracle*.

.NET Framework được tạo bởi từ hàng trăm lớp. Nhiều ứng dụng mà ta xây dựng trong .NET đang tận dụng các lớp này theo cách này hay cách khác. Vì số lượng các lớp là quá lớn, .NET Framework tổ chức các lớp này vào một cấu trúc lớp được gọi là một namespace. Có một số lượng lớn các namespace và chúng được tổ chức theo cách dễ hiểu và minh bạch. System là một Namespace cơ sở trong .NET Framework. Tất cả các Namespace được cung cấp trong .NET framework bắt đầu với Namespace cơ sở này. Ví dụ, những lớp phục vụ việc truy cập và thao tác dữ liệu được tìm thấy trong Namespace *System.Data*. Những ví dụ khác bao gồm *System.IO*, *System.XML*, *System.Collections*, *System.Drawing* và .v.v..

Ví dụ 1.1: minh họa cách sử dụng Namespace trong C#.

```
using System;
class HelloWorld
{
    public static void Main()
    {
        Console.WriteLine ("Hello World ! ");
    }
}
```

1.2. Ngôn ngữ lập trình C#

1.2.1. Giới thiệu

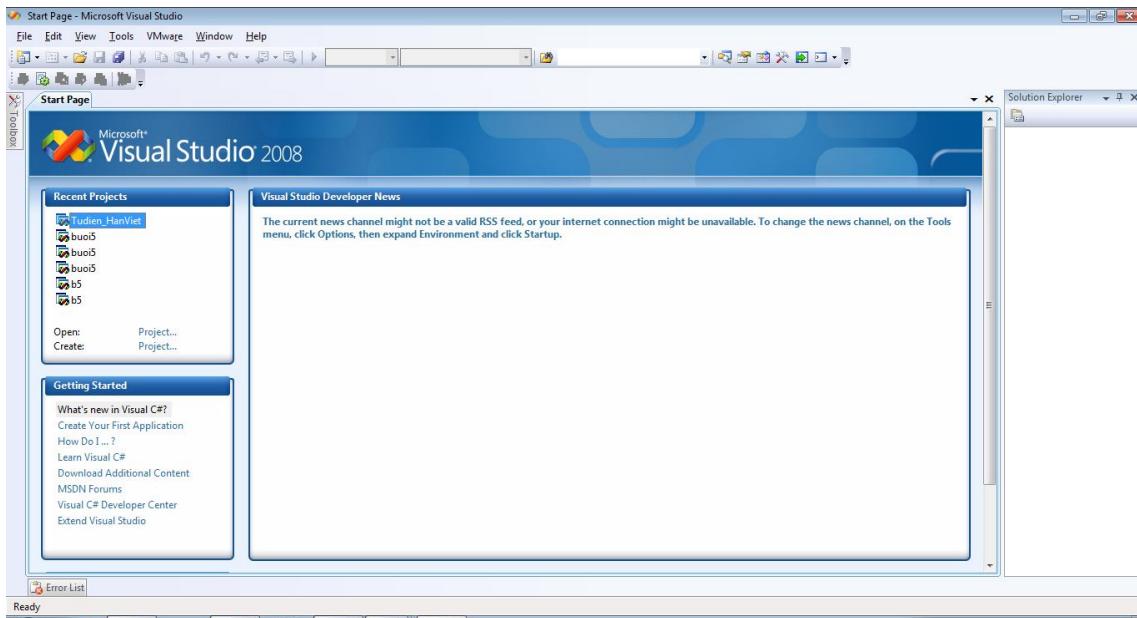
Visual C# là một ngôn ngữ lập trình đơn giản, hiện đại, hướng đối tượng an toàn kiểu (type-safe) và có nguồn gốc từ các ngôn ngữ C và C++. C# là một ngôn ngữ rất thân thiện với người lập trình C và C++. C# là kết quả của việc kết hợp hiệu năng cao của Visual Basic và sức mạnh của C++. C# được Microsoft giới thiệu để xây dựng với Web và đòi hỏi quyền được cung cấp một môi trường đồng bộ với HTML, XML và SOAP. Tóm lại C# là một ngôn ngữ lập trình hiện đại và là một môi trường phát triển đầy tiềm năng để tạo ra các dịch vụ Web XML, các ứng dụng dựa trên Microsoft .NET và cho cả nền tảng Microsoft Windows cũng như tạo ra các ứng dụng Internet thế hệ kế tiếp một cách nhanh chóng và hiệu quả.

1.2.2. Xây dựng ứng dụng trên C#

Sử dụng C#, ta có thể tạo ra rất nhiều kiểu ứng dụng, ở đây ta quan tâm đến ba kiểu ứng dụng chính: Console, Window và ứng dụng Web.

Để xây dựng các ứng dụng trên C# ta làm theo các bước sau đây:

```
Start → Programs → MS Visual Studio.Net 2008 (hoặc phiên bản cao hơn).
```



Hình 1.2. Giao diện Visual Studio .Net 2008

1.2.3. Ứng dụng Console

Ứng dụng Console là ứng dụng có giao diện text, chỉ xử lý nhập xuất trên màn hình Console, tương tự với các ứng dụng DOS trước đây.

Ứng dụng Console thường đơn giản, ta có thể nhanh chóng tạo chương trình hiển thị kết xuất trên màn hình. Do đó, các minh họa, ví dụ ngắn gọn ta thường sử dụng dạng chương trình Console để thể hiện.

Để tạo ứng dụng Console ta làm như sau

Trong Visual Studio, chọn File → New → Project. Visual Studio sẽ trình bày hộp thoại New Project. (hình 1.3)

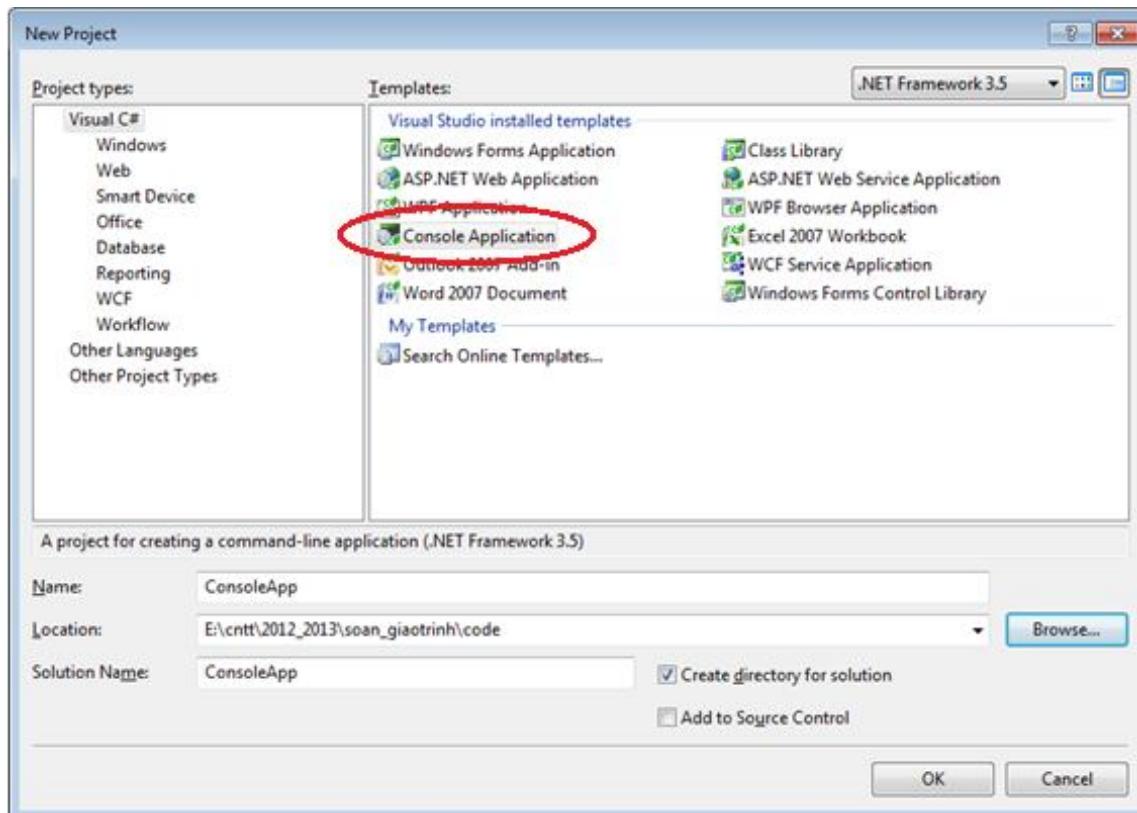
Trong hộp thoại New Project, kích biểu tượng ứng dụng ConSole (Console Application). Trong ô name, gõ tên chương trình (dự án). Trong ô Location, gõ tên của thư mục mà ta muốn Visual Studio lưu dự án. Nhấn OK.

Visual Studio sẽ hiển thị cửa sổ. Ta nhập code vào trong cửa sổ này.

Ví dụ 1.2: Chương trình Console sau đây sử dụng hai phương thức Console.ReadLine và Console.WriteLine để nhập và xuất số nguyên a ra màn hình:

```
-----  
static void Main(string[] args)  
{  
    int a = int.Parse(Console.ReadLine());  
    Console.WriteLine("a = " + a);  
    Console.ReadLine();  
}
```

Chạy chương trình: Để chạy chương trình, ta chọn Debug → Start hoặc nhấn F5, Visual Studio sẽ hiển thị cửa sổ Console cho phép nhập và in số nguyên.



Hình 1.3. Tạo giao diện Console

1.2.4. Ứng dụng Window

Là ứng dụng được hiển thị với giao diện cửa sổ đồ họa. Chúng ta chỉ cần kéo và thả các điều khiển (control) lên cửa sổ Form. Visual Studio sẽ sinh mã trong chương trình để tạo ra, hiển thị các thành phần trên cửa sổ.

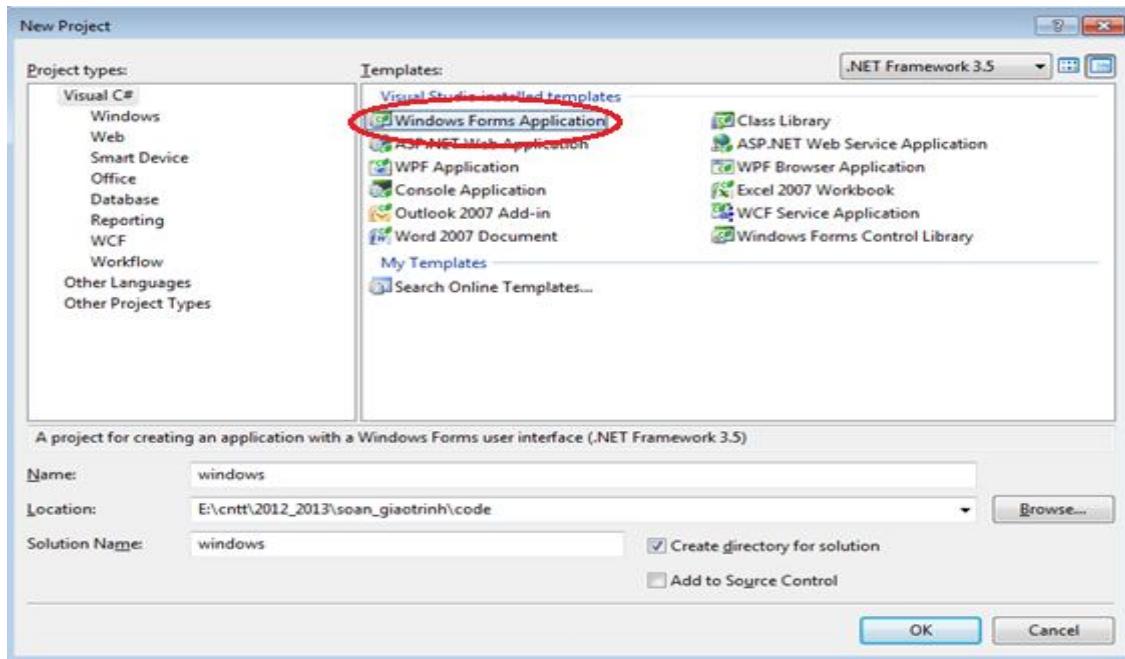
Để tạo ứng dụng Window ta làm như sau:

File → New → Project. Visual Studio sẽ trình bày hộp thoại New Project (hình 1.4).

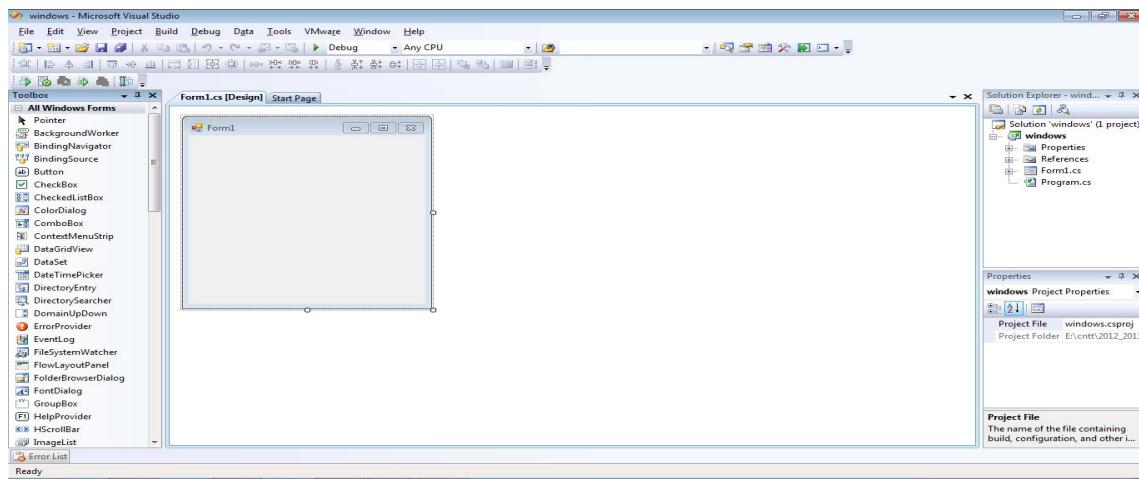
Trong hộp thoại New Project, kích biểu tượng ứng dụng Windows (Windows Application). Trong ô Name, gõ tên mô tả chương trình mà ta dự định tạo (tên dự án). Tiếp theo, trong ô Location, gõ tên của thư mục mà ta muốn Visual Studio lưu dự án. Nhấn OK. Visual Studio sẽ hiển thị cửa sổ thiết kế (hình 1.5). Ta có thể kéo và thả các thành phần giao diện (control) lên Form.

Để hiển thị cửa sổ Toolbox chứa những điều khiển mà ta có thể kéo và thả lên Form, ta chọn View → Toolbox từ menu.

Biên dịch và chạy chương trình: Để biên dịch chương trình, ta chọn Build → Build Solution. Để chạy chương trình, ta chọn Debug → Start. Nếu ta có thay đổi nội dung của Form, như đặt thêm điều khiển khác lên Form chẳng hạn, ta phải yêu cầu Visual Studio biên dịch lại.



Hình 1.4. Tạo giao diện cửa sổ đồ họa



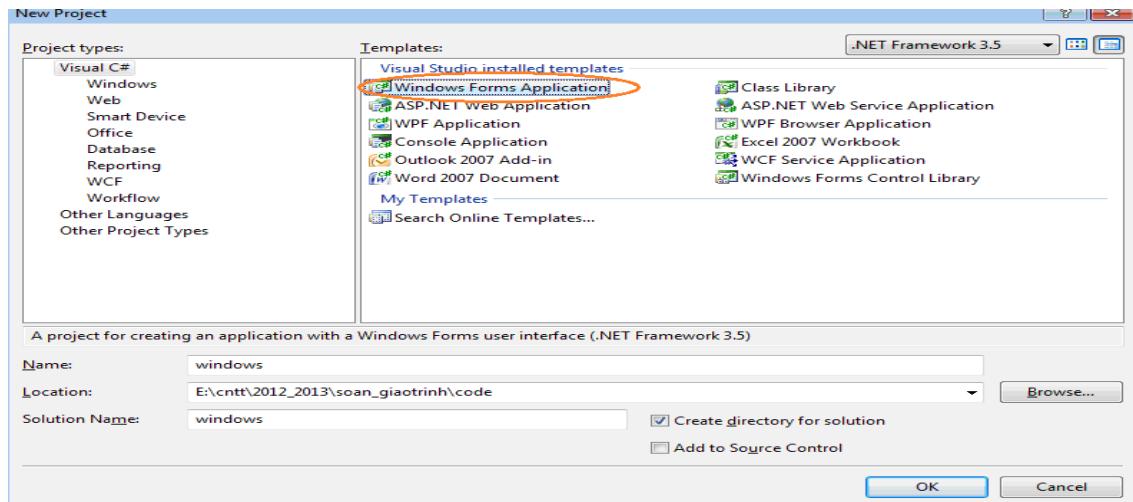
Hình 1.5. Giao diện cửa sổ đồ họa cơ bản

1.2.5. Ứng dụng Web

Môi trường .NET cung cấp công nghệ ASP.NET giúp xây dựng những trang Web động. Để tạo ra một trang ASP.NET, người lập trình sử dụng ngôn ngữ biên dịch như C# hoặc C# để viết mã. Để đơn giản hóa quá trình xây dựng giao diện người dùng cho trang Web, .NET giới thiệu công nghệ Webform. Cách thức tạo ra các Web control tương tự như khi ta xây dựng ứng dụng trên Window Form.

Để tạo ứng dụng Web ta làm như sau

File → New → Project → Visual Basic Projects → ASP.NET Web Application (hình 1.6)



Hình 1.6 Tạo giao diện Web Application

1.3. Một số cấu trúc cơ bản của ngôn ngữ lập trình C#

1.3.1. Biến và kiểu dữ liệu trong C#

Để lưu thông tin trong quá trình mã thực thi, chương trình đặt dữ liệu vào trong các biến (variable). Một chương trình có thể sử dụng biến để cất giữ chuỗi tên, giá trị số, ngày tháng, ...

Bảng 1.2. Kiểu dữ liệu số nguyên

Tên	Mô tả	Khoảng giá trị (nhỏ nhất:lớn nhất)
Sbyte	8-bit signed integer	-128:127 (-2 ⁷ :2 ⁷ -1)
Short	16-bit signed integer	-32,768:32,767 (-2 ¹⁵ :2 ¹⁵ -1)
Int	32-bit signed integer	-2,147,483,648:2,147,483,647 (-2 ³¹ :2 ³¹ -1)
Long	64-bit signed integer	-9,223,372,036,854,775,808:9,223,372,036,854,775,807 (-2 ⁶³ :2 ⁶³ -1)
Byte	8-bit signed integer	0:255 (0:2 ⁸ -1)
Ushort	16-bit signed integer	0:65,535 (0:2 ¹⁶ -1)
Uint	32-bit signed integer	0:4,294,967,295 (0:2 ³² -1)
Ulong	64-bit signed integer	0:18,446,744,073,709,551,615(0:2 ⁶⁴ -1)

Bảng 1.3. Kiểu dữ liệu số dấu chấm di động (Floating Point Types)

Name	Khoảng giá trị
Float	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$
Double	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$

Bảng 1.4. Kiểu Boolean

Tên	Giá trị
bool	true hoặc false

Bảng 1.5. Kiểu Character Type

Tên	Giá trị
Char	Represents a single 16-bit (Unicode) character

Bảng 1.6. Kiểu tham khảo định nghĩa trước

C# hỗ trợ hai kiểu dữ liệu được định nghĩa trước:

Tên	Mô tả
object	Kiểu dữ liệu gốc, được kế thừa bởi tất cả các kiểu dữ liệu khác
String	Chuỗi ký tự Unicode

Bảng 1.7. Các ký tự escape thông dụng

Dãy escape	Ký tự
'	'
\"	"
\\	\
\0	Null
\b	Backspace
\n	Xuống dòng mới
\r	Quay về đầu dòng
\t	Tab

Khai báo biến trong chương trình

<kiểu dữ liệu> <tên biến>;

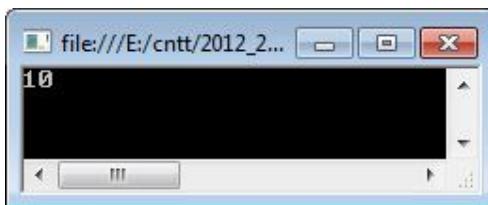
Ví dụ 1.3: khai báo và khởi tạo biến. Sau đó hiển thị ra màn hình:

```

static void Main(string[] args)
{
    int a = 10;
    Console.WriteLine(a);
    Console.Read
}

```

Kết quả:



Hình 1.7 Kết quả ví dụ Biến

Chuyển đổi giá trị các biến

Một kiểu của biến chỉ rõ phạm vi của những giá trị mà một biến có thể cất giữ. Trong chương trình, sẽ có lúc ta phải gán một giá trị của một biến cho một biến có kiểu giá trị khác.

Sau đây là một số hàm chuyển đổi kiểu dữ liệu từ string sang số:

- int int.parse (string s): Chuyển chuỗi sang số nguyên.
- float float.parse (string s): chuyển chuỗi sang số thực.
- double double.parse (string s).

Các toán tử

Bảng 1.8. Các loại toán tử trong C#

Loại	Toán tử
Số học	+ - * / %
Luận lý	&& (và) (hoặc) ! (phủ định)
Cộng chuỗi	+
Tăng giảm	++ --
So sánh	== != < > <= >=
Gán	= += -= *= /= %=

Toán tử thu gọn (shortcut operators)

Bảng dưới đây trình bày một danh sách đầy đủ của toán tử có giá trị trong C#:

Bảng 1.9. Danh sách các toán tử trong C#

Toán tử thu gọn	Tương đương
$x++, ++x$	$x = x + 1$
$x--, --x$	$x = x - 1$
$x += y$	$x = x + y$
$x -= y$	$x = x - y$
$x *= y$	$x = x * y$
$x /= y$	$x = x / y$
$x \% y$	$x = x \% y$

Toán tử ba ngôi (ternary operator)

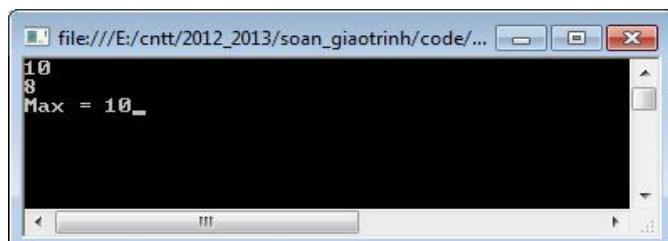
Cú pháp:

<điều kiện> ? <giá trị tương ứng điều kiện đúng> : <giá trị tương ứng điều kiện sai>

Ví dụ 1.4: hiển thị số lớn nhất trong 2 số nguyên và b

```
int a = int.Parse(Console.ReadLine());
int b = int.Parse(Console.ReadLine());
int m = a > b ? a : b;
Console.WriteLine("Max = " + m);
```

Kết quả:



Hình 1.8 Kết quả ví dụ phép toán ba ngôi

1.3.2. Cấu trúc điều kiện

Cấu trúc If.....Else

```
if (<điều kiện>
    <các câu lệnh>
```

Hoặc

```

if (<điều kiện>
    <các câu lệnh>
else if (< điều kiện>
    <các câu lệnh>
.....
else
    <các câu lệnh>

```

Điều kiện trong các cấu trúc trên trả về một trong hai giá trị là: True hoặc False. Trong biểu thức điều kiện thường chứa các toán tử sau: >, >=, <, <=, =, != (khác)...

Ví dụ 1.5: minh họa cách sử dụng phát biểu If để tìm số lớn nhất trong hai số nguyên a, b

```

static void Main(string[] args)
{
    int a, b;
    a = int.Parse(Console.ReadLine());
    b = int.Parse(Console.ReadLine());
    int m = a;
    if (m < b)
        m = b;
    Console.WriteLine("Max = " + m);
    Console.ReadLine();
}

```

Kết quả:



Hình 1.9 Kết quả ví dụ If

Cấu trúc switch case

Khi lập trình về điều kiện xử lý, ta thường chọn cấu trúc ifelse. Bên cạnh đó, C# còn cung cấp thêm cấu trúc switch giúp chương trình có thể đơn giản hóa mã phải viết để điều khiển các biểu thức lựa chọn.

Cấu trúc

```

switch (<biểu thức>)
{
    case <giá trị 1>:
        <Khối lệnh 1>;
    case <giá trị 2>:
        <Khối lệnh 2>;
    .....
    [default: câu lệnh mặc định]
}

```

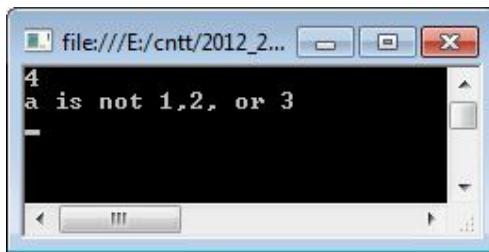
Ví dụ 1.6: kiểm tra a thỏa mãn các trường hợp 1, 2, 3 không. Nếu không thỏa mãn sẽ thực thi trường hợp default:

```

int a = int.Parse(Console.ReadLine());
switch (a)
{
    case 1:
        Console.WriteLine("a =1");
        break;
    case 2:
        Console.WriteLine("a =2");
        break;
    case 3:
        Console.WriteLine("a =3");
        break;
    default:
        Console.WriteLine("a is not 1,2, or 3");
        break;
}

```

Kết quả tương ứng với a = 4



Hình 1.10 Kết quả ví dụ switch case

1.3.3. Vòng lặp

Vòng lặp cho phép thực hiện lặp lại một nhóm phát biểu lệnh với số lần xác định hoặc cho đến khi một điều kiện nào đó được ước lượng là True.

Vòng lặp với số lần xác định

```
for ( [ <biểu thức 1>]; [ <biểu thức 2> ] ; [ <biểu thức 3>])
    <câu lệnh>;
```

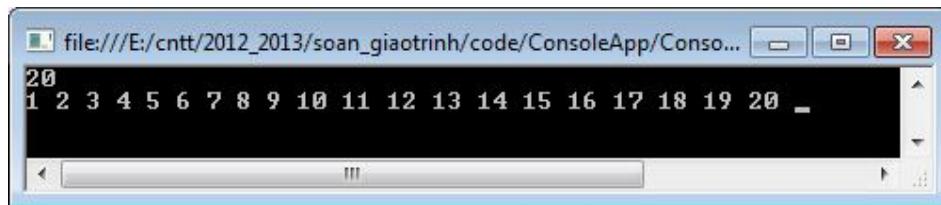
- Bước 1: Xác định giá trị của biểu thức 1, thông thường là giá trị khởi tạo.
- Bước 2: Xác định giá trị của biểu thức 2, thông thường là biểu thức điều kiện.
- Bước 3: Nếu điều kiện biểu thức 2 sai thì sẽ thoát vòng lặp for. Nếu biểu thức 2 đúng thì máy sẽ thực hiện <câu lệnh>
- Bước 4: Tính giá trị của biểu thức 3 và quay lại Bước 2

Vòng lặp sẽ lặp các bước theo trình tự <biểu thức 2> -> <câu lệnh> -> <biểu thức 3>. Thực hiện đến khi nào biểu thức điều kiện của <bước 2> bị sai thì dừng vòng lặp.

Ví dụ 1.7: hiển thị số 1 đến n (n là số nguyên dương nhập từ bàn phím).

```
int n = int.Parse(Console.ReadLine());
for (int i = 1; i <= n; ++i)
    Console.WriteLine(i + " ");
```

Kết quả:



Hình 1.11 Kết quả ví dụ vòng lặp For

Vòng lặp không xác định

Cấu trúc 1: While

Khi ta không biết chắc là vòng lặp sẽ thực hiện bao nhiêu lần thì tốt nhất là dùng while. Khác với vòng lặp for, trong while ta phải khởi tạo giá trị ban đầu và tăng giá trị cho biến điều kiện. Nếu biểu thức điều kiện là True thì thực thi những dòng code trong <Các câu lệnh>.

```
while (<điều kiện>)
    <Các câu lệnh>;
```

Ví dụ 1.8: minh họa cách sử dụng vòng lặp while để thực hiện ví dụ ở vòng lặp for.

```
int n = int.Parse(Console.ReadLine());
int i = 1;
while (i <= n)
{
    Console.WriteLine(i + " ");
    ++i;
}
```

Kết quả tương tự như kết quả ở vòng lặp for.

Câu trúc 2: do while

Cú pháp:

```
do
{
    <các câu lệnh>
} while (<điều kiện>);
```

Khác với vòng lặp while, vòng lặp do while sẽ thực hiện <các câu lệnh> xong rồi mới kiểm tra <điều kiện> có đúng hay không. Nếu đúng thì thực hiện tiếp <các câu lệnh>, nếu sai thì thoát khỏi vòng lặp. Do đó, đối với vòng lặp này, <các câu lệnh> tối thiểu được thực hiện một lần bất kể <điều kiện> đúng hay sai.

Ví dụ 1.9: Nhập một dãy số nguyên dương, nhập đến số 0 thì dừng, tính tổng dãy số đó.

☞ **Lưu ý:** Ta phải nhập số trước khi kiểm tra nó có phải là số 0 hay không -> dùng do while

```
int n, s = 0;
do
{
    n = int.Parse(Console.ReadLine());
    if (n > 0)
        s = s + n;
} while (n != 0);
Console.WriteLine("Tong = " + s);
```

Kết quả:



Hình 1.12 Kết quả ví dụ vòng lặp do while

Câu lệnh goto: Nhảy đến nhãn xác định

Cú pháp:

```
goto <Label>;
```

Trong đó, <Label> là nhãn xác định nơi mà dòng lệnh nhảy đến.

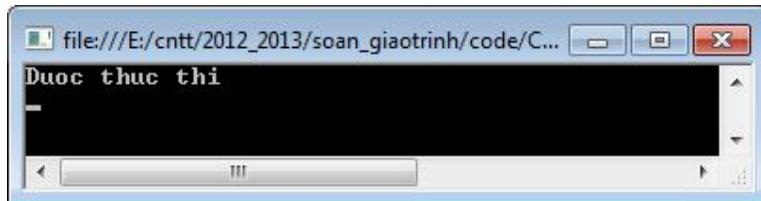
Ví dụ 1.10: minh họa goto, chương trình hiển thị kết quả “Được thực thi” mặc dù câu lệnh in thông báo này nằm sau câu lệnh in thông báo “Không được thực thi”.

```

goto Label1;
Console.WriteLine("Khong duoc thuc thi");
Label1:
    Console.WriteLine("Duoc thuc thi");

```

Kết quả:



Hình 1.13 Kết quả ví dụ câu lệnh Go to

Câu lệnh break: Thoát khỏi vòng lặp đang chứa nó.

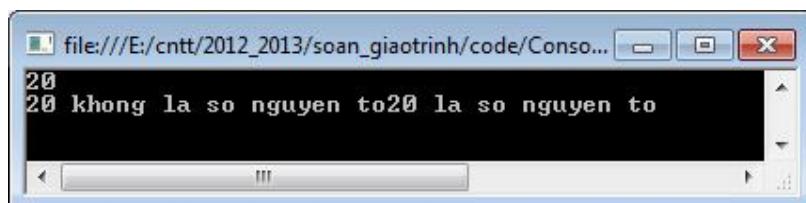
Ví dụ 1.11: kiểm tra số nguyên tố sau sẽ minh họa câu lệnh break.

```

int n = int.Parse(Console.ReadLine());
for (int i = 2; i<=n/2; ++i)
    if (n % i == 0)
    {
        Console.WriteLine(n + " khong la so nguyen to");
        break;
    }
Console.WriteLine(n + " la so nguyen to");

```

Kết quả



Hình 1.14 Kết quả ví dụ câu lệnh break

Câu lệnh continue: Bỏ qua lần lặp hiện hành, nhảy đến lần lặp kế tiếp.

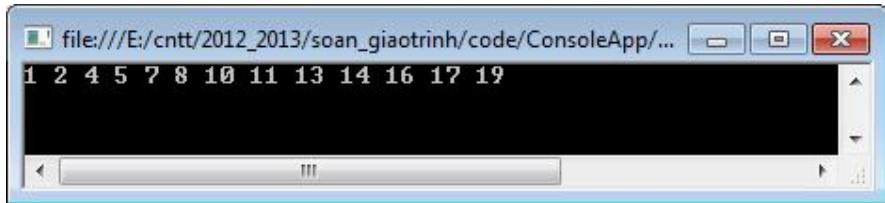
Ví dụ 1.12: hiển thị các số nhỏ hơn 20 và không chia hết cho 3

```

for (int i = 1; i < 20; ++i)
{
    if (i % 3 == 0)
        continue;
    Console.WriteLine(i + " ");
}

```

Kết quả



Hình 1.15 Kết quả ví dụ câu lệnh continue

1.3.4. Phương thức

Khi chương trình trở nên lớn và phức tạp, người ta thường chia chương trình thành những đơn thể nhỏ hơn, mỗi đơn thể sẽ thực hiện một nhiệm vụ cụ thể.

Để tổ chức các lệnh gom nhóm theo nhiệm vụ, ta thường dùng hàm (function, method)

Cú pháp:

```
<kiểu dữ liệu trả về> <tên hàm> ([<các tham số>])
{
    <các câu lệnh xử lý>
}
```

- <tên hàm> phải đặt hợp lệ (không trùng tên biến và từ khóa).
- <Kiểu dữ liệu trả về> là kiểu của kết quả trả về của hàm. Hàm không có giá trị trả về thì ta dùng kiểu void.
- <Các tham số>: liệt kê các tham số đầu vào của hàm, các được ngăn cách nhau bởi dấu phẩy (,)

Ví dụ 1.13: phương thức sau trả trả về giá trị lớn nhất của hai số nguyên a, b:

```
int max(int a, int b)
{
    int m = a;
    if (m < b)
        m = b;
    return m;
}
```

1.3.5. Cấu trúc Mảng (Array)

Trong một chương trình, biến cho phép chương trình lưu trữ và khôi phục các giá trị khi chương trình thực thi. Thông thường biến chỉ cất giữ một giá trị tại mỗi thời điểm. Tuy nhiên, ta có thể sử dụng biến ở dạng cấu trúc mảng (array) để cất giữ cùng lúc nhiều giá trị.

Để tạo ra mảng, ta khai báo một biến theo kiểu xác định sau đó chỉ rõ số phần tử mảng mà biến sẽ cất giữ. Ta khai báo một mảng như sau:

```
<kiểu dữ liệu>[] <tên mảng>;
```

Ví dụ: int[] a;

Khai báo mảng với số phần tử xác định

<kiểu dữ liệu>[] <tên mảng> = new <kiểu dữ liệu>[<số phần tử>;

Ví dụ: int[] a = new int[10]; //Khai báo và cấp phát cho mảng a 10 phần tử int

Một số thuộc tính và hàm thông dụng:

- Thuộc tính Length: cho biết số phần tử của mảng.
- Phương thức sort: Array.Sort (a);
- Phương thức Array.Reverse: Đảo ngược các phần tử của mảng.

Ví dụ 1.14: Nhập mảng a gồm n phần tử nguyên, xuất mảng vừa nhập. Sắp xếp mảng và in mảng sau khi đã sắp xếp.

```
static int[] nhap(int[] a)
{
    Console.Write("Nhập tổng số phần tử: ");
    int n = int.Parse(Console.ReadLine());
    a = new int[n];
    Console.WriteLine("Nhập từng phần tử: ");
    for (int i = 0; i < n; ++i)
        a[i] = int.Parse(Console.ReadLine());
    return a;
}
static void xuat(int[] a)
{
    for (int i = 0; i < a.Length; ++i)
        Console.Write(a[i] + " ");
}
static void sapxep(int[] a)
{
    Array.Sort(a);
    xuat(a);
}
```

Viết code sau vào hàm Main:

```
int[] a = null;
Console.Write("Nhập mảng: ");
a = nhap(a);
Console.WriteLine("Mảng chưa sắp xếp: ");
xuat(a);
Console.WriteLine();
Console.WriteLine("Sau khi sắp xếp: ");
sapxep(a);
```

Kết quả:

```

file:///E:/cntt/2012_2013/soan_giaotinh/code/ConsoleApp/ConsoleApp...
Nhập mảng: Nhập tổng số phần tử: 5
Nhập từng phần tử:
6
2
5
1
9
Mảng chưa sắp xếp:
6 2 5 1 9
Sau khi sắp xếp:
1 2 5 6 9 -

```

Hình 1.16 Kết quả ví dụ Mảng

1.3.6. Cấu trúc chuỗi

Bảng 1.10. Một số thuộc tính và phương thức cơ bản sau:

Phương thức	Mục đích
Length	Chiều dài chuỗi
IndexOf (s1)	Vị trí xuất hiện đầu tiên của chuỗi s1 trong chuỗi ban đầu
LastIndexOf (s1)	Giống IndexOf, nhưng tìm lần xuất hiện cuối cùng
Replace (s1, s2)	Thay thế chuỗi s1 thành s2 trong chuỗi ban đầu
Split (<ký tự ngắt>)	Tách chuỗi thành mảng chuỗi bởi <ký tự ngắt>
Substring (vt, l)	Trả về chuỗi con bắt đầu ở một vị trí vt và chiều dài l.
ToLower ()	Chuyển chuỗi thành chữ thường
ToUpper ()	Chuyển chuỗi thành chữ in
Trim ()	Xóa khoảng trắng ở đầu và cuối chuỗi

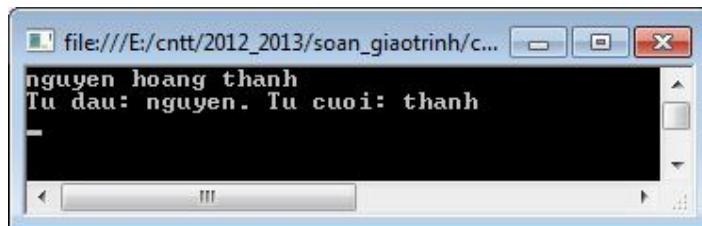
Ví dụ 15: Hiển thị từ đầu tiên và cuối cùng của 1 chuỗi.

```

string s = Console.ReadLine();
string[] s1 = s.Split(' ');
if (s1.Length >= 2)
{
    string tudau = s1[0];
    string tucuoai = s1[s1.Length - 1];
    Console.WriteLine("Tu dau: " + tudau + ". Tu cuoi: " +
tucuoai);
}

```

Kết quả



Hình 1.17 Kết quả ví dụ Chuỗi

1.4. Xử lý lỗi ngoại lệ

1.4.1. Giới thiệu

Ngôn ngữ C# cũng giống như bất cứ ngôn ngữ hướng đối tượng khác, cho phép xử lý những lỗi và các điều kiện không bình thường với những ngoại lệ. Ngoại lệ là một đối tượng đóng gói những thông tin về sự cố của một chương trình không bình thường.

Một điều quan trọng để phân chia giữa bug lỗi và ngoại lệ. Một bug là một lỗi lập trình có thể được sửa chữa trước khi mã nguồn được chuyển giao. Những ngoại lệ thì không được bảo vệ và tương phản với những bug. Mặc dù một bug có thể là nguyên nhân sinh ra ngoại lệ, chúng ta cũng không dựa vào những ngoại lệ để xử lý những bug trong chương trình, tốt hơn là chúng ta nên sửa chữa những bug này.

Khi một chương trình gặp một tình huống ngoại lệ, nó sẽ tạo một ngoại lệ. Khi một ngoại lệ được tạo ra, việc thực thi của các chức năng hiện hành sẽ bị treo cho đến khi nào việc xử lý ngoại lệ tương ứng được tìm thấy.

Điều này có nghĩa rằng nếu chức năng hoạt động hiện hành không thực hiện việc xử lý ngoại lệ, thì chức năng này sẽ bị chấm dứt và hàm gọi sẽ nhận sự thay đổi đến việc xử lý ngoại lệ. Nếu hàm gọi này không thực hiện việc xử lý ngoại lệ, ngoại lệ sẽ được xử lý sớm bởi CLR, điều này dẫn đến chương trình của chúng ta sẽ kết thúc.

1.4.2. Đón bắt lỗi ngoại lệ

Cú pháp:

```

try
{
    <câu lệnh>
}
catch ([<kiểu ngoại lệ> <tên biến>])
{
    <câu lệnh xử lý ngoại lệ>
}

```

Ví dụ 1.16: thực hiện phép chia số nguyên a cho số nguyên b. Ngoại lệ phát sinh khi b = 0, chương trình sẽ đón bắt ngoại lệ này.

```

int a, b;
a = int.Parse(Console.ReadLine());
b = int.Parse(Console.ReadLine());
try
{
    Console.WriteLine("Thuong = " + a / b);
}
catch (Exception e)
{
    Console.WriteLine("Bi loi, mau so phai khac 0");
}

```

Kết quả khi nhập b bằng 0:



Hình 1.18 Kết quả ví dụ try catch

1.4.3. Xử lý ngoại lệ phát sinh

Khi chương trình sử dụng khối lệnh try và lỗi phát sinh, quá trình đang thực thi của những lệnh trong khối try sẽ hủy bỏ để chuyển qua khối lệnh catch. Các thao tác xử lý tài nguyên có thể đang dở dang. Ta có thể sử dụng khối biểu finally để thực hiện dọn dẹp tài nguyên (như đóng kết nối cơ sở dữ liệu, đóng file, xóa file tạm, ...). Cấu trúc như sau.

```

try
{
    <Các câu lệnh>
}
catch ([<kiểu dữ liệu> <tên biến>]) { <câu lệnh xử lý ngoại lệ>}
finally
{
    <Các câu lệnh dọn dẹp tài nguyên>
}

```

Ví dụ minh họa phần này sẽ được trình bày ở chương 3: tập tin và thư mục.

BÀI TẬP CHƯƠNG I

TỔNG QUAN - ĐIỀU KIỆN

- Viết chương trình nhập vào số nguyên, kiểm tra số đã nhập là số âm hay số dương
- Viết chương trình nhập vào 2 số a, b. Tìm số lớn nhất giữa 2 số.
- Viết chương trình nhập vào 2 số nguyên, so sánh 2 giá trị vừa nhập vào (“Bằng nhau, nhỏ hơn, lớn hơn”).
- Viết chương trình nhập vào đơn giá 1 mặt hàng, và số lượng bán của mặt hàng. Tính tiền khách phải trả, với thông tin như sau:

- Thành tiền: đơn giá * số lượng
- Giảm giá: Nếu thành tiền > 100, thì giảm 3% thành tiền, ngược lại không giảm
- Tổng tiền phải trả: thành tiền – giảm giá.

- Viết chương trình tính tiền điện sử dụng trong tháng:

- Từ 1 – 100KW: 5\$
- Từ 101 – 150KW: 7\$
- Từ 151 – 200KW: 10\$
- Từ 201 – 300KW: 15\$
- Từ 300KW trở lên: 20\$

VÒNG LẶP FOR

- Viết chương trình tính tổng: $1 + 2 + 3 + 4 + 5 + \dots + 20$
- Viết chương trình tính tích: $1*2*3*4*5*\dots*n$, trong đó n nhập từ phím.
- Viết chương trình tính tổng: $1*2 + 2*3 + 3*4 + 4*5 + \dots + n(n+1)$.
- Viết chương trình tính tổng: $\frac{1}{1.2.3} + \frac{1}{2.3.4} + \frac{1}{3.4.5} + \dots + \frac{1}{n(n+1)(n+2)}$
- Viết chương trình in bảng cửu chương 1 số
- Viết chương trình in bảng cửu chương từ 1 -> 9 theo hàng ngang
- In tam giác cân đặc, rỗng hình ngôi sao.
- Viết chương trình vẽ hình chữ nhật có kích thước d x r, trong đó d là chiều dài, và r

là chiều rộng được nhập từ phím.

* * * * *

* * * * *

* * * * *

VÒNG LẶP WHILE

1. Tính n!
2. Dùng vòng lặp While làm lại các bài tập vòng lặp For ở phần trên.
3. Viết chương trình nhập vào 2 số nguyên, tìm USCLN, BSCNN.
4. Xuất dãy Fibonaci < n
5. Tìm ước chung lớn nhất của hai số nguyên a và b.
6. Nhập dãy số nguyên (nhập số 0 thì dừng)
 - a. Tìm tổng dãy số đó
 - b. Tìm max của dãy số đó
 - c. Tìm min của dãy số đó
7. Nhập một số. In số đảo ngược của chúng. Ví dụ: 123 xuất 321

PHƯƠNG THỨC

1. Kiểm tra n có phải là số nguyên tố hay không?
2. Liệt kê các số nguyên tố < n
3. Kiểm tra có phải là số hoàn hảo không?
4. Liệt kê các số hoàn hảo < n
5. Kiểm tra n có phải là số chính phương không?
6. Viết chương trình tính tổng: $1! + 2! + 3! + \dots + n!$, trong đó n nhập từ phím.
7. Viết chương trình hoán đổi giá trị giữa 2 biến số nguyên.
8. Viết chương trình đếm số nguyên tố nhỏ hơn n (n nhập từ phím) và in các số nguyên tố đó ra màn hình.

MẢNG 1 CHIỀU

1. Nhập mảng a gồm n phần tử.
2. Xuất mảng a.
3. Liệt kê các phần tử dương.
4. Liệt kê các phần tử lẻ ở vị trí chẵn.
5. Liệt kê các số nguyên tố trong mảng.
6. Tìm phần tử âm đầu tiên trong mảng.
7. Tìm phần tử chính phương đầu tiên trong mảng.
8. Tìm max, min, max âm, min âm, max dương, min dương.
9. Có bao nhiêu số chẵn/lẻ, âm/dương, nguyên tố trong mảng.

10. Tính tổng các phần tử trong mảng
11. Thêm phần tử x vào vị trí vt.
12. Thêm x vào vị trí đầu tiên.
13. Xóa phần tử đầu tiên = x.
14. Xóa tất cả phần tử = x.

MẢNG 2 CHIỀU

1. Viết chương trình nhập các giá trị cho mảng 2 chiều, sau đó xuất mảng vừa nhập.
2. Viết hàm tính tổng các giá trị mảng.
3. Viết hàm tìm giá trị lớn nhất trong mảng.
4. Viết hàm tìm số lẻ nhỏ nhất trong mảng.
5. Viết hàm tìm vị trí của giá trị x trong mảng, x nhập từ phím.
6. Viết hàm đếm số lần phần tử x xuất hiện trong mảng.
7. Viết hàm đếm số nguyên tố trong mảng.
8. Viết hàm xuất các phần tử ở dòng i trong mảng, và tính tổng trên dòng i, i nhập từ phím.
9. Viết hàm tìm giá trị nhỏ nhất trên cột j, j nhập từ phím.
10. Nhập và in ma trận vuông cấp n.
 - a. Liệt đường chéo chính, phụ.
 - b. Liệt kê tam giác trên, dưới
 - c. Tổng đường chéo chính, phụ.
 - d. Tổng tam giác trên, dưới.

CHUỖI

1. Viết chương trình nhập tên sinh viên, chuyển tên thành kí tự in hoa, sau đó xuất tên sinh viên và chiều dài chuỗi tên sinh viên vừa nhập.
2. Viết chương trình nhập vào họ tên đầy đủ của sinh viên, sau đó xuất ra chuỗi tên họ sinh viên.
3. Nhập chuỗi s (là họ tên):
 - a. S có phải là chuỗi đối xứng không?
 - b. Xóa khoảng trắng thừa trong chuỗi s
 - c. Có bao nhiêu từ trong chuỗi s
 - d. Lấy ra họ
 - e. Lấy ra tên
 - f. Lấy ra họ lót (nếu có)
 - g. Đổi ký tự đầu từ thành chữ hoa
 - h. Đổi ký tự thường thành chuỗi hoa và ngược lại

i. Có bao nhiêu nguyên âm, phụ âm, khoảng trắng trong chuỗi s.

CHƯƠNG 2. WINDOWS FORM

Mục tiêu:

Sau khi học xong chương này, sinh viên có khả năng:

- Phân tích, phân biệt công dụng của các control.
- Áp dụng các Windows Control để xây dựng các ứng dụng Window Form.

2.1. Giới thiệu

Windows Form là tập hợp những lớp chứa giao diện đồ họa người dùng (GUI) của ứng dụng desktop cổ điển. Trước đây, mỗi ngôn ngữ lập trình đều có cách khác nhau để tạo ra Windows, Texboxes, Buttons, ... Chức năng này đã được đưa vào trong thư viện lớp của .Net Framework, trở thành những thành phần của namespace System.Windows.Forms.

Trong chương này, form được xem như bộ phận trung tâm của ứng dụng desktop. Chúng ta sẽ xem xét Form được khởi tạo như thế nào và nó phát sinh sự kiện ra sao. Bên cạnh đó, chúng ta sẽ khảo sát những Windows Controls để xây dựng nên những ứng dụng Windows hiệu quả.

2.2. Form

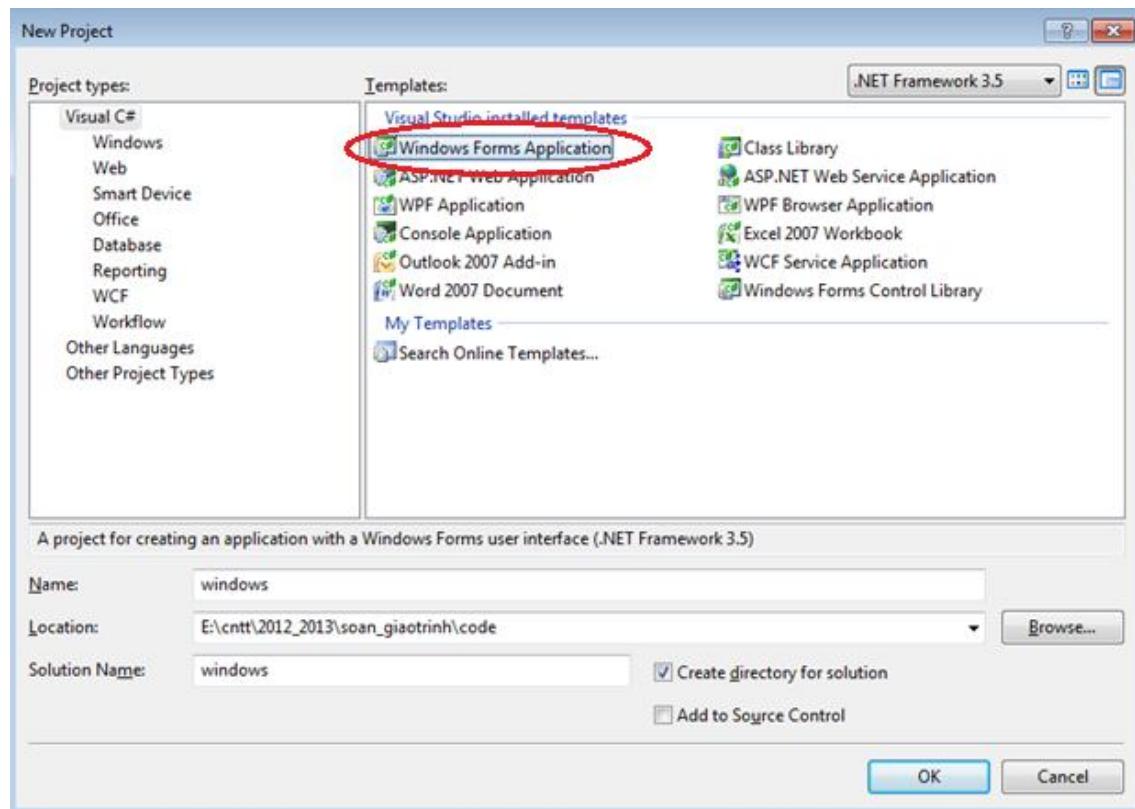
2.2.1. Tạo Form

Cách dễ nhất để tạo một Form là sử dụng công cụ Windows Forms Designer trong Visual Studio.Net. Công cụ này cho phép chúng ta sử dụng những công cụ trực quan để trình bày form, code tạo form sẽ được tự động tạo ra. Nếu chúng ta không có Visual Studio.Net, chúng ta có thể viết code C#.Net trực tiếp và không cần dùng Designer.

Một form được định nghĩa bằng cách dẫn xuất một lớp từ lớp Form (được định nghĩa trong System.Windows.Forms). Lớp form này chứa đựng cách thức để trình bày một form rỗng bao gồm title bar và một số thành phần khác được kế thừa từ Windows form. Chúng ta có thể thêm vào lớp mới này những thành phần cũng như ta có thể override những thành phần từ lớp cơ sở Form.

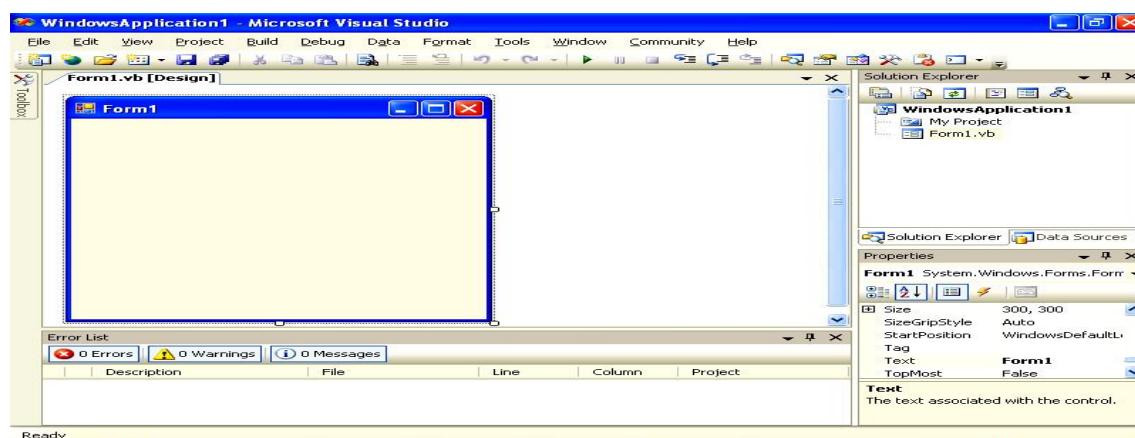
Cách tạo form bằng C#.Net.

- Chọn File → New Project. Project dialog xuất hiện



Hình 2.1. Tạo giao diện Window Form

- Chọn Visual C# trong Project types.
- Chọn Windows Application trong khung Templates.
- Đặt tên trong ô name.
- Chọn OK, Visual Studio.Net sẽ tạo một Project mới với một form và trình bày trong cửa sổ thiết kế.



Hình 2.2. Giao diện cửa sổ đồ họa cơ bản

Ta khảo sát một vài phương thức, thuộc tính quan trọng của lớp Form1 trong đoạn code trên.

- Phương thức New: Khởi tạo các thành phần của Form.
- Phương thức Dispose: là nơi mà tất cả các đối tượng giải phóng tài nguyên.

Trong trường hợp này, nó sẽ gọi phương thức Dispose của lớp cơ sở để giải phóng tài nguyên mà nó chiếm giữ, sau đó sẽ gọi phương thức Dispose của Components. Sau khi phương thức Dispose được phát sinh, Designer sẽ không quan tâm đến nó nữa.

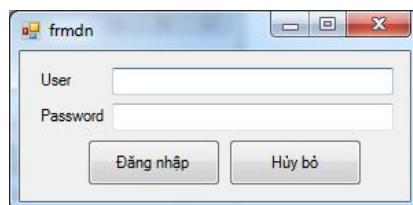
- Trường (field) Components: Trường Components là một đối tượng kiểu Icontainer (được định nghĩa trong namespace System.ComponentModel). Code được phát sinh bởi Designer sử dụng trường Components này để quản lý sự kết thúc của các thành phần (component) được thêm vào Form.
- Phương thức InitializeComponent: Chúng ta không nên thêm code vào trong phương thức này, Windows Form Designer sẽ tự động thêm code vào khi cần thiết. Khi các control được thêm vào form (bằng cách dùng Designer), code sẽ được thêm vào phương thức này để thiết lập các control và khởi tạo các thuộc tính ban đầu cho chúng.

2.2.2. Một số thuộc tính của Form.

Để hiển thị cửa sổ thuộc tính của Form, ta click chuột phải vào form, chọn Property. Sau đây là một vài thuộc tính thông dụng của form.

- Backcolor: Thiết lập thuộc tính màu nền của form.
- ControlBox: Thiết lập thuộc tính cho các nút Min, Max, Close của form. Nếu ta chọn thuộc tính này là false thì Form sẽ không có các nút trên. Sự chọn lựa này thường được áp dụng cho các form đăng nhập.
- Font: Thiết lập thuộc tính font cho form. Thuộc tính font của form sẽ là thuộc tính font mặc định cho tất cả các control được thêm vào form sau này.
- MaximizeBox: Thuộc tính cho nút Max. Thiết lập giá trị cho thuộc tính này là True thì form sẽ có nút Max.
- MinimizeBox: Thuộc tính cho nút Min. Thiết lập giá trị cho thuộc tính này là True thì form sẽ có nút Min.
- Icon: Thiết lập icon cho form.
- Text: Thiết lập tiêu đề cho form.

Ví dụ 2.1: sau hiển thị form đăng nhập mà ta thường thấy. Tạo mới một form, cho thuộc tính ControlBox là false. Thêm vào form hai Label, hai TextBox và hai Button (như hình). Biên dịch và chạy chương trình sẽ cho kết quả như hình sau:



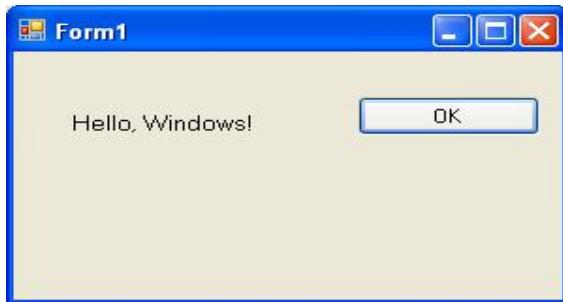
Hình 2.3. Form đăng nhập

2.2.3. Thêm control vào form

Control có thể được đưa vào form từ hộp công cụ (toolbox) của Visual Studio.Net. Để hiển thị hộp công cụ, ta chọn View → Toolbox từ menu của Visual Studio.Net. Sau khi hộp công cụ xuất hiện, ta chỉ cần click vào control nào cần đưa vào form, sau đó ta có thể dùng chuột để kéo control đến vị trí thích hợp trên form.

Tương tự như khi tạo form, khi ta thêm control vào form, Designer tự động phát sinh code tương ứng để tạo ra các control.

Ví dụ 2.2. ta thêm vào form control Label và control Button như hình sau.



Hình 2.4. Thêm control vào form

2.2.4. Xử lý các sự kiện cho các control

Ứng dụng Hello, Windows! trên được thiết kế với một button OK nhưng ứng dụng chưa hồi đáp lại sự kiện Click button. Để thêm bộ xử lý sự kiện Click cho button OK, ta double click vào button trong Windows Forms Designer. Designer sẽ chuyển sang màn hình code của form và thêm vào một hàm **void** để xử lý sự kiện Click. Một cách khác, chúng ta có thể chọn vào biểu tượng  trên thanh Property để chọn sự kiện cần

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

xử lý. Hàm được Designer phát sinh như sau:

Phần thân của thủ tục này sẽ được ta thêm vào. Giả sử như ta muốn khi người dùng click vào button OK thì sẽ thoát ứng dụng, ta làm như sau.

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

2.2.5. Xử lý sự kiện của form

Chúng ta sẽ khảo sát một vài sự kiện chính của lớp form, bao gồm sự mô tả về ý nghĩa và cú pháp của từng sự kiện.

- Sự kiện Activated: Xuất hiện khi form được kích hoạt.

- Sự kiện FormClosed: Xuất hiện khi form đã được đóng.
- Sự kiện FormClosing: Xuất hiện khi form chuẩn bị đóng .
- Sự kiện Load: Xuất hiện khi form được tải xuống.

Ví dụ 2.3: sau minh họa sự kiện Load của form. Khi form được tải xuống, nó sẽ hiển thị thông điệp “Form is being loaded” (như hình) trước khi form hình thành.



Hình 2.5. Form Load

```
private void frmRichTextbox_4_Load(object sender, EventArgs e)
{
    MessageBox.Show("Form is being loaded");
}
```

2.3. Các control thông dụng

2.3.1. Button

Control Button là một trong những control được sử dụng nhiều nhất trong các ứng dụng Windows. Sự kiện Click là sự kiện mà chúng ta thường xuyên sử dụng khi lập trình với control này.

Lớp Button kế thừa hai thuộc tính quan trọng từ lớp ButtonBase, đó là thuộc tính Flatstyle và thuộc tính Image. Thuộc tính Flatstyle có bốn tùy chọn: Flat, popup, Standard (mặc định) và System. Hình minh họa sau sẽ thể hiện bốn tùy chọn trên.



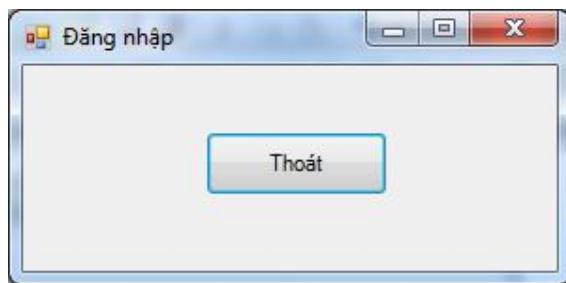
Hình 2.6.Thuộc tính của Button

Thuộc tính Image cho phép chúng nhúng hình ảnh vào trong button. Đoạn code sau trình bày cú pháp thiết lập thuộc tính Image cho button.

```
Button1.Image = New System.Drawing.Bitmap (filepath).
```

Ví dụ 2.4: Chương trình sau bao gồm một button (như hình). Sau khi biên dịch và thực thi chương trình, ta click chuột vào button thì sẽ chấm dứt chương trình. Ta gõ

đoạn code sau vào sự kiện Click của Button.



Hình 2.7. Form đăng nhập

```
private void button1_Click(object sender, EventArgs e)
{
    this.Close();
}
```

2.3.2. Label

Trong form, điều khiển Label cho phép ta đặt văn bản miêu tả. Label thường đặt trước các điều khiển khác với văn bản mô tả cách sử dụng điều khiển. Thường người lập trình sẽ kéo và thả Label lên form, gán văn bản mô tả cho nó thông qua thuộc tính text. Ta có thể sử dụng một số thuộc tính khác để thay đổi màu nền (BackColor), màu chữ (ForeColor), khung viền (BorderStyle), font.

Ngoài chức năng hiển thị văn bản, Label còn có thể được sử dụng để hiển thị hình ảnh. Ta sử dụng thuộc tính Image của Label để hiển thị hình ảnh mong muốn. Ví dụ sau sẽ đưa một hình ảnh vào control Label thông qua thuộc tính. Chương trình hiển thị một form chứa hai control Label và một Button, mỗi Label chứa hình ảnh kích thước cố định (như hình). Khi nhấn Button sẽ thoát chương trình.

Để hiển thị hình ảnh trong Label, trước hết ta tạo ra một đối tượng Image, sau đó tải ảnh tương ứng vào đối tượng Image thông qua phương thứcFromFile của lớp Image. Tiếp theo ta thực hiện gán thuộc tính Image của Label cho đối tượng Image ở trên.

Ví dụ 2.5: minh họa label hiển thị hình ảnh



Hình 2.8.Hiển thị ảnh trong Label

```

Image1 = Image.FromFile("lab.jpg") ;
Image2 = Image.FromFile("dal.jpg") ;
//Gán hai đối tượng Image trên vào thuộc tính Image của hai
//Label.
Label1.Image = Image1;
Label2.Image = Image2;

```

2.3.3. Textbox

Textbox thường được sử dụng để nhập thông tin người dùng ở dạng văn bản. Trong control Textbox chương trình có thể cho phép chỉnh sửa dữ liệu (hoặc không), cũng như cho phép cắt, dán, copy như những chương trình soạn thảo cơ bản trong Windows.

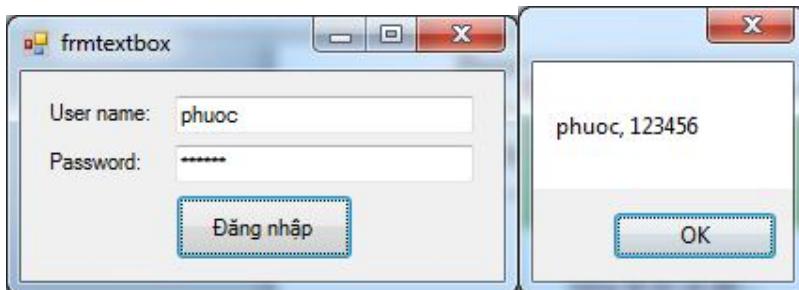
Thuộc tính:

- **Multiline:** Thuộc tính này cho phép chúng ta thiết lập là true hoặc false. Nếu là true textbox sẽ chứa được nhiều dòng, ngược lại textbox chỉ có một dòng. Mặc định của thuộc tính này là false.
- **Passwordchar:** Định nghĩa kí tự mặt nạ hiển thị các kí tự do người dùng nhập vào. Thuộc tính này chỉ áp dụng cho textbox một dòng (Multiline = false) và nó chỉ ảnh hưởng đến hình thức trình bày mà không ảnh hưởng đến giá trị nhập vào textbox.
- **ScrollBars:** Thuộc tính này chỉ áp dụng cho textbox dạng Multiline, nó xác định xem thanh cuộn có xuất hiện trên textbox hay không. C#.Net hỗ trợ bốn tùy chọn cho thuộc tính này: None (mặc định, không có thanh cuộn), Horizontal (thanh cuộn ngang), Vertical (thanh cuộn dọc) và Both (có cả thanh cuộn ngang và dọc).
- **TextAlign:** Thuộc tính này xác định cách thức canh lề cho textbox. C#.Net hỗ trợ ba tùy chọn cho thuộc tính này: Center (canh giữa), Left (canh trái) và Right (canh phải).

Sự kiện:

- Textchange: Sự kiện này xảy ra khi có sự thay đổi nội dung trong textbox.
- Keydown: Sự kiện này xả khi người dùng nhấn phím trên textbox.
- Keyup: Sự kiện này xảy ra khi người dùng nhả phím trên textbox.

Ví dụ 2.6: sử dụng các control textbox, Label, Button để tạo màn hình đăng nhập. Textbox1 dùng để nhập tên người dùng, textbox2 dùng để nhập Password (gán thuộc tính Passwordchar của textbox2 là kí tự tùy ý). Khi ta nhấn nút Đăng nhập hoặc nhấn phím Enter trong textbox Password, chương trình sẽ hiển thị nội dung của textbox1 và textbox2.



Hình 2.9. Minh họa Textbox

Soucre code: gọi hàm sau trong sự kiện click của Button Đăng nhập:

```
MessageBox.Show(textBox1.Text + " , " + textBox2.Text);
```

Hoặc xử lý trong sự kiện **keyup** của Textbox Passwor:

```
private void textBox2_KeyUp(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
        MessageBox.Show(textBox1.Text + " , " +
textBox2.Text);
}
```

2.3.4. Richtextbox

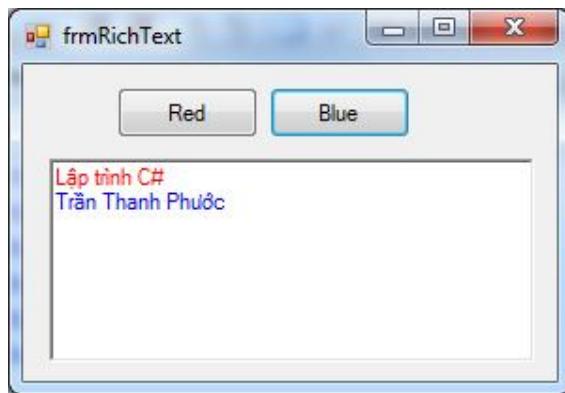
Với Textbox chúng ta chỉ hiển thị văn bản ở dạng đơn (cùng một font chữ, màu sắc, cỡ chữ....). Để trình diễn dữ liệu tốt hơn, .Net cung cấp cho ta control mở rộng RichTextBox có thể hiển thị tài liệu được định dạng như một nội dung Word. Ta có thể sử dụng các font chữ khác nhau cho nhiều đoạn văn bản chứa bên trong, màu sắc khác nhau, cỡ chữ khác nhau,... Chương trình Wordpad của Windows là một thể hiện của RichTextBox, còn chương trình NotePad là thể hiện của Textbox.

Thuộc tính:

- Font: Cho biết kiểu font, kích cỡ, kiểu in đậm hoặc nghiêng... Khác với TextBox, RichTextBox có thể định dạng font chữ cho từng khối (bôi đen) văn bản.
- ForeColor: Cho biết màu chữ của RichTextBox.

- Multiline: Cho biết RichTextBox có nhiều dòng hay không. Nếu thuộc tính này là True thì RichTextBox có nhiều dòng, ngược lại chỉ có một dòng. Mặc định của thuộc tính này là True.
- ScrollBars: Thuộc tính này cho phép RichTextBox có thanh cuộn ngang hay dọc hay không.
- Text: Cho biết nội dung của RichTextBox.
- SelectionFont: Cho biết Font của phần văn bản đang được chọn.
- SelectionColor: Cho biết màu của phần văn bản đang được chọn.

Ví dụ 2.7: sau sẽ minh họa cách thay đổi màu của phần văn bản được chọn trong RichTextBox. Form gồm có hai Button là Red và Blue, một RichTextBox (như hình). Biên dịch và thực thi chương trình, ta gõ một đoạn văn bản vào RichTextBox, sau đó chọn khói rồi nhấn nút Red (văn bản chuyển thành màu đỏ) hoặc Blue (văn bản chuyển thành màu xanh).



Hình 2.10. Minh họa RichTextBox

Source code:

```
private void bred_Click(object sender, EventArgs e)
{
    rtf.SelectionColor = Color.Red;
}
private void bblue_Click(object sender, EventArgs e)
{
    rtf.SelectionColor = Color.Blue;
}
```

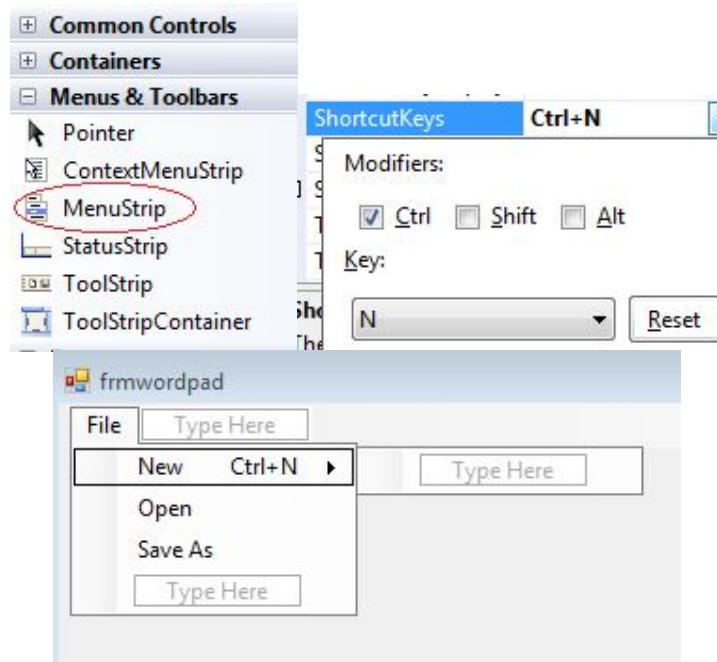
2.3.5. Menu

Menu là một thành phần rất thông dụng trong các ứng dụng Windows cho phép người dùng có thể chọn một thao tác xử lý đặc biệt, chương trình có thể sử dụng Control Menu để xây dựng trình đơn.

Để đặt một menu lên trên một form trong Visual Studio, ta có thể kéo và thả control Menu lên form. Visual Studio sẽ hiển thị một ô gắn nhãn “Type Here” trên đỉnh của form. Một Menu lại có nhiều MenuItem. Để thêm vào các MenuItem đơn giản ta chỉ

cần gõ vào tên mục chọn cho menu. Visual Studio sẽ tự mở rộng menu để cho ta thêm mục chọn mới bên dưới mục vừa tạo. Để xử lý một mục chọn trên menu hoặc MenuItem, chương trình phải cung cấp một bộ xử lý sự kiện cho mỗi mục chọn. Ta chỉ chọn một mục menu hoặc MenuItem, chương trình sẽ tự động phát sinh sự kiện tương ứng.

Ví dụ 2.8: sau sẽ tạo ra menu cho form chương trình tương tự Notepad, Wordpad. MenuItem New có shortcut key là Ctrl + N

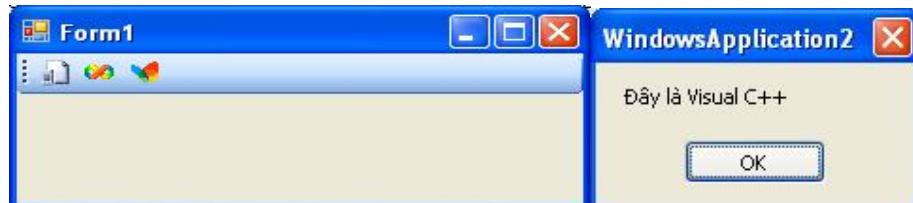


Hình 2.11. Minh họa Menu

2.3.6. Toolbar

Trong môi trường Windows, người dùng đã quen sử dụng thanh công cụ trong một cửa sổ để thực hiện những thao tác nhanh. Ví dụ, trong MS Word ta có thể sử dụng nút thanh công cụ Save File để cất giữ file hoặc nút Print để in tài liệu. Để sử dụng thanh công cụ, ta đặt control ToolStrip lên form sau đó lèn lượt tạo ra các nút nhấn là đối tượng ToolStripButton bằng cách click chuột vào một nút có sẵn trên ToolStrip.

Ví dụ 2.9: minh họa cách sử dụngToolBar. Kéo chọn control ToolStrip lên form, sau đó tạo ra ba button trên toolbar. Mỗi button ta chọn một hình riêng biệt bằng cách click chọn thuộc tính Image trên hộp toolbar (như hình).



Hình 2.12. Minh họa Toolbar

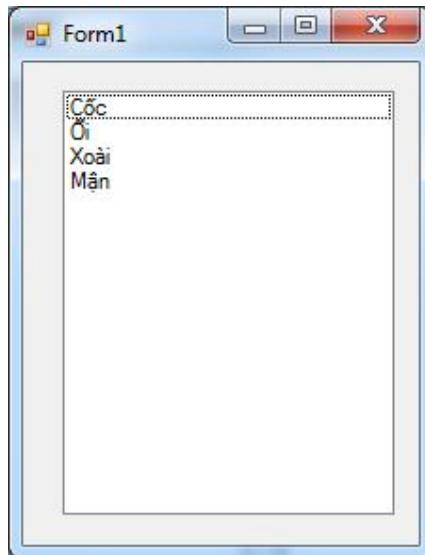
2.3.7. ContextMenu

ContextMenu hay còn gọi là menu ngữ cảnh, là loại menu xuất hiện khi chúng ta click chuột phải trong các ứng dụng windows. Các bước tạo menu ngữ cảnh như sau:

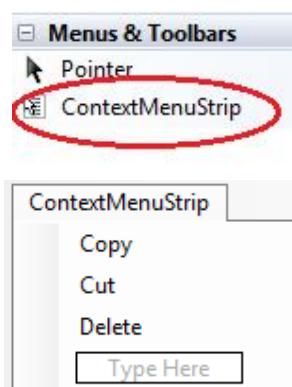
- Tạo contextmenu: Kéo thả từ thanh Toolbox
- Thiết lập các menu con cho contextmenu
- Gắn thuộc tính contextmenustrip của control cho contextmenu vừa tạo.

Ví dụ 2.10: minh họa cách tạo contextmenu cho control Listbox.

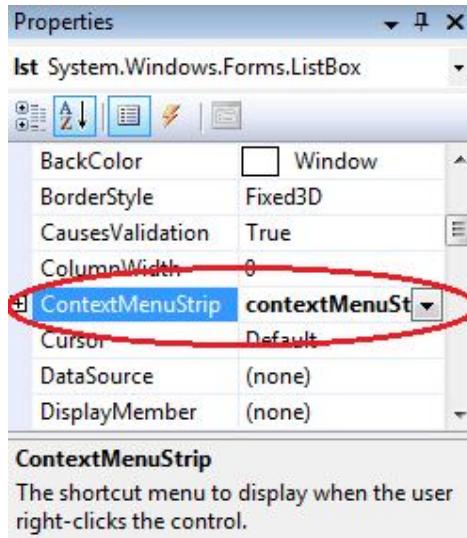
- Tạo Form chứa Listbox với các items như sau:



- Tạo contextmenu



- Gắn contextmenu vào listbox



Hình 2.13 Sử dụng ContextMenu

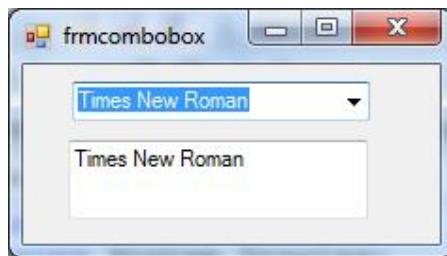
2.3.8. Combobox

.Net cung cấp cho ta điều khiển Combobox cho phép người dùng chọn một mục từ danh sách. Để sử dụng combobox, tương tự như các control khác, đơn giản ta chỉ cần kéo và thả control Combobox lên form.

Hai thuộc tính quan trọng của ComboBox là Items và Text. Thuộc tính Text cho ta biết phần tử mà ta vừa chọn. Thuộc tính Items trả về đối tượng ComboBox.ObjectCollection. Trong đối tượng này có các phương thức quan trọng như.

- Add (item as Object): Thêm một phần tử item vào trong ComboBox.
- Clear: Xóa tất cả các phần tử trong ComboBox.
- Remove (value as Object): Xóa phần tử value trong ComboBox.
- RemoveAt (index as Integer): Xóa một phần tử trong ComboBox tại vị trí index.

Ví dụ 2.11: minh họa các sử dụng control ComboBox. Thêm control ComboBox (cbfont) và TextBox vào form (như hình). Trong sự kiện load của form, ta thêm vào ComboBox các font hệ thống. Khi ta click chọn phần tử nào đó của ComboBox, thì TextBox (txt) sẽ hiển thị font đó.



Hình 2.14. Minh họa ComboBox

Souce code:

```

private void frmcombobox_Load(object sender, EventArgs e)
{
    for (int i = 0; i < FontFamily.Families.Length; ++i)
        cbfont.Items.Add(FontFamily.Families[i].Name);
}
private void cbfont_SelectedIndexChanged(object sender,
EventArgs e)
{
    txt.Text = cbfont.Text;
}

```

2.3.9. Listbox

Control Listbox cho phép chúng ta thể hiện một danh sách để người dùng lựa chọn. Không giống như ComboBox, ListBox hiển thị danh sách các mục cho ta chọn trực tiếp.

Chúng ta khảo sát một vài thuộc tính cơ bản của ListBox.

- MultiColumn: Xác định ListBox có một hay nhiều cột. Nếu thuộc tính này bằng True thì ListBox có nhiều cột, ngược lại chỉ có một cột.
- Items: Đây là thuộc tính quan trọng nhất của ListBox, kết quả trả về là lớp đối tượng ListBox.ObjectCollection, bao gồm tập hợp những phần tử của ListBox. Chúng ta có thể thêm một phần tử vào ListBox bằng phương thức Add, hoặc dùng phương thức AddRange của lớp ListBox.ObjectCollection để thêm một tập hợp các phần tử vào trong ListBox. Đoạn code sau sẽ dùng phương thức AddRange để thêm các phần tử hoa quả vào trong ListBox1.

```

listBox1.Items.AddRange(New Object() {"apple", "avocado",
"banana", "carrot", "mandarin", "orange"})

```

- SelectedIndex: Cho ta biết chỉ số của phần tử được chọn trong ListBox. Nếu như có nhiều hơn một phần tử được chọn thì thuộc tính này sẽ trả về chỉ số của phần tử thấp nhất. Nếu không có phần tử nào được chọn thì thuộc tính này có giá trị là -1.
- SelectedItem: Đây là thuộc tính dạng Read-Only, trả về phần tử được chọn có kiểu là Object. Chúng ta cần phải chuyển đổi giá trị trả về sang kiểu thích hợp, thường là kiểu String. Nếu có nhiều hơn một phần tử được chọn, thì thuộc tính sẽ trả về phần tử với chỉ số thấp nhất.
- SelectedItems: Đây cũng là thuộc tính dạng Read-Only, nó trả về tất cả những phần tử được chọn từ ListBox và có kiểu là ListBox.SelectedItemsCollection. Lớp ListBox.SelectedItemsCollection có thuộc tính Count trả về tổng số phần tử được chọn trong tập hợp và thuộc tính Item cho biết phần tử được chọn. Đoạn code sau sẽ trình bày tất cả những phần tử được chọn của ListBox1.
- Text: Trả về tiêu đề của phần tử được chọn.

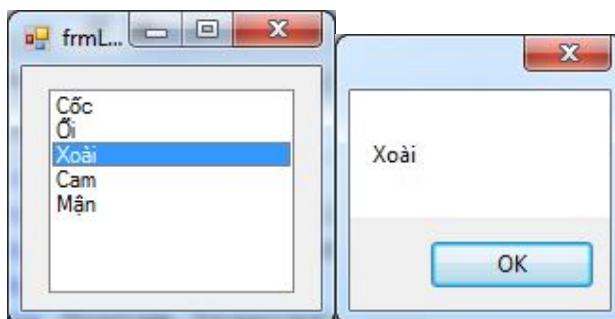
ListBox.ObjectCollection

Lớp này đại diện cho tất cả những phần tử của đối tượng ListBox. Thuộc tính Count của lớp này cho biết tổng số phần tử trong ListBox và thuộc tính Items trả về phần tử với chỉ số vị trí xác định. Đoạn code sau sẽ hiển thị tất cả những phần tử của đối tượng ListBox1.

Sau đây là một vài phương thức quan trọng của lớp này.

- Add: Thêm một phần tử vào trong ListBox. Cú pháp: `ListBox.ObjectCollection.Add(item)`
- AddRange: Thêm vào một hoặc nhiều phần tử vào trong ListBox. Cú pháp: `ListBox.ObjectCollection.AddRange (items())`
- Clear: Xóa ListBox, xóa tất cả các phần tử của ListBox. Cú pháp: `ListBox.ObjectCollection.Clear ()`
- Remove: Xóa phần tử được xem như là một tham số của phương thức này từ ListBox. Cú pháp: `ListBox.ObjectCollection.Remove(value)`, value là một đối tượng đại diện cho phần tử bị xóa từ trong tập hợp.
- RemoveAt: Xóa một phần tử tại vị trí có chỉ số xác định. Cú pháp: `ListBox.ObjectCollection.RemoveAt (index)`, với index là chỉ số vị trí của phần tử trong tập hợp.

Ví dụ 2.12: minh họa cách sử dụng Control ListBox (lst). Chương trình gồm một ListBox chứa danh sách các trái cây. Để thêm các phần tử vào LisstBox, ngoài cách dùng phương thức Add, ta có thể trực tiếp thêm vào thông qua thuộc tính Items trên cửa sổ thuộc tính. Khi ta click chọn một phần tử của ListBox, chương trình sẽ hiển thị tên của phần tử vừa được chọn.



Hình 2.15. Minh họa ListBox

Source code:

Chọn thuộc tính Items của listbox và gõ vào các trái cây trên, xử lý sự kiện SelectedIndexChanged

```

private void lst_SelectedIndexChanged(object sender,
EventArgs e)
{
    MessageBox.Show(lst.Text);
}

```

2.3.10. Radiobutton

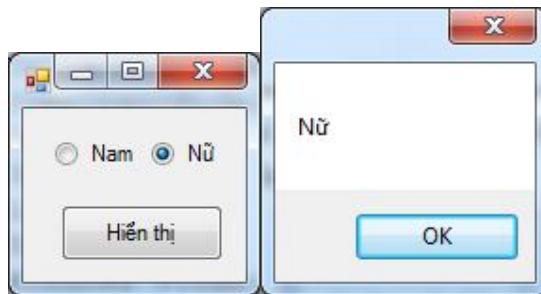
Nút Radio cho phép người dùng chọn một tùy chọn từ một tập hợp những tùy chọn. Ta chỉ có thể chọn được một trong danh sách các mục chọn. Thông thường chương trình ít khi trả lời sự kiện của control RadioButton. Thay vào đó, chương trình sẽ kiểm tra giá trị của chúng (thông qua thuộc tính Checked) khi bắt đầu thao tác xử lý.

```

if (RadioButton1.Checked)
{
    <Câu lệnh xử lý>;
}

```

Ví dụ 2.13: hiển thị hai radiobutton nam và nữ, khi chọn button Hiển thị sẽ hiển thị giới tính vừa chọn



Hình 2.16. Minh họa RadioButton

Source code:

```

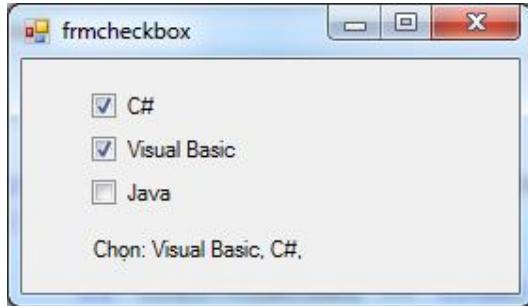
private void bht_Click(object sender, EventArgs e)
{
    if (rdnam.Checked == true)
        MessageBox.Show("Nam");
    else
        MessageBox.Show("Nữ");
}

```

2.3.11. Checkbox

Trong một form, control Checkbox cho phép người dùng chọn nhiều tùy chọn cùng lúc (ngược lại với RadioButton). Thuộc tính quan trọng nhất của CheckBox là Checked, nếu thuộc tính này là True thì CheckBox được chọn, ngược lại là không được chọn.

Ví dụ 2.14: minh họa cách dùng của CheckBox. Chương trình gồm ba CheckBox và một Label (như hình). Sau khi biên dịch và thực thi chương trình, khi CheckBox được chọn chương trình cập nhật Label tương ứng với CheckBox.



Hình 2.17. Minh họa CheckBox

Source code:

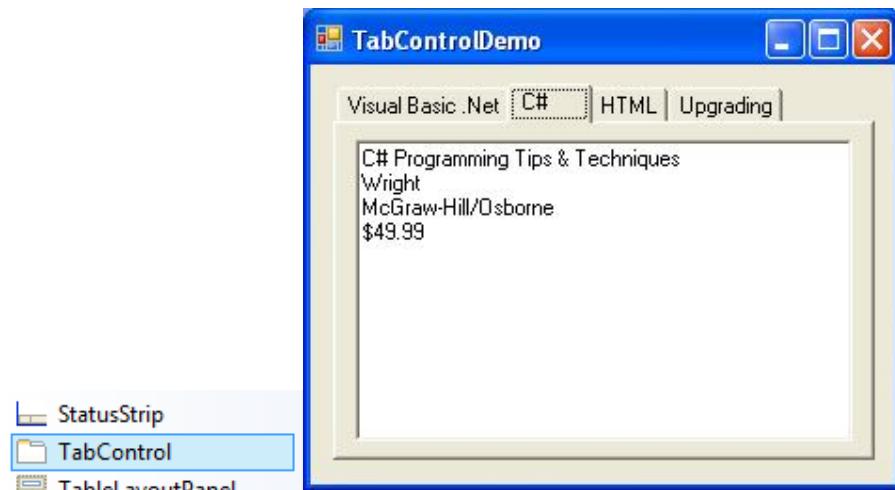
```
private void ck1_CheckedChanged(object sender, EventArgs e)
{
    string kq = "Chọn: ";
    if (ck1.Checked == true)
        kq = kq + ck1.Text + ", ";
    if (ck2.Checked == true)
        kq = kq + ck2.Text + ", ";
    if (ck3.Checked == true)
        kq = kq + ck3.Text;
    lb.Text = kq;
}
private void ck2_CheckedChanged(object sender, EventArgs e)
{
    string kq = "Chọn: ";
    if (ck2.Checked == true)
        kq = kq + ck2.Text + ", ";
    if (ck1.Checked == true)
        kq = kq + ck1.Text + ", ";
    if (ck3.Checked == true)
        kq = kq + ck3.Text;
    lb.Text = kq;
}
private void ck3_CheckedChanged(object sender, EventArgs e)
{
    string kq = "Chọn: ";
    if (ck3.Checked == true)
        kq = kq + ck3.Text + ", ";
    if (ck1.Checked == true)
        kq = kq + ck1.Text + ", ";
    if (ck2.Checked == true)
        kq = kq + ck2.Text;
    lb.Text = kq;
}
```

2.3.12. Tab

Tùy vào lượng thông tin mà chương trình cần hiển thị, nếu thông tin quá nhiều ta có thể tách thông tin hiển thị trên nhiều bảng Tab khác nhau. Tab là control tương các ngăn ta có thể dùng để tài liệu hoặc phân cách tài liệu trong những cặp hồ sơ. Mỗi một ngăn như thế ta gọi là một trang (page). Khi sử dụng control Tab ta có thể đặt nhiều control trên từng Page của Tab.

Để tạo Tab, đơn giản ta chỉ cần kéo thả control Tab lên form. Sau đó, nếu ta muốn thêm Page ta chỉ cần click chuột phải lên Tab rồi chọn Add Tab. Ta cũng có thể sử dụng thuộc tính TabPages để thêm các trang vào Tab.

Ví dụ 2.15: sử dụng một control Tab để hiển thị thông tin về nhiều tựa sách khác nhau. Mỗi khi ta chọn một Page, chương trình sẽ trình bày thông tin của tựa sách tương ứng.

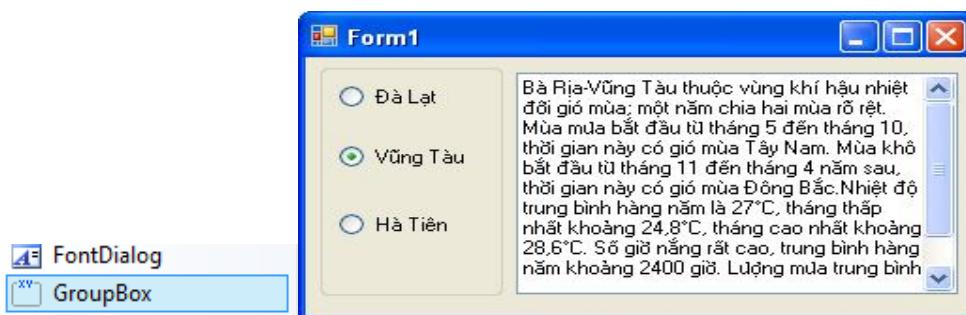


Hình 2.18. Minh họa CheckBox

2.3.13. GroupBox

Nếu form chứa hai hoặc nhiều nhóm lựa chọn (đặc biệt là lựa chọn RadioButton), ta nên đặt mỗi nhóm trong control GroupBox. Trong Visual Studio ta đơn giản kéo và thả control GroupBox lên trên form. Đặt tiêu đề (thuộc tính Text) cho GroupBox, sau đó đặt các control cùng nhóm vào trong GroupBox.

Ví dụ 2.16: minh họa của control RadioButton ở trên sử dụng GroupBox để nhóm các RadioButton.



Hình 2.19. Minh họa CheckBox

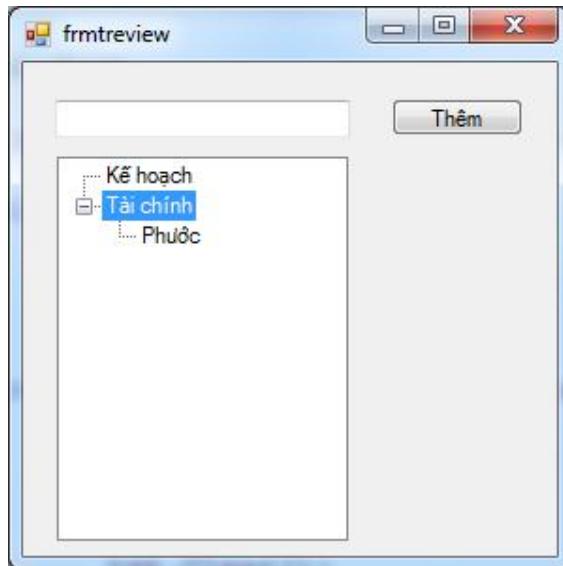
2.3.14. Treeview

Trong môi trường lập trình Windows, các ứng dụng như Windows Explorer sử dụng cấu trúc cây của control TreeView để trình diễn file và Window. Phụ thuộc vào thông tin mà chương trình cần hiển thị ta có thể dùng cấu trúc cây tương tự như Explorer thể hiện cấu trúc quan hệ theo cấp, cha/con. Nhánh bên trái của Explorer là TreeView, nhánh bên phải là ListView (nghiên cứu sau).

Thuộc tính quan trọng nhất của TreeView là Nodes. Kết quả trả về là lớp đối tượng ListView.ObjectCollection. Chúng ta có thể thêm một phần tử vào ListBox bằng phương thức Add. Cú pháp như sau:

Treeview1.Nodes.Add (new TreeNode (tên thư mục hoặc tên file))

Ví dụ 2.17: hiển thị cách sử dụng của control TreeView. Chương trình gồm một textbox (txt), một button thêm (bthem) và một Treeview (tr). Khi chọn bthem, chương trình sẽ thêm nội dung trong txt vào con của node đang chọn. Nếu hiện tại không có node nào được chọn thì nội dung trong txt sẽ được thêm vào node gốc



Hình 2.20. Minh họa TreeView

Source code:

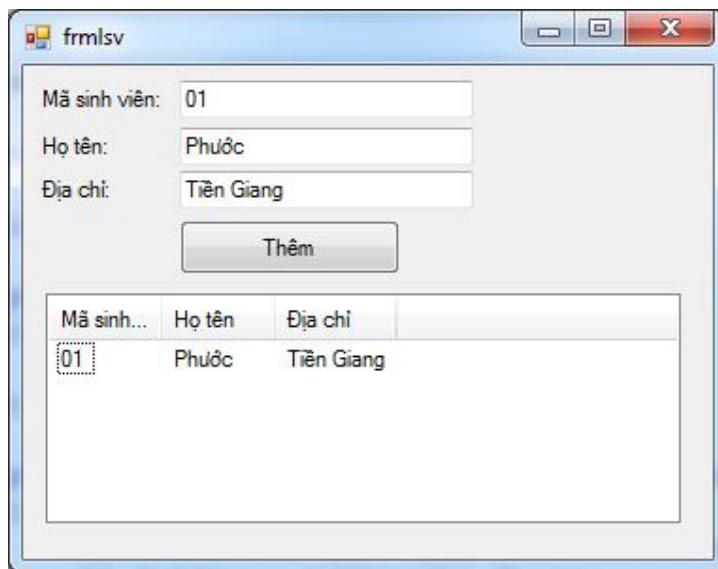
```
if (tr.SelectedNode == null)
{
    tr.Nodes.Add(txt.Text);
    txt.Clear();
    txt.Focus();
}
else
{
    tr.SelectedNode.Nodes.Add(txt.Text);
    txt.Clear();
    txt.Focus();
}
```

2.3.15. Listview

Trên đây ta đã học cách sử dụng control TreeView để hiển thị dữ liệu dạng cây. Phần này chúng ta sẽ sử dụng control ListView để trình bày dữ liệu theo dạng danh sách.

Thuộc tính quan trọng nhất của ListView là *Items* trả về đối tượng *ListViewItemCollection*. Phương thức *Add* của đối tượng này sẽ thêm các phần tử vào trong ListView. Tương tự như *ListBox.ObjectCollection* đã trình bày ở trên, đối tượng này cũng cung cấp những phương thức như *Clear*, *Remove*, *RemoveAt*, ... giúp cho chúng ta có thể cập nhật các phần tử trong ListView (xem thêm trong MSDN).

Ví dụ 2.18: trình bày cách sử dụng control ListView. Chương trình bao gồm ba textbox: mã sinh viên (txtmasv), họ tên (txtht), địa chỉ (txtdc), một listview (lsv) và một button thêm (bthem). Khi chọn nút thêm, dữ liệu trên các textbox sẽ được thêm vào listview.



Hình 2.21. Minh họa TreeView

Source code:

☞ **Lưu ý:** Thiết lập thuộc tính view của listview là Details:

```
private void frmlsv_Load(object sender, EventArgs e)
{
    lsv.Columns.Add("Mã sinh viên");
    lsv.Columns.Add("Họ tên");
    lsv.Columns.Add("Địa chỉ");
}

private void bthem_Click(object sender, EventArgs e)
{
    ListViewItem li = new ListViewItem(txtmasv.Text);
    li.SubItems.Add(txtht.Text);
    li.SubItems.Add(txtdc.Text);
    lsv.Items.Add(li);
}
```

2.3.16. 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. Một số thuộc tính quan trọng của Timer như sau:

- Name: Tên điều khiển Timer.
- Interval: (giá trị là một số n) là chu kỳ thực hiện sự kiện Tick của điều khiển Timer, n là số nguyên, được tính bằng mili giây (ms) và có giá trị >0 .
- Enabled: Enabled = True: cho phép điều khiển Timer hoạt động. Enabled = False: không cho phép điều khiển Timer hoạt động.

Ví dụ 2.19: minh họa cách sử dụng control Timer. Chương trình gồm một Label (lb), hai button Bắt đầu (bstart) và Kết thúc (bend) và một Timer (timer1). Khi chọn bắt đầu, lb sẽ hiển thị đồng hồ gồm phút và giây theo định dạng mm:ss, sau mỗi giây, ss sẽ tăng thêm 1, sau 60 giây mm sẽ tăng thêm 1.



Hình 2.22 Minh họa Timer

Source code:

Lưu ý: chọn thuộc tính interval của timer1 là 1000. Khai báo hai biến toàn cục m và s; cả hai có giá trị ban đầu là 0 (thiết lập trong Formload)

```

private void bstart_Click(object sender, EventArgs e)
{
    timer1.Enabled = true;
}
private void bend_Click(object sender, EventArgs e)
{
    timer1.Enabled = false;
}
private void timer1_Tick(object sender, EventArgs e)
{
    ++s;
    if (s == 60)
    {
        s = 0;
        ++m;
    }
    string kq = "";
    if (m < 10)
        kq = "0" + m + ":" ;
    else
        kq = m + ":" ;
    if (s < 10)
        kq = kq + "0" + s;
    else
        kq = kq + s;
    lb.Text = kq;
}
private void frmtimer_Load(object sender, EventArgs e)
{
    m = 0;
    s = 0;
}

```

2.4. Form và sự thể hiện của các control

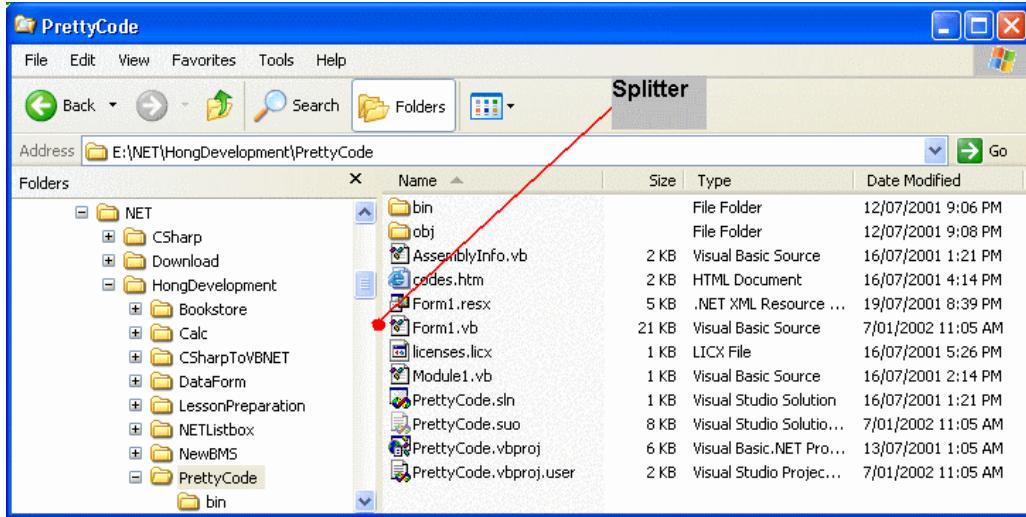
- Tự động thay đổi kích thước (Resize) và định chỗ (Positioning).

Những chương trình ứng dụng chuyên nghiệp thường thường có đặc tính tự thay đổi kích thước của các controls hay định vị trí của các controls trên form một cách tự động. Để xử lý các chức năng trên, ở Visual Basic 6 ta phải thực hiện rất khó khăn. Ta phải ghi nhớ vị trí và kích thước của mỗi control trên form để mỗi lần người dùng thay đổi kích thước form thì ta phải theo đó để thay đổi và định vị trí của control.

.NET cho ta thêm các thuộc tính Anchor và Dock cho mỗi control. Ngoài ra .NET còn cung cấp control Splitter để cho phép ta nắm một thanh phân hai kéo qua, kéo lại hay kéo lên, kéo xuống tùy thích, để mở rộng thêm một bên trong khi bên kia bị thu hẹp.

Áp dụng thông dụng nhất của Splitter là trong Windows Explorer. Trong đó ta có hai phần: bên trái là một Treeview chứa cái cây của ổ đĩa và file folders, bên phải là một Listview chứa icons hay chi tiết của các folder và files. Muốn xem Treeview

nhiều hơn, ta nấm thanh phân hai ở giữa kéo qua bên phải một chút.

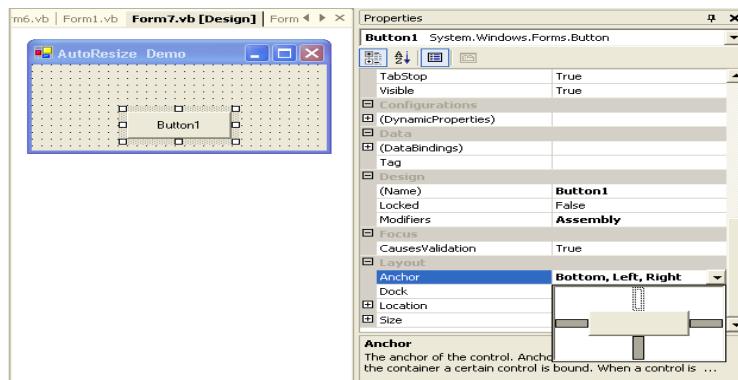


Hình 2.23. Minh họa Splitter

- Thuộc tính Anchor của các control.

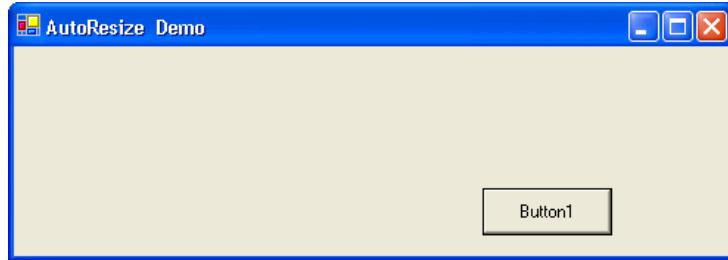
Control trong .NET có thuộc tính Anchor để ta chỉ định nó được buộc vào góc nào đó của form: Left, Right, Bottom hay Top.

Trong lúc thiết kế, sau khi chọn một control (ví dụ Button1), ta vào cửa sổ Properties và click hình tam giác nhỏ bên phải thuộc tính Anchor. Một hình vuông với bốn thanh ráp lại giống hình chữ thập màu trắng sẽ hiện ra. Mỗi thanh tượng trưng cho một góc mà ta có thể chỉ định để buộc control vào form. Ví dụ ta click vào thanh dưới và hai thanh hai bên, ta sẽ có Bottom, Left, Right như trong hình dưới đây.



Hình 2.24. Minh họa Anchor của các control

Khi Button1 có Anchor là Bottom, Right thì mỗi khi góc phải dưới của form di chuyển vì thay đổi kích thước, Button1 sẽ chạy theo góc ấy.



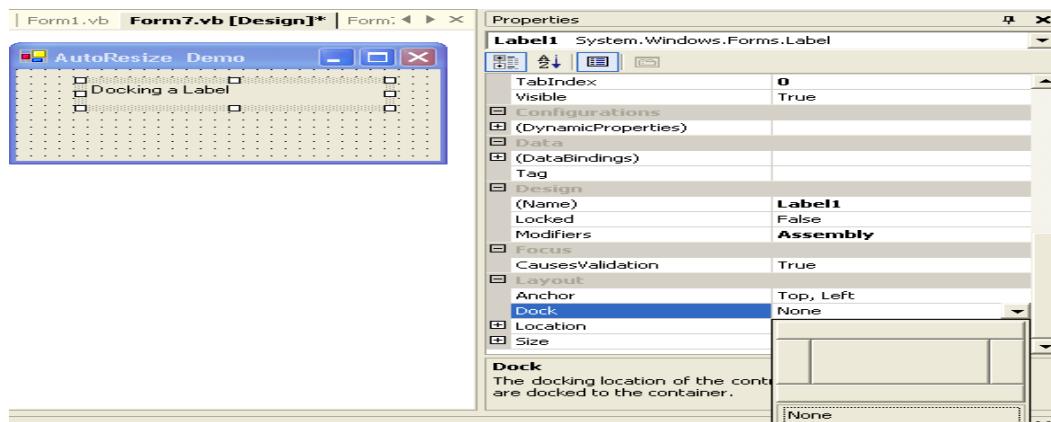
Hình 2.25. Minh họa Anchor cho Button

- Thuộc tính Dock của các control.

Khi ta gắn (docking) một control vào một cạnh của form có nghĩa là ta dán dính nó vào cạnh đó. Áp dụng ta thường thấy nhất của Docking là ToolBar và StatusBar. ToolBar thì gắn vào phía trên của form, còn StatusBar thì gắn vào phía dưới của một form. Chúng dán ra chiếm từ trái qua phải của form, người dùng không thể chỉ định chiều rộng của chúng. Khi form được thay đổi kích thước thì ToolBar và StatusBar cũng dán ra hay co vào theo chiều rộng của form.

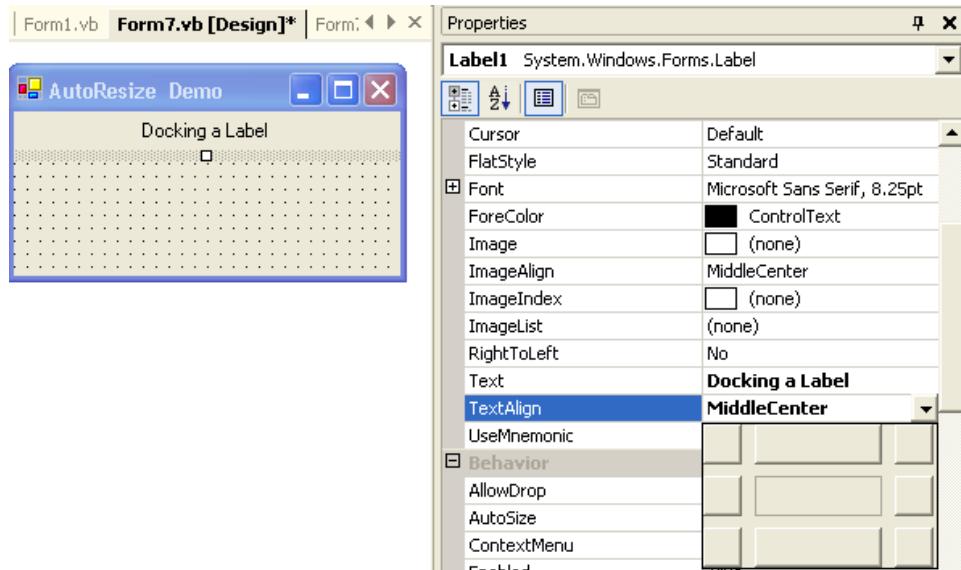
Ta chỉ có thể gắn một control vào một trong bốn cạnh của form, chứ không thể gắn vào bốn cạnh của form. Tuy nhiên, thuộc tính Dock có trị số fill có thể làm cho control chiếm hết bên trong phần còn lại của container chứa nó.

Trong lúc thiết kế, sau khi chọn một control (ví dụ Label1), ta vào cửa sổ Properties và click hình tam giác nhỏ bên phải thuộc tính Dock. Một thanh vuông nhiều thanh màu xám sẽ hiện ra. Mỗi thanh tượng trưng cho một cạnh mà ta có thể chỉ định để gắn control vào form (Top, Bottom, Left hay Right), hình vuông ở giữa tượng trưng cho trị số Fill, và thanh dưới cùng có chữ None cho phép ta xóa không chọn trị số Dock nào cả.



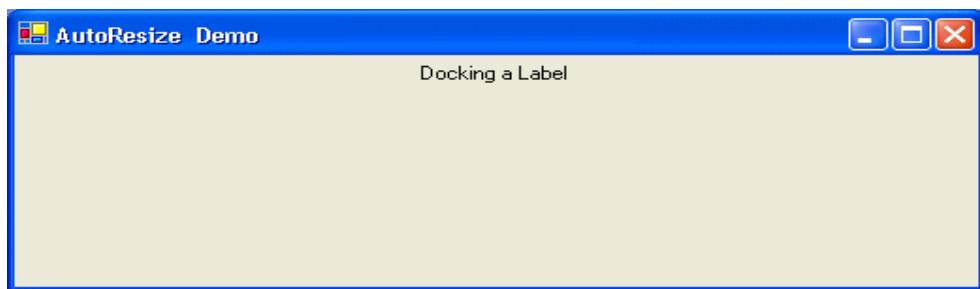
Hình 2.26. Minh họa Dock của các control

Giả sử ta thiết lập thuộc tính TextAlign của Label là MiddleCenter bằng cách chọn thanh xám nằm ngay giữa trong số 9 thanh tượng trưng cho các vị trí của Text có thể nằm trong Label1 như trong hình dưới đây.



Hình 2.27. Minh họa thuộc tính Dock cho Label

Khi chạy chương trình và thay đổi kích thước form cho lớn ra, ta sẽ thấy Label1 dãn ra hai bên, nhưng không hề tăng bè cao, và Text của Label1 luôn luôn nằm ở giữa.



Hình 2.28. Minh họa Dock cho Label

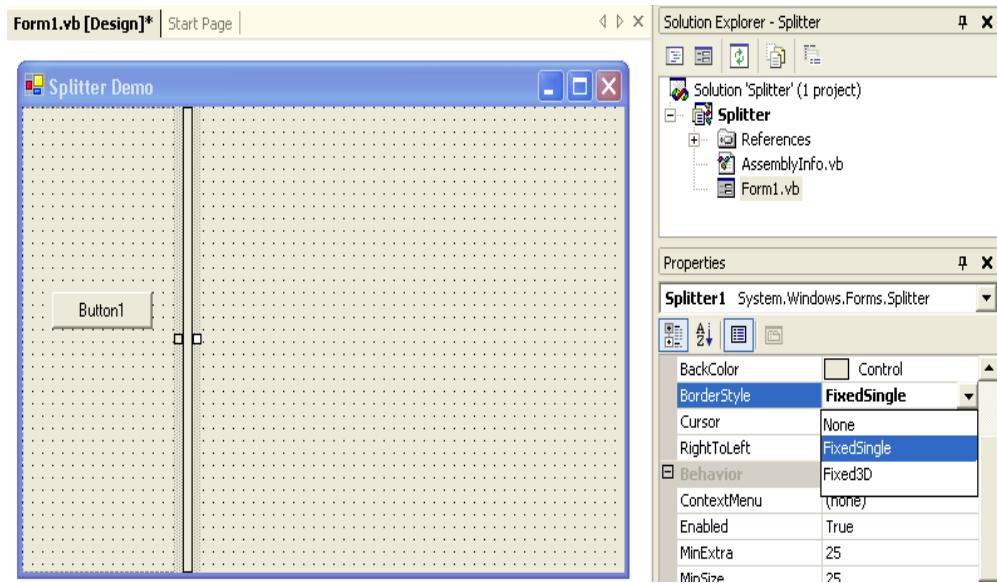
– Control Splitter.

Ta đã hiểu rõ các đặc tính và cách dùng hai thuộc tính Anchor và Dock của control, sau đây ta sẽ áp dụng kiến thức ấy vào việc thiết kế dùng Splitter trong một form. Ví dụ sau sẽ giúp ta hiểu rõ hơn về hai thuộc tính trên và control Splitter.

Tạo một ứng dụng mới, đặt một Panel (một control dạng Container) lên phía trái của form chính để nó chiếm bên trái của form bằng cách thiết lập thuộc tính Dock của nó thành Left. Ta gọi Panel ấy là Panel1.

Đặt một Splitter lên form (tránh đặt nó lên Panel1 vì Panel cũng là một loại container nên có thể chứa Splitter được). Splitter sẽ tự động dock Left vào form tức là nằm bên phải Panel1.

Đặt một button lên Panel1 và thiết lập thuộc tính Anchor của nó thành Top, Left, Right. Böyle giờ form sẽ giống như dưới đây.

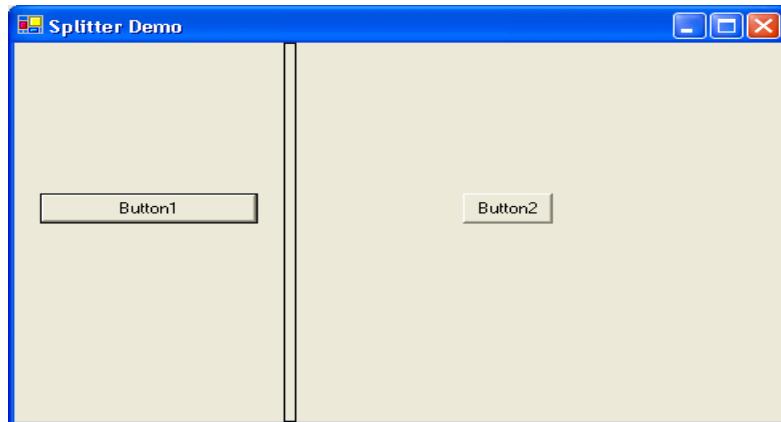


Hình 2.29. Minh họa thuộc tính Splitter

Tiếp theo, đặt một Panel lên bên phải của form, gọi là Panel2, và thiết lập thuộc tính Dock nó thành Fill. Có nghĩa là ta muốn Panel2 chiếm hết phần còn lại bên phải của form.

Thêm vào trong Panel2 này một Button, gọi là Button2, và thiết lập thuộc tính Anchor của nó thành Top, Left, Right.

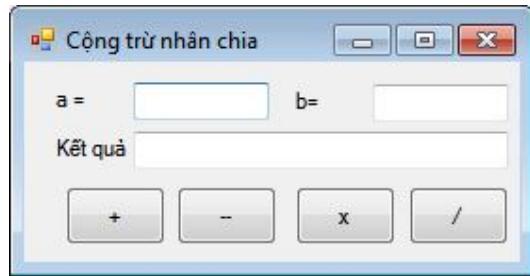
Khi chạy chương trình, mỗi lần ta nắm Splitter kéo qua phải thì Button1 dãn ra và Button2 co lại.



Hình 2.30. Minh họa Splitter

BÀI TẬP CHƯƠNG II

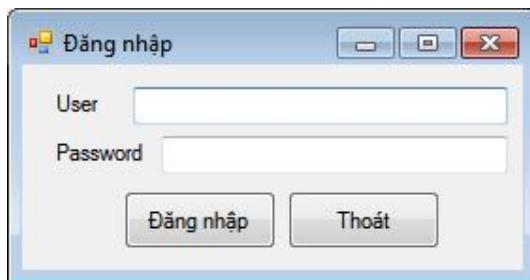
1. Thiết kế Form tính +, -, x, /
- a. Giao diện



b. Yêu cầu: Khi chọn +, -, *, /: hiển thị kết quả tương ứng với phép toán lên Textbox kết quả

2. Thiết kế Form đăng nhập

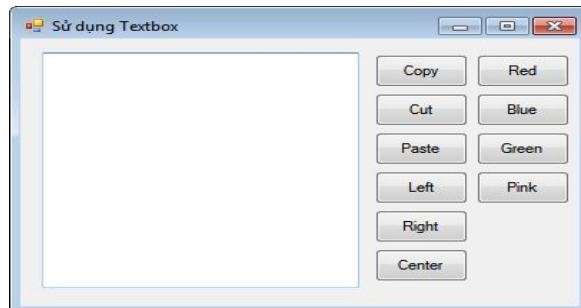
a. Giao diện



b. Yêu cầu:

- Khi chọn nút Đăng Nhập hoặc bấm phím Enter khi gõ Password: Hiển thị User và Password vừa nhập
- Khi chọn nút Thoát: Thoát khỏi Form

3. Thiết kế Form sử dụng Textbox cho phép Copy, Cut, Paste, canh lè, chỉnh màu.

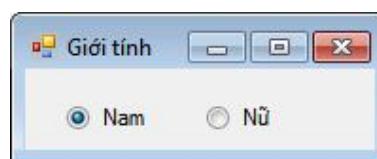


4. Thiết kế Form sử dụng RichTextbox cho phép Copy, Cut, Paste, canh lè, chỉnh màu.



5. Thiết kế Form chọn Giới tính.

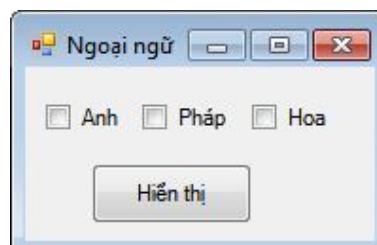
a. Giao diện



b. Yêu cầu: khi chọn giới tính nào thì hiển thị giới tính đó (messagebox)

6. Thiết kế Form chọn Ngoại ngữ

a. Giao diện



– Yêu cầu: chọn hiển thị: hiển thị các ngoại ngữ đã chọn (messagebox)

7. Thiết kế Form tính +, -, x, /

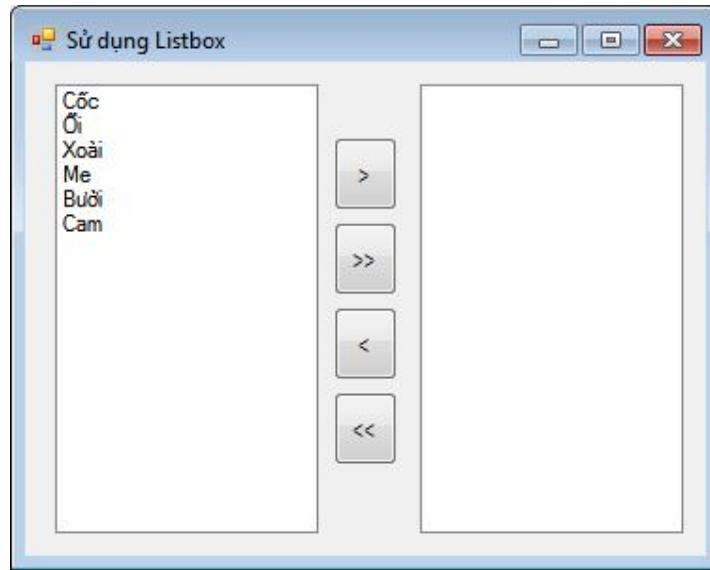
a. Giao diện



b. Yêu cầu: chọn Tính: hiển thị kết quả tương ứng với phép toán đang được chọn

8. Thiết kế Form sử dụng Listbox

a. Giao diện



b. Yêu cầu

- Chọn nút >: Chuyển phần tử đang chọn qua listbox phải
- Chọn nút >>: Chuyển tất cả phần tử qua listbox phải
- Chọn nút <: Chuyển phần tử đang chọn qua listbox trái
- Chọn nút <<: Chuyển tất cả phần tử qua listbox trái

9. Thiết kế Form hiển thị dân tộc

a. Giao diện

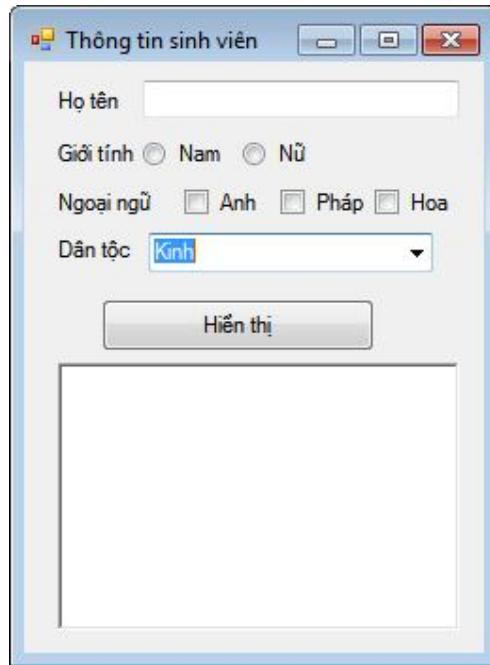


b. Yêu cầu

- Formload: gồm 5 dân tộc: Kinh, Hoa, K'Me, H'Mong, Khác
- Chọn Combobox: Hiển thị dân tộc được chọn (messagebox)

10. Thiết kế Form hiển thị thông tin sinh viên

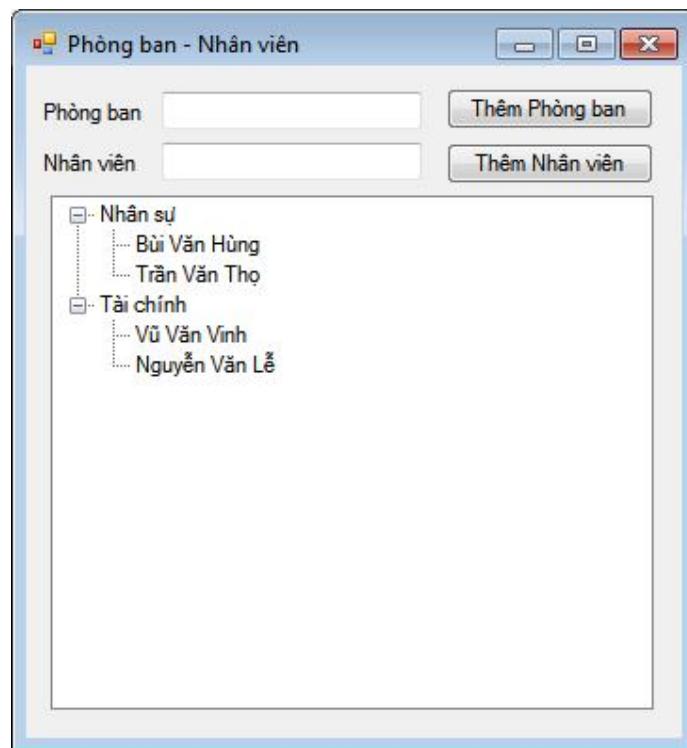
a. Giao diện



b. Yêu cầu: chọn nút Hiển thị: hiển thị các thông tin vừa nhập lên Textbox Hiển thị, mỗi thông tin hiển thị trên một dòng

11. Thiết kế Form hiển thị Phòng ban, Nhân viên

a. Giao diện



b. Yêu cầu

- Chọn nút Thêm Phòng Ban: Thêm Phòng Ban vừa nhập vào Treeview.
- Chọn nút Thêm Nhân Viên: Thêm Nhân viên vừa nhập vào Phòng ban đang

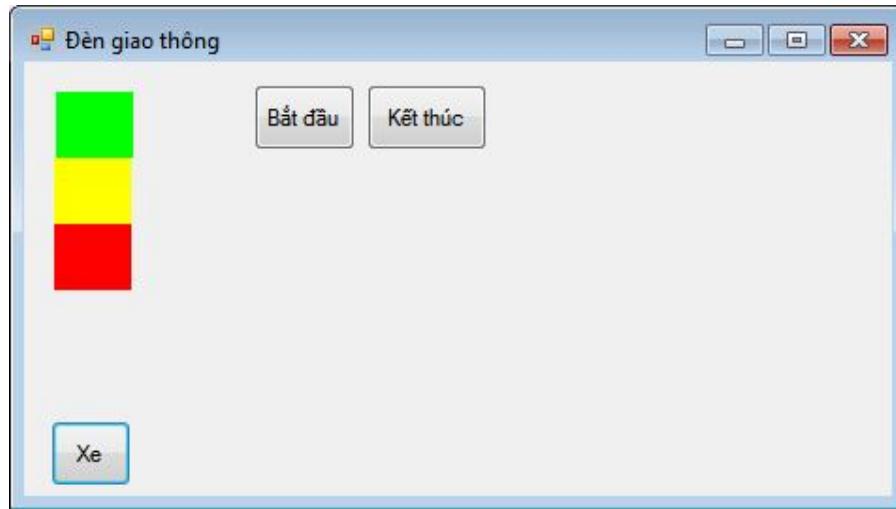
chọn

12. Thiết kế Form với yêu cầu như câu 8, sử dụng Listview (thay vì Listbox)
13. Thông tin sinh viên gồm: Mã số, họ tên, ngày sinh. Thiết kế Form như sau:
 - a. Giao diện

Mã số	Họ tên	Ngày sinh
01	Trần Thanh Phước	12/07/1981
02	Vũ Văn Vinh	12/07/1983

14. Thiết kế Form minh họa đồng hồ tính giờ
 - a. Giao diện

15. Thiết kế Form hiển thị đèn giao thông
 - a. Giao diện

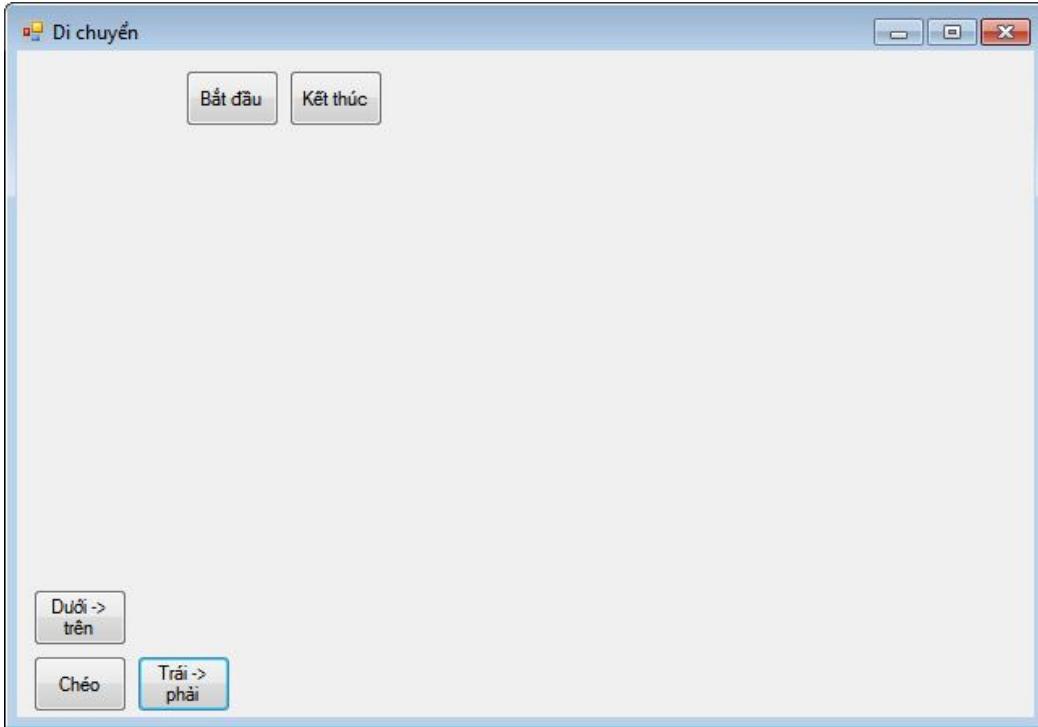


b. Yêu cầu

- Bắt đầu:
 - + Đèn xanh mở
 - + Xe chạy nhanh
- 10 giây sau:
 - + Đèn vàng mở
 - + Xe chạy chậm
- 5 giây sau:
 - + Đèn đỏ mở
 - + Xe dừng hẳn
- 10 giây sau: quay về đèn xanh như lúc đầu
- Kết thúc: Dừng đèn, xe

16. Thiết kế Form di chuyển

a. Giao diện



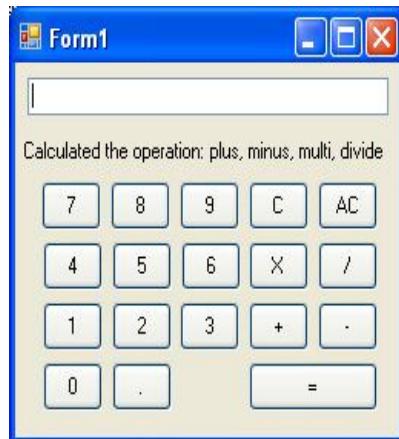
b. Yêu cầu

- Bắt đầu: xe di chuyển qua phải, lên trên, di chuyển chéo
- Kết thúc: Dừng di chuyển

BÀI TẬP NÂNG CAO

1. Xây dựng chương trình mô phỏng máy tính bỏ túi 1 – Calculator1.

a. Giao diện



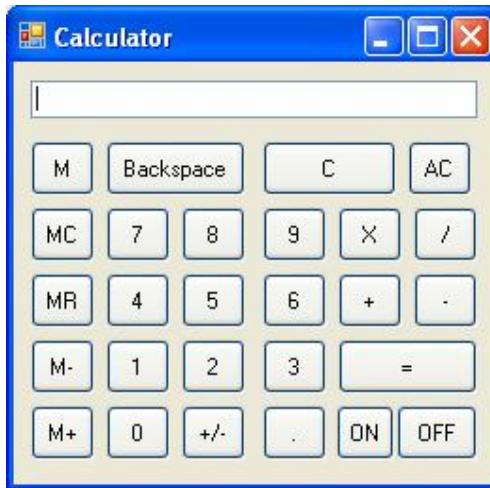
b. Yêu cầu

- Textbox: Chức năng tương tự như màn hình của máy tính.
- Nút lệnh C: Dùng để xoá màn hình hiện hành.
- Nút lệnh AC: Thực hiện tính toán mới.
- Nút lệnh = hiển thị kết quả của các phép tính.

- Nút lệnh 0,1,2,3,4,5,6,7,8,9: Nhập các số cần thực hiện tính toán.
- Nút lệnh +, -, x, /: Chọn phép tính tương ứng.

2. Xây dựng chương trình máy tính bỏ túi 2 – Calculator2.

a. Giao diện



b. Yêu cầu

- ON: Giống chức năng mở máy tính, sau khi nhấn nút này thì:
 - + Không cho thao tác nút ON.
 - + Tất cả các nút còn lại được thao tác.
 - + Màn hình máy tính xuất hiện số 0.
- OFF: Giống như chức năng tắt máy tính, sau khi nhấn nút này thì:
 - + Chỉ cho thao tác nút ON.
 - + Tất cả các nút còn lại không được thao tác.
 - + Màn hình máy tính không xuất hiện gì (đèn trống).
- Backspace: Xóa kí tự cuối cùng trên màn hình.
 - + Ví dụ: Nếu trên màn hình xuất hiện 1234, thì khi nhấn vào nút Backspace, màn hình sẽ còn lại 123.
 - + Nếu trên màn hình xuất hiện số 1, thì khi nhấn vào nút Backspace, màn hình sẽ còn lại 0.
- M+: Cộng giá trị hiện tại trên màn hình máy tính vào bộ nhớ, ví dụ:
 - + Nếu giá trị bộ nhớ = 0 và giá trị trên màn hình = 10 thì kết quả = 10.
 - + Nếu giá trị bộ nhớ = 10 và giá trị trên màn hình bằng 20 thì kết quả là 30.
- M-: Trừ giá trị hiện tại trên màn hình máy tính vào bộ nhớ. Ví dụ:
 - + Nếu giá trị bộ nhớ = 0 và giá trị trên màn hình = 10 thì kết quả = - 10.
 - + Nếu giá trị bộ nhớ = 30 và giá trị trên màn hình = 20 thì kết quả = 10.
- MC: Xóa bộ nhớ.

- MR: Hiển thị giá trị trong bộ nhớ ra màn hình.
- M: Cho biết trạng thái của bộ nhớ.
 - + M: Đang nhớ (giá trị bộ nhớ $<> 0$).
 - + R: Không thực hiện chức năng nhớ (giá trị ô nhớ = 0).
- +/-: Đổi giá trị trên màn hình, số âm thành số dương và ngược lại.

3. Xây dựng chương trình minh họa chức năng của một máy điện thoại để bàn với giao diện như sau

a. Giao diện



b. Yêu cầu

- Nút 0,1,...,9: Nhập các số điện thoại.
- Nút SP: Thực hiện bắt đầu quay số hay kết thúc cuộc gọi (tương tự như chức năng nhắc máy và bỏ máy).
 - + Nhấn nút SP lần thứ nhứt bắt đầu quay số.
 - + Nhấn nút SP lần hai bỏ máy xuống (gác máy).
 - + Ngoài ra nút SP còn có một số chức năng khác (xem thêm kết hợp SP với các nút M1, M2...).
- >>: Bắt đầu cuộc gọi.
 - + Thực hiện nhấn nút SP lần thứ nhất.
 - + Quay số cần gọi.
 - + Nhấn >> để bắt đầu cuộc gọi.
 - + Redial: Quay lại số vừa gọi.
- M1, M2, M3, M4: Lưu hay quay các số trong bộ nhớ.

- + Lưu số vào bộ nhớ.
- + Quay số sử dụng bộ nhớ

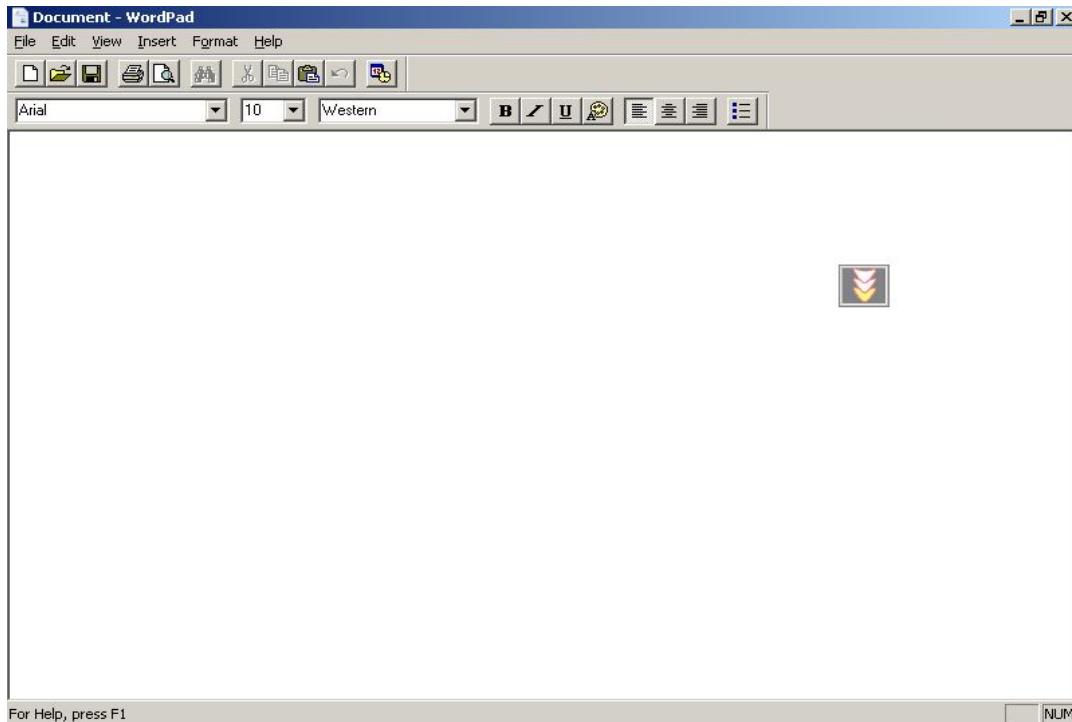
Quy trình hoạt động của một số chức năng trong điện thoại.

- Gọi số mới.
 - + Nhấn nút SP.
 - + Quay số bằng cách nhấn các nút số: 0,1,2,3,...
 - + Nhấn nút >> để bắt đầu cuộc gọi.
- Kết thúc cuộc gọi.
 - + Nhấn nút SP.
- Lưu số vào bộ nhớ.
 - + Nhấn nút SP.
 - + Quay số cần lưu bằng cách nhấn các nút số: 0,1,2,3...
 - + Nhấn nút M1 (M2, M3, M4) để lưu số vừa quay vào bộ nhớ M1 (M2,M3,M4).
- Gọi từ bộ nhớ.
 - + Nhấn nút SP.
 - + Nhấn nút M1 (M2, M3, M4) để lấy số từ bộ nhớ M1 (M2, M3, M4) đưa lên màn hình.
 - + Nhấn nút >> để bắt đầu cuộc gọi.
- Gọi lại số vừa gọi.
 - + Nhấn nút SP.
 - + Nhấn nút Redial.
- Sau khi kết thúc cuộc gọi, thực hiện tính cước điện thoại theo qui tắc sau:
 - + 1 phút đầu: 2000.
 - + 2 phút tiếp theo mỗi phút: 1000.
 - + Từ phút thứ 4 – 5: Mỗi phút 500.
 - + Từ phút 6 – 8: Mỗi phút 300.
 - + Từ phút 9 trở đi mỗi phút: 200.
- Ví dụ tính cước điện thoại.
 - + Giả sử khách hàng gọi 1 phút thì số tiền khách hàng phải trả là Số tiền = 2000.
 - + Giả sử khách hàng gọi 2 phút thì số tiền khách hàng phải trả là Số tiền = 2000 + 1000 * (2-1).
 - + Giả sử khách hàng gọi 5 phút thì số tiền khách hàng phải trả là Số tiền = 2000 + 1000*2 + (5-3)*500.

- + Giả sử khách hàng gọi 7 phút thì số tiền khách hàng phải trả là Số tiền = $2000 + 1000*2 + 500*2 + (7-5)*300$.
- + Giả sử khách hàng gọi 10 phút thì số tiền khách hàng phải trả là Số tiền = $2000 + 1000*2 + 500*2 + 300*3 + (10-8)*200$.

4. Xây dựng chương trình WordPad.

a. Giao diện

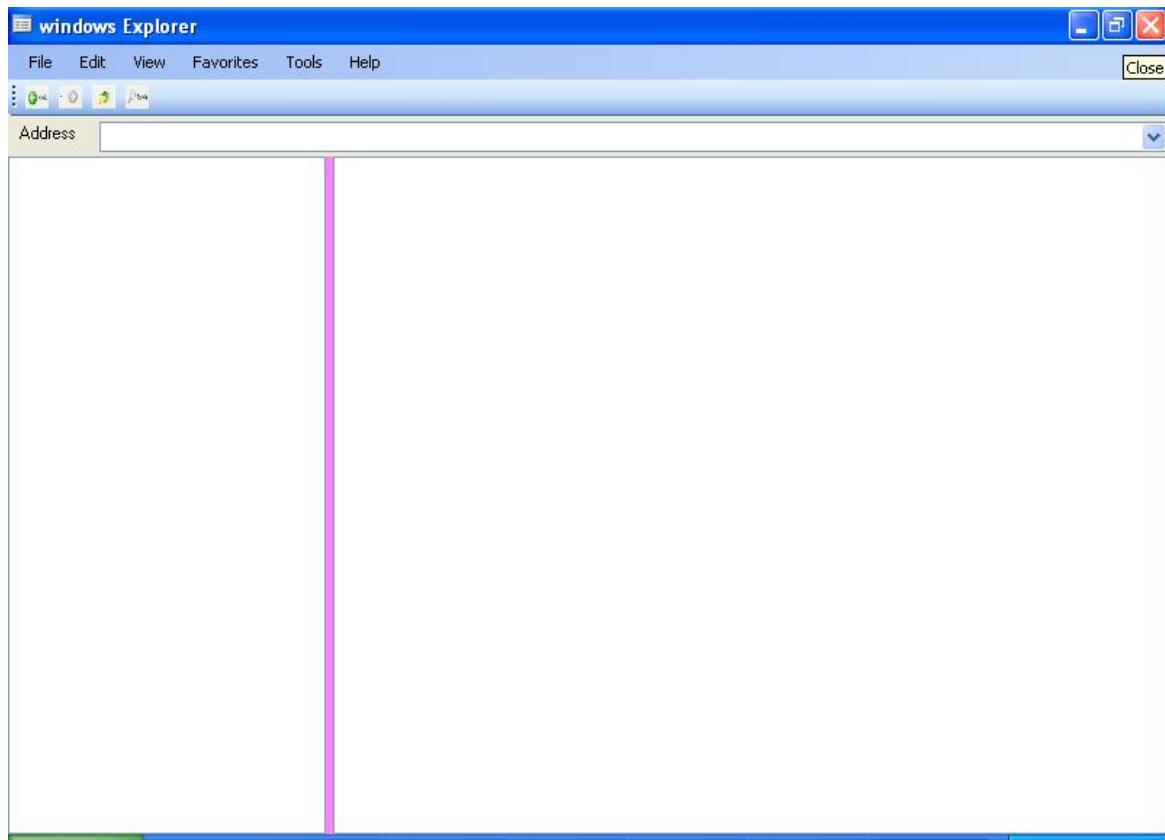


b. Yêu cầu

- Menu file gồm các phần tử con sau: New, Open, Save, Save As, Print, Print Preview, Page Setup, Exit.
- Menu Edit gồm: Cut, Copy, Paste, Clear, Select All.
- Menu View gồm: ToolBar, Format Bar.
- Menu Format gồm: Font, Color

5. Xây dựng chương trình Windows Explorer.

a. Giao diện



b. Yêu cầu

- Menu file gồm: New, Open, Delete, Rename, Close.
- Menu Edit gồm: Cut, Copy, Paste, Select All.
- Menu View gồm: Standard Button, Address Bar.

CHƯƠNG 3. COMMON DIALOG

Mục tiêu:

Sau khi học xong chương này, sinh viên có khả năng:

- Nhận biết được sự cần thiết của các Common Dialog trong các ứng dụng Window.
- Áp dụng các Common Dialog để hoàn chỉnh các ứng dụng của chương 3: Notepad, Wordpad, Windows Explorer.

3.1. Giới thiệu

Các chương trình Window phần lớn hỗ trợ lập trình giao diện đồ họa thông qua các cửa sổ Form. Để cho phép chương trình thực hiện các thao tác như mở, lưu giữ, hoặc in file, môi trường .NET cung cấp các hộp thoại chuẩn cho phép chương trình sử dụng tương tác với người dùng. Trong chương này chúng ta sẽ khảo sát cách sử dụng chi tiết mỗi lớp đối tượng biểu diễn hộp hội thoại chuẩn (common Dialog).

3.2. Mở tập tin

3.2.1. Giới thiệu OpenFileDialog

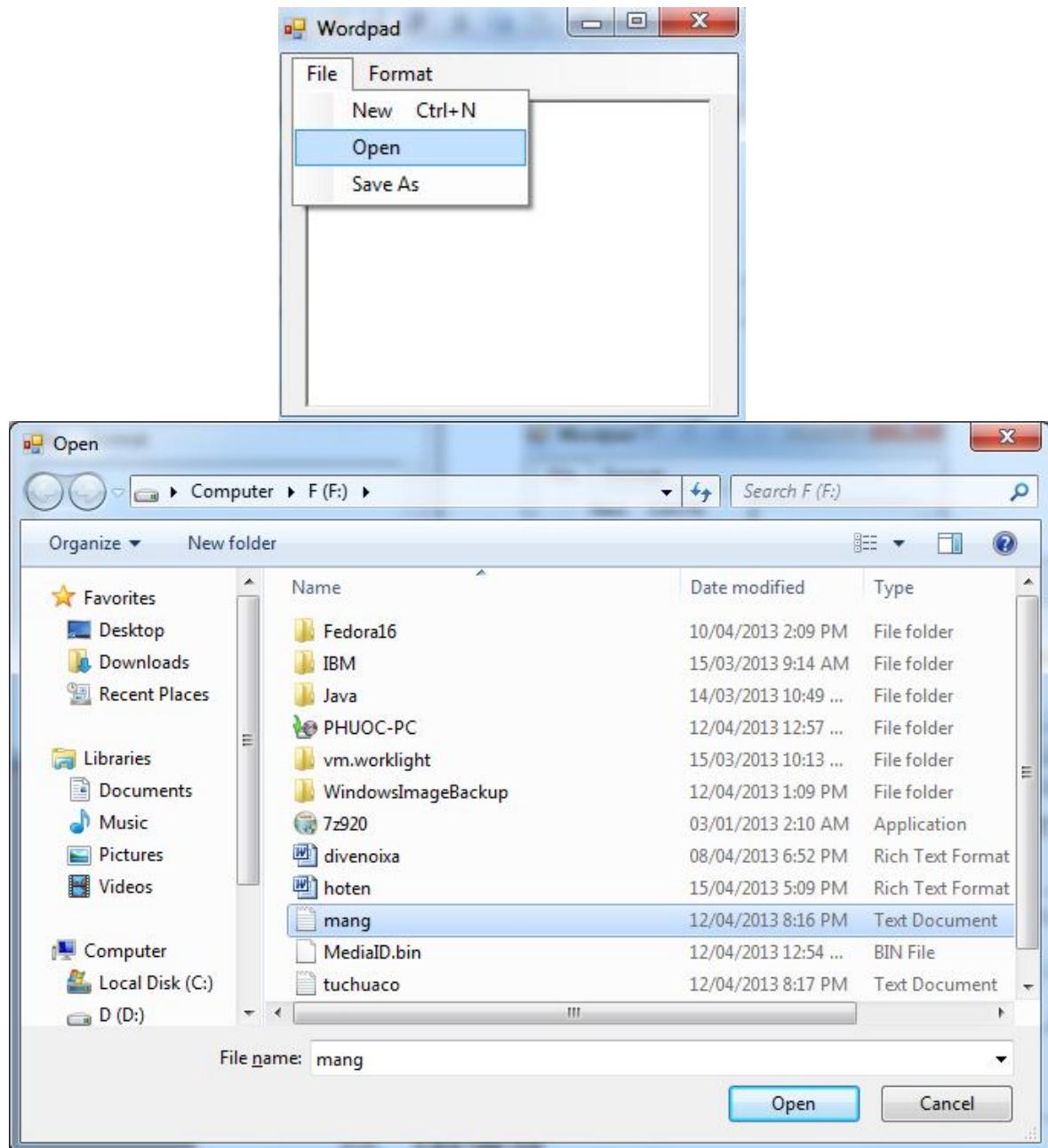
Hộp thoại mở file được sử dụng rất rộng rãi. Trong hộp thoại Open người dùng có thể duyệt danh sách file qua ô đĩa logic của bộ hoặc mạng để định vị trí file mà chương trình cho phép mở.

3.2.2. Một số thuộc tính và phương thức cơ bản

- Khai báo và khởi tạo:
 - + OpenFileDialog OpenFileDialog
 - + openFileDialog = new OpenFileDialog()
- Thuộc tính Filter: Lọc ra những loại File cần hiển thị. Ta phân cách các bộ lọc bằng dấu | . Ví dụ: openFileDialog.Filter = "All files | .* | Word files | *.doc | Text files | *.txt"
- Thuộc tính FilterIndex:
 - + Dùng để chỉ định loại file làm bộ lọc mặc định.
 - + Ví dụ: để Word file làm bộ lọc mặc định: openFileDialog.FilterIndex = 2.
- Thuộc tính InitialDirectory:
 - + Chỉ định thư mục ban đầu, chương trình có thể gán đường dẫn thư mục tương ứng cho thuộc tính InitialDirectory.
 - + Ví dụ: openFileDialog.InitialDirectory = "C:\Programs".
- Phương thức ShowDialog(): dùng để hiển thị hộp thoại mở File.

- Thuộc tính Filename: Trả về tên file đã chọn trong hộp thoại mở file.

Ví dụ 3.1: minh họa cách sử dụng openFileDialog. Chương trình tựa wordpad bao gồm các menu: File (New, Open, Save As), Format (Font, Color), ví dụ này cũng được thực hiện cho các common dialog tiếp theo. Khi chọn open, xuất hiện hộp thoại Open; chọn file, chọn open -> hiển thị file vừa chọn.





Hình 3.1. Minh họa OpenFileDialog

Source code:

```
private void mnopen_Click(object sender, EventArgs e)
{
    OpenFileDialog o = new OpenFileDialog();
    o.InitialDirectory = "F:\\";
    if (o.ShowDialog() == DialogResult.OK)
        MessageBox.Show(o.FileName);
}
```

3.3. Lưu tập tin

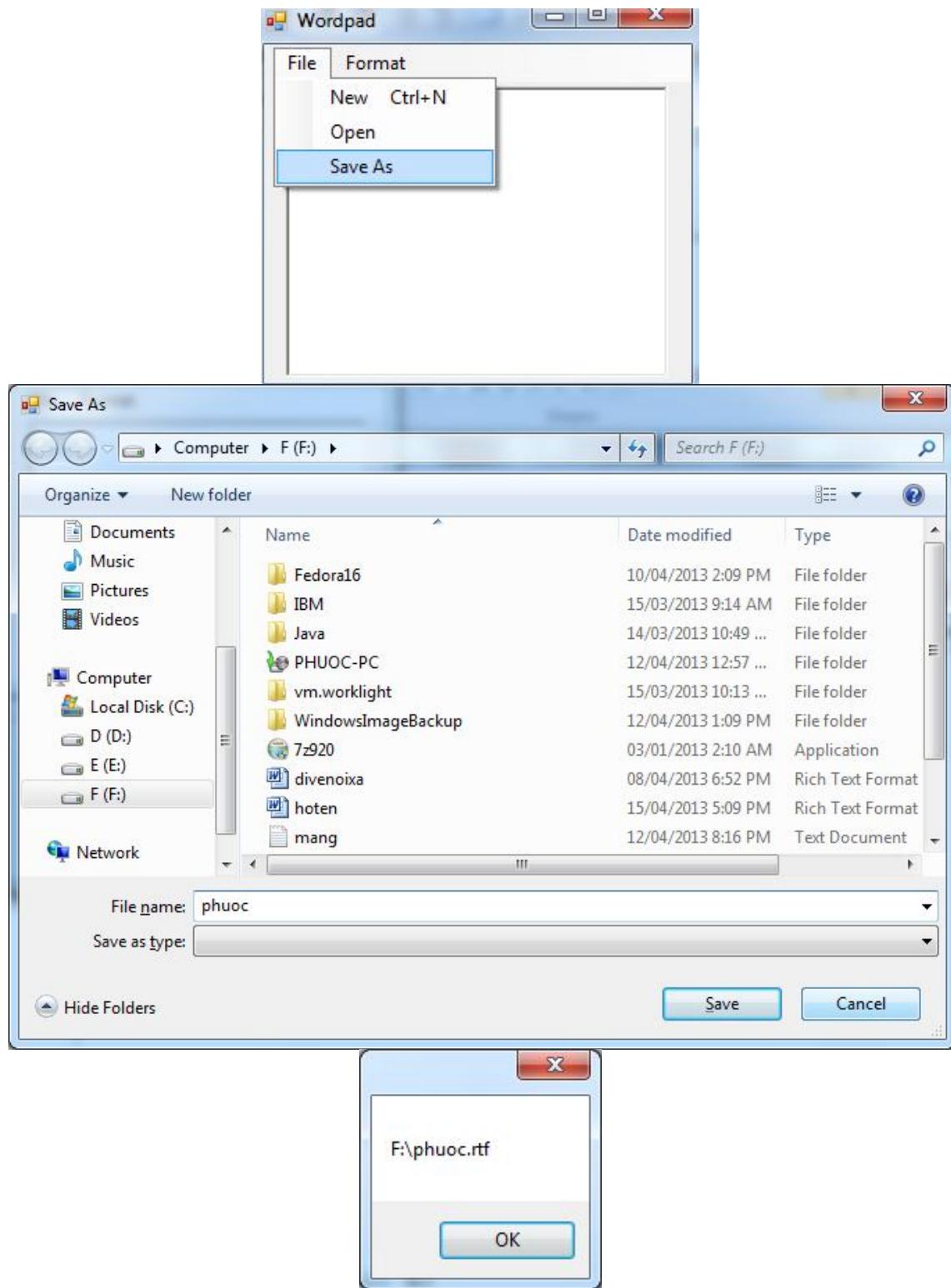
3.3.1. Giới thiệu SaveFileDialog

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

3.3.2. Một số thuộc tính và phương thức cơ bản

- Khai báo và khởi tạo:
 - + SaveFileDialog SaveFile
 - + SaveFile = new SaveFileDialog()
- Thuộc tính Filter, FilterIndex, InitialDirectory, phương thức ShowDialog(): Giống với đối tượng OpenFileDialog.
- Thuộc tính InitialDirectory:
 - + Chỉ định thư mục ban đầu, chương trình có thể gán đường dẫn thư mục tương ứng cho thuộc tính InitialDirectory.
 - + Ví dụ: SaveFile.InitialDirectory = “C:\Programs”.
- Thuộc tính AddExtension: Là True hoặc False, cho phép hộp thoại tự nối phần tên mở rộng vào tên file hay không nếu không được người dùng chỉ rõ.
- Thuộc tính DefaultExt: Tên mở rộng được thêm vào file nếu người dùng không chỉ rõ, giả sử thuộc tính AddExtension là true.
- Thuộc tính Filename: Tên file mà người dùng vừa nhập trong hộp thoại.
- Phương thức ShowDialog(): Dùng để hiển thị hộp thoại lưu file.

Ví dụ 3.2: minh họa cách sử dụng SaveFileDialog, cách thực hiện tương tự như trường hợp OpenFileDialog:



Hình 3.2. Minh họa SaveFileDialog

Source code:

```

private void mnsaveas_Click(object sender, EventArgs e)
{
    SaveFileDialog s = new SaveFileDialog();
    s.InitialDirectory = "F:\\";
    s.AddExtension = true;
    s.DefaultExt = ".rtf";
    if (s.ShowDialog() == DialogResult.OK)
        MessageBox.Show(s.FileName);
}

```

3.4. Chọn Font chữ

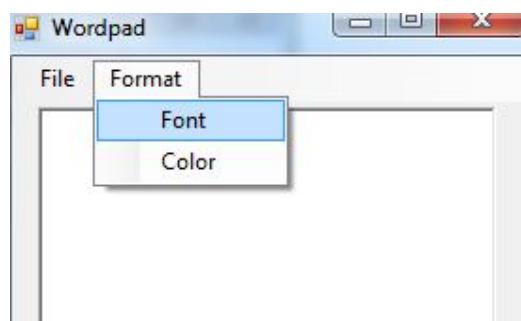
3.4.1. Giới thiệu FontDialog

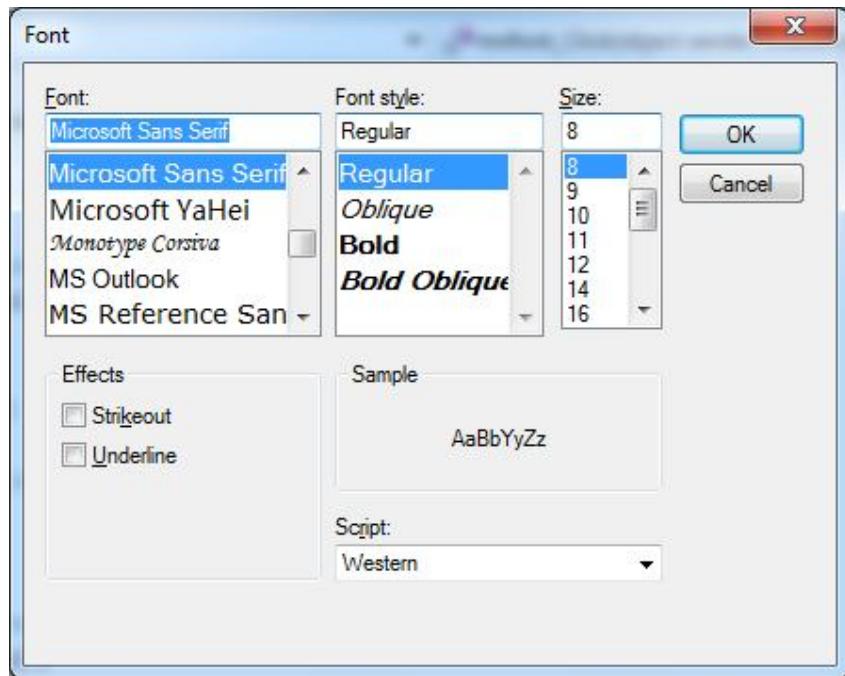
Trong các ứng dụng Window ngày nay, hộp thoại Font được sử dụng rất rộng rãi. Hộp thoại Font cho phép chúng ta chọn tên Font, loại Font, kích thước.....

3.4.2. Một số thuộc tính và phương thức cơ bản

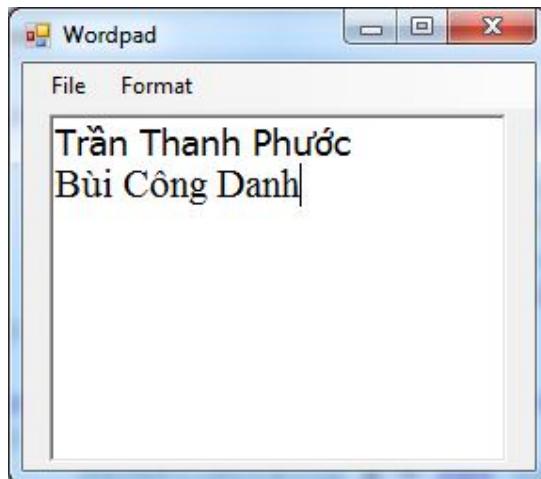
- Khai báo và khởi tạo:
 - + FontDialog fontdb
 - + fontdb = new FontDialog()
- Thuộc tính quan trọng nhất của lớp FontDialog là Font, trong thuộc tính Font có một số thuộc tính quan trọng sau:
 - + Name: Tên Font.
 - + Bold: Là True thì in đậm.
 - + Italic: Là True thì có in nghiêng.
 - + Size: Kích cỡ Font.
 - + Strikeout: Là True thì có gạch ngang.
 - + Underline: Là True thì có gạch dưới.
- Phương thức ShowDialog(): Dùng để hiển thị hộp thoại chọn Font.

Ví dụ 3.3: minh họa cách sử dụng FontDialog. Chương trình cho phép người dùng định dạng Font cho phần văn bản đang chọn.





Hình 3.3. Hộp thoại chọn Font



Hình 3.4. Thông tin Font đã chọn

Source code:

```
private void mnfont_Click(object sender, EventArgs e)
{
    FontDialog f = new FontDialog();
    if (f.ShowDialog() == DialogResult.OK)
        rtf.SelectionFont = f.Font;
}
```

3.5. Chọn màu

3.5.1. Giới thiệu ColorDialog

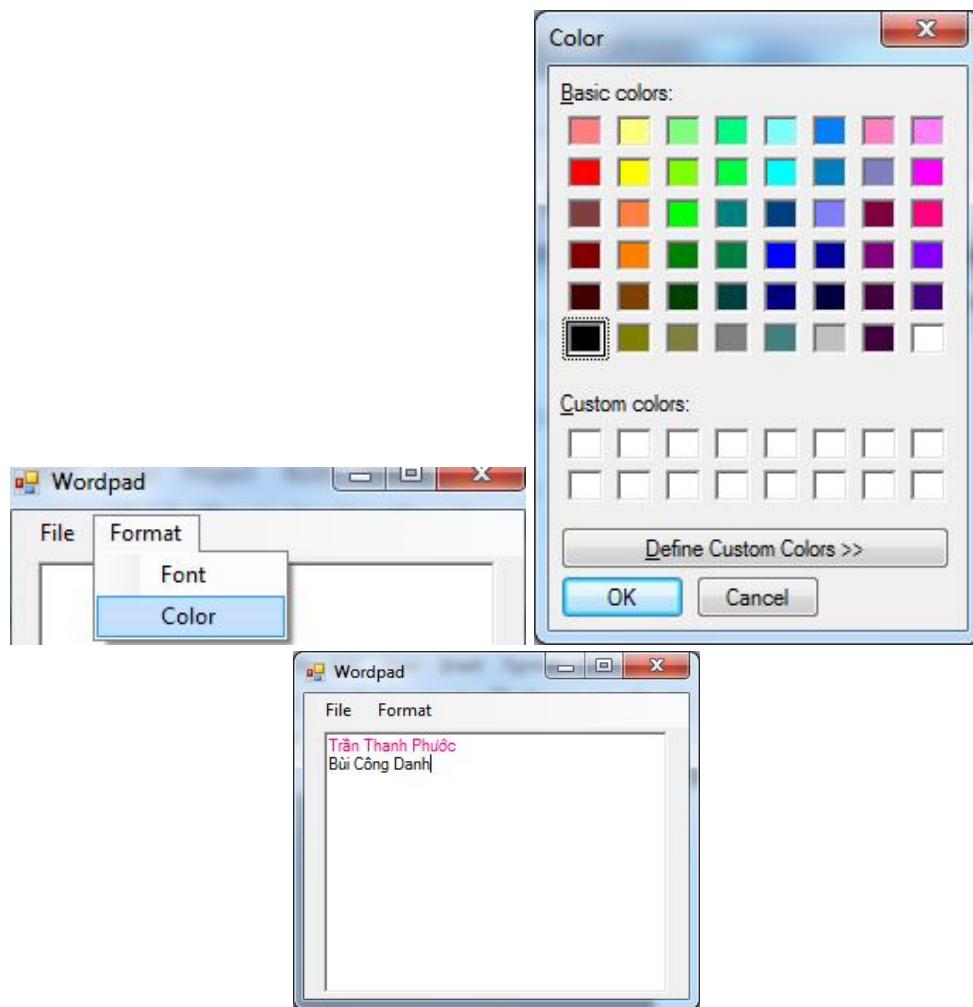
Tương tự như chọn Font, hộp thoại chọn màu(Color) cũng được sử dụng khá rộng rãi để chọn màu tông, màu nền, màu font chữ,... Windows cung cấp hộp thoại màu chuẩn, người dùng có thể lựa chọn từ một trong số 64 màu đặt sẵn(48 màu định nghĩa

bởi Windows và 16 màu định nghĩa bởi chương trình hoặc người dùng).

3.5.2. Một số thuộc tính và phương thức cơ bản

- Khai báo và khởi tạo:
 - + ColorDialog myColor
 - + myColor = new ColorDialog()
- Phương thức ShowDialog(): Dùng để hiển thị hộp thoại chọn màu.
- Thuộc tính Color: Trả về giá trị màu mà người dùng vừa chọn trong hộp thoại chọn màu.

Ví dụ 3.4: minh họa cách sử dụng ColorDialog. Chương trình cho phép người dùng định dạng màu sắc cho phần văn bản đang chọn.



Hình 3.5. Hộp thoại ColorDialog

Source code:

```

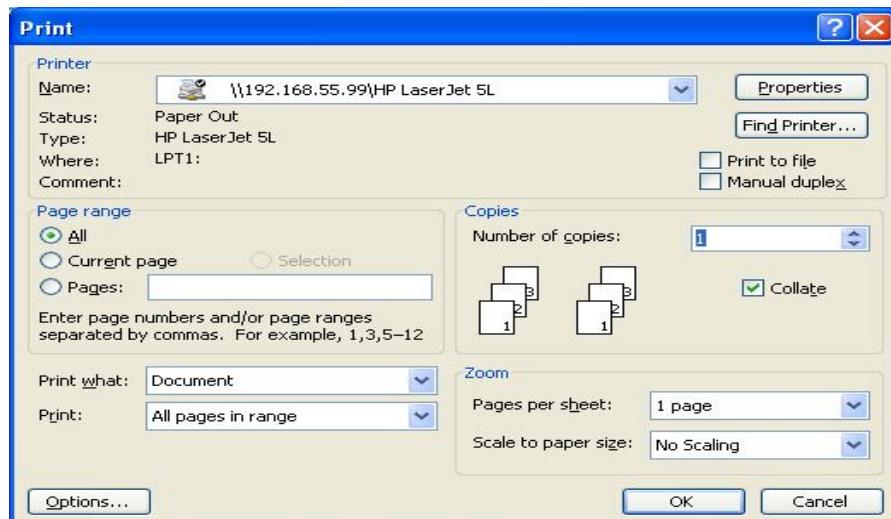
private void mnccolor_Click(object sender, EventArgs e)
{
    ColorDialog c = new ColorDialog();
    if (c.ShowDialog() == DialogResult.OK)
        rtf.SelectionColor = c.Color;
}

```

3.6. In ấn

3.6.1. PrintDialog

Khi người dùng tùy chọn Print trên menu File, đa số các ứng dụng Windows hiển thị hộp thoại như hình bên dưới; trong đó người dùng có thể chỉ rõ các thông số trước khi in, như lựa chọn máy in, trang in, in qua mạng...



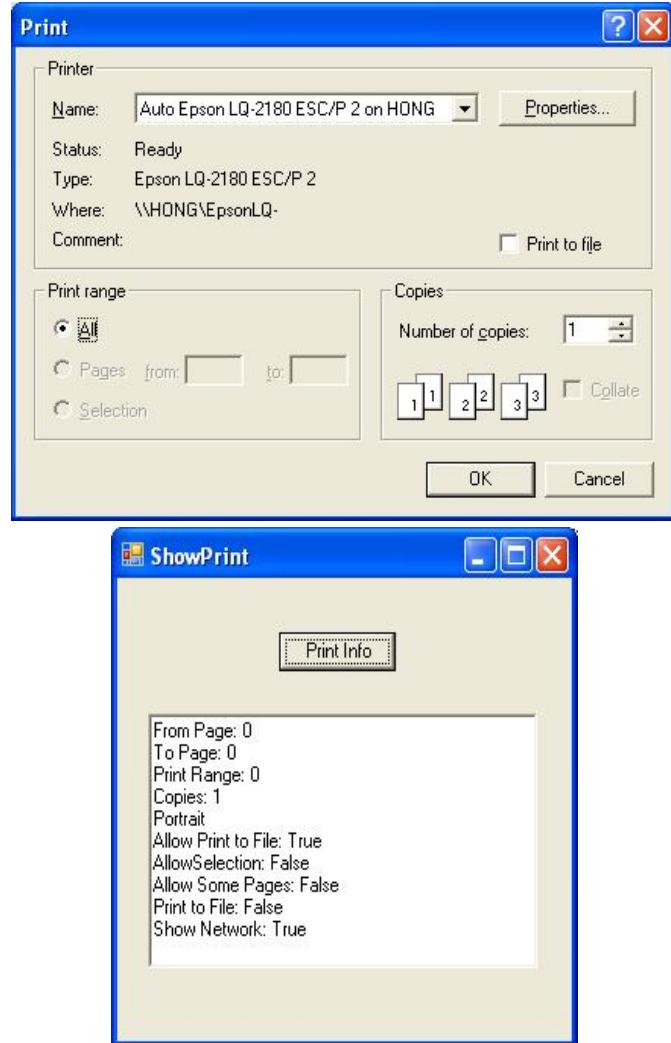
Hình 3.6. Hộp thoại PrintDialog

- Khai báo và khởi tạo: PrintDialog PrintDB = new PrintDialog();

Trước khi hiển thị hộp thoại in, chương trình phải cung cấp đối tượng PrintDialog cho đối tượng PrinterSettings chỉ rõ thông tin cấu hình cần phải in. Một trong những cách cung cấp đối tượng PrinterSettings là gán đối tượng PrintDocument cho thuộc tính Document của lớp đối tượng PrintDialog như sau:

- PrintDb.Document = new System.Drawing.Printing.PrintDocument();
- Phương thức ShowDialog(): Dùng để hiển thị hộp thoại in.

Ví dụ 3.5: tạo mới một Form và thêm một Button tên Button1, một textbox tên textBox1 lên Form. Gõ đoạn code sau trong sự kiện Click của Button1. Biên dịch và thực thi chương trình, khi người dùng click chuột vào button sẽ hiển thị hộp thoại PrintDialog. Sau khi người dùng lựa chọn các thông số in và nhấn nút OK, chương trình sẽ hiển thị những thông tin lựa chọn của người dùng.



Hình 3.7. Thông tin hộp thoại PrintDialog

Source code:

```

Private void Button1_Click(Object sender, EventArgs e)
{
    PrintDB.Document = New System.Drawing.Printing.PrintDocument()
    if (PrintDB.ShowDialog() == DialogResult.OK)
    {
        TextBox1.Text = "";
        TextBox1.AppendText("Printer: " +
        PrintDB.PrinterSettings.PrinterName);
        TextBox1.AppendText("\n");
        TextBox1.AppendText("From Page: " +
        PrintDB.PrinterSettings.FromPage);
        TextBox1.AppendText("\n");
        TextBox1.AppendText("To Page: " +
        PrintDB.PrinterSettings.ToPage);
        TextBox1.AppendText("\n");
        TextBox1.AppendText("Print Range: " +
        PrintDB.PrinterSettings.PrintRange);
    }
}

```

```

        TextBox1.AppendText("\n");
        TextBox1.AppendText("Print Range: " +
PrintDB.PrinterSettings.PrintRange);
        TextBox1.AppendText("\n");
        TextBox1.AppendText("Copies: " +
PrintDB.PrinterSettings.Copies);
        TextBox1.AppendText("\n");
        if (PrintDB.PrinterSettings.LandscapeAngle ==90)
            TextBox1.AppendText("Landscape");
        else
            TextBox1.AppendText("Portrait");
        TextBox1.AppendText("\n");
        TextBox1.AppendText("Allow Print to File: " +
PrintDB.AllowPrintToFile);
        TextBox1.AppendText("\n");
        TextBox1.AppendText("AllowSelection: " +
PrintDB.AllowSelection);
        TextBox1.AppendText("\n");
        TextBox1.AppendText("Allow Some Pages: " +
PrintDB.AllowSomePages);
        TextBox1.AppendText("\n");
        TextBox1.AppendText("Print to File: " +
PrintDB.PrintToFile);
        TextBox1.AppendText("\n");
        TextBox1.AppendText("Show Network: " + PrintDB.ShowNetwork);
    }
}

```

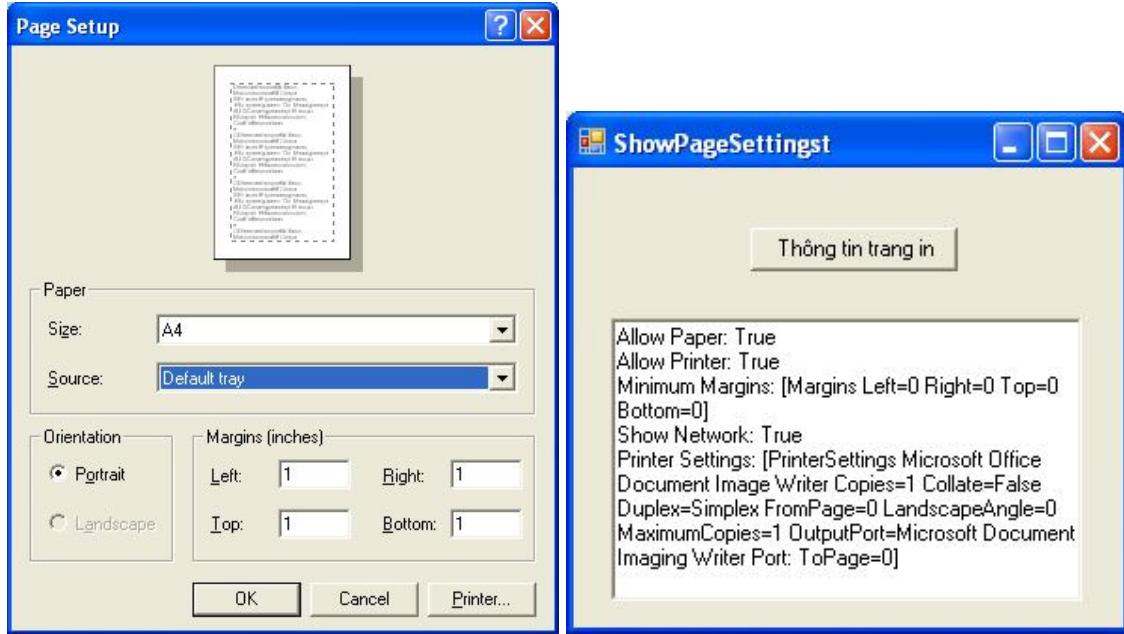
3.6.2. Pagesetupdialog

Khi người dùng chọn Page Setup từ menu File của các ứng dụng Windows, hệ thống thường hiển thị hộp thoại cho PageSetup cho phép xác định thông số trang in như kích thước, hướng trang, nguồn in...

Trong C# ta có thể hiển thị hộp thoại PageSetup xử lý thông số trang qua lớp PageSetupDialog.

- Khai báo và khởi tạo: PageSetupDialog pageDb = neww PageSetupDialog()
- pageDb.Document = new System.Drawing.Printing.PrintDocument()
- Hiển thị hộp thoại: pageDb.showDialog().

Ví dụ 3.6: Tạo mới một Form và thêm một Button tên Button1, một textbox tên textBox1 lên Form. Gõ đoạn code sau trong sự kiện Click của Button1. Biên dịch và thực thi chương trình, khi người dùng click chuột vào button sẽ hiển thị hộp thoại Page Setup. Sau khi người dùng lựa chọn các thông số in và nhấn nút OK, chương trình sẽ hiển thị những thông tin lựa chọn của người dùng.



Hình 3.8. Hộp thoại PageSetup

Source code:

```

Private void Button1_Click (Object sender, EventArgs e)
{
    PageSetupDialog PageDB = New PageSetupDialog()
    PageDB.Document = New System.Drawing.Printing.PrintDocument();
    if (PageDB.ShowDialog() == DialogResult.OK)
    {
        TextBox1.Text = "";
        TextBox1.AppendText("Allow Margins: " +
        PageDB.AllowMargins);
        TextBox1.AppendText("\n");
        TextBox1.AppendText("Allow Orientation: " +
        PageDB.AllowOrientation);
        TextBox1.AppendText("\n");
        TextBox1.AppendText("Allow Paper: " + PageDB.AllowPaper);
        TextBox1.AppendText("\n");
        TextBox1.AppendText("Allow Printer: " +
        PageDB.AllowPrinter);
        TextBox1.AppendText("\n");
        TextBox1.AppendText("Minimum Margins: " +
        PageDB.MinMargins.ToString());
        TextBox1.AppendText("\n");
        TextBox1.AppendText("Show Network: " + PageDB.ShowNetwork);
        TextBox1.AppendText(Vb"\n");
        TextBox1.AppendText("Printer Settings: " +
        PageDB.PrinterSettings.ToString());
        TextBox1.AppendText("\n");
    }
}

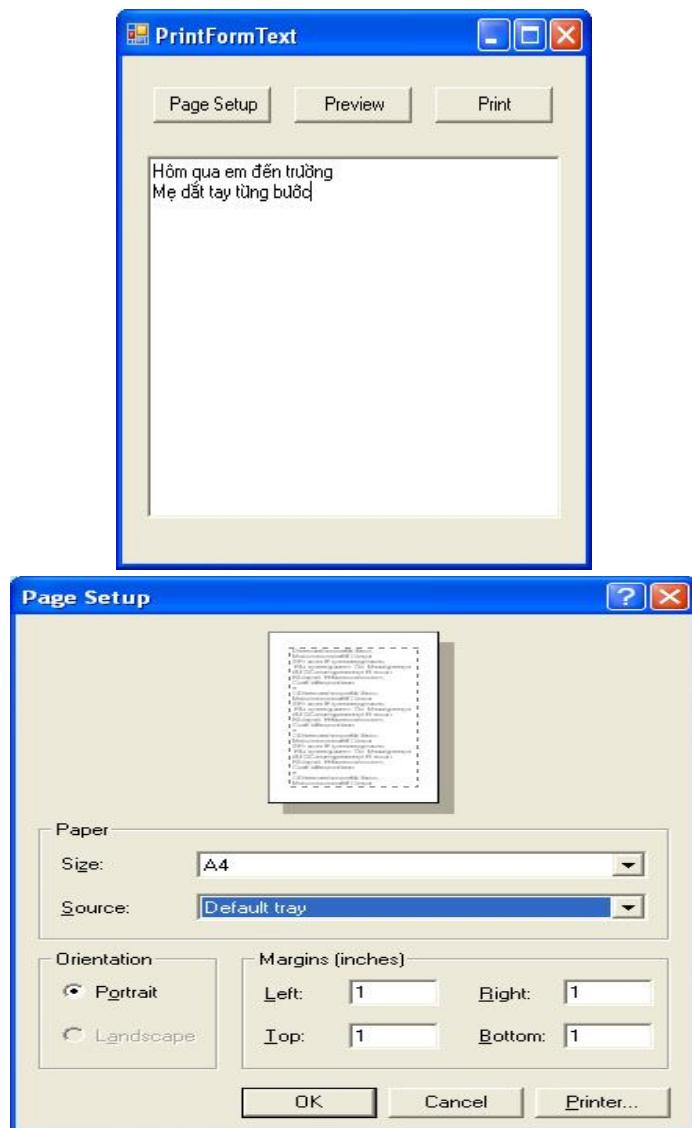
```

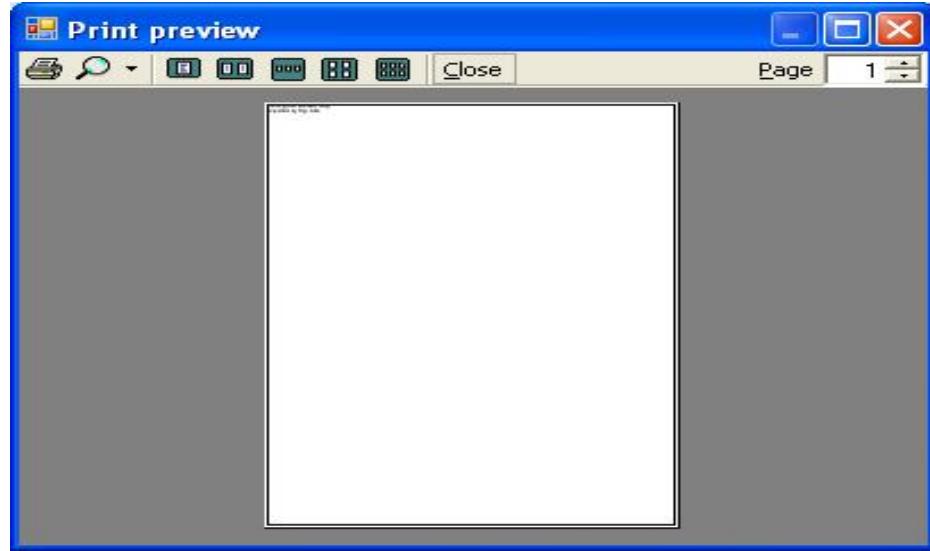
3.6.3. PrintPreviewDialog

Chúng ta đã có thể chọn máy in, đặt thông số in, trang in, phần còn lại là những thao tác để in dữ liệu dựa trên những thông số đã chọn.

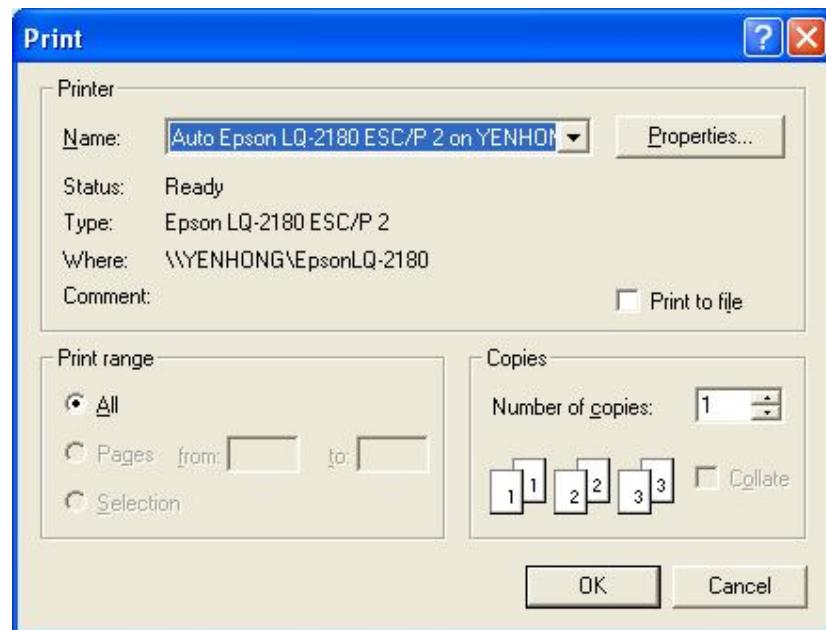
Lớp *PrintPreviewDialog* sẽ hiển thị trang in cho ta xem trước khi in.

Ví dụ 3.7: trình bày chương trình mẫu kết hợp ba lớp *PrintDialog*, *PageSetupDialog* và *PrintPreviewDialog* để in một văn bản đơn giản. Tạo mới một Form và thêm ba Button tên *Button1*, *button2*, *button3*, một textbox tên *textbox1* lên Form. Gõ các đoạn code sau trong sự kiện Click của các Button. Biên dịch và thực thi chương trình, ta gõ một vài dòng văn bản vào textbox. Khi người dùng lần lượt click chuột vào *button1*, *button2*, *button3* chương trình sẽ hiển thị hộp thoại *Page Setup*, *PrintPreview* và *Print*. Sau khi người dùng lựa chọn các thông số in và nhấn nút *OK* trong hộp thoại *Print*, chương trình sẽ in nội dung trong *Textbox* theo những định dạng vừa chọn.





Hình 3.9. PrintPreviewDialog



Hình 3.10. Hộp thoại kết nối với máy In

Source code:

```
using System.Drawing.Printing
Public Class Form1
{
    PrintDialog PrtSetupDB = New PrintDialog();
    Private WithEvents System.Drawing.Printing.PrintDocument
    PrtDocument = New
    System.Drawing.Printing.PrintDocument();
    Private PageSetupDialog PageSetupDB = New
    PageSetupDialog();
```

```

Private PrintPreviewDialog PrintPreviewDB = New
PrintPreviewDialog();
Private System.Drawing.Printing.PrinterSettings
PrinterSettings = New
System.Drawing.Printing.PrinterSettings();
Private void Button1_Click(Object sender EventArgs e)
{
    PageSetupDB.Document = PrtDocument ;
    PageSetupDB.ShowDialog() ;
}
Private void Button2_Click(Object sender, EventArgs e)
{
    PrintPreviewDB.Document = PrtDocument;
    PrintPreviewDB.ShowDialog();
}
Private void Button3_Click(Object sender, EventArgs e)
{
    PrtSetupDB.Document = PrtDocument;
    PrtSetupDB.PrinterSettings = PrinterSettings;
    if (PrtSetupDB.ShowDialog() = DialogResult.OK)
        PrtDocument.Print();
}
Private void PrtDocument_PrintPage(Object sender,
PrintPageEventArgs ev)
{
    ev.Graphics.DrawString(TextBox1.Text, TextBox1.Font,
    Brushes.Black, 0, 0)
}
}

```

Vấn đề chủ yếu cho thao tác PrintPreview và in tài liệu là định nghĩa xử lý sự kiện PrintPage. Khi ta xem trước một trang trong hộp thoại PrintPreview, hộp thoại sẽ sử dụng bộ xử lý sự kiện để vẽ nội dung trang. Tương tự, khi ta gọi phương thức Print của lớp PrintDocument để in tài liệu, chương trình sẽ gọi bộ xử lý sự kiện để thực hiện quá trình in thực tế ra máy in. Trong trường hợp này, nội dung văn bản khá đơn giản, chương trình sử dụng phương thức DrawString để vẽ chuỗi văn bản ra thiết bị in: *ev.Graphics.DrawString(TextBox1.Text, TextBox1.Font);*

BÀI TẬP CHƯƠNG III

1. Nâng cấp ứng dụng Notepad.

- Menu File.
 - + New: Màn hình soạn thảo trống cho phép nhập văn bản mới.
 - + Open: Hiển thị hộp thoại mở File.
 - + Save, Save As...: Hiển thị hộp thoại lưu File.
 - + Print: Hiển thị hộp thoại in ấn.
 - + Exit: Thoát khỏi chương trình.
- Menu Edit gồm có: Cut, Copy, Paste, Delete. Kết hợp với lớp ClipBoard (sinh

viên tự nghiên cứu lớp này trong MSDN) để thực hiện các thao tác di chuyển, sao chép, dán, xóa các khối văn bản.

- Menu Format gồm có: Font dùng để hiển thị hộp thoại chọn Font, cho phép người dùng định dạng kiểu Font, kích thước,

2. Nâng cấp ứng dụng Wordpad.

- Menu file.
 - + New: Tạo màn hình trống, cho phép nhập văn bản mới.
 - + Open: Hiển thị hộp thoại mở File.
 - + Save, Save As....: Hiển thị hộp thoại Lưu File.
 - + Print, Print Preview, Page Setup: Hiển thị các hộp thoại định dạng in ấn tương ứng, in được văn bản ra giấy in.
 - + Exit: Thoát khỏi chương trình.
- Menu Edit: Kết hợp với lớp ClipBoard để thực hiện các thao tác sau:
 - + Cut: Di chuyển khối văn bản vào trong ClipBoard.
 - + Copy: Sao chép khối văn bản vào trong ClipBoard.
 - + Paste: Dán khối văn bản trong ClipBoard vào màn hình soạn thảo.
 - + Clear: Xóa khối văn bản.
 - + Select All: Chọn tất cả văn bản trong màn hình soạn thảo.
- Menu View gồm: ToolBar, Format Bar.
 - + ToolBar: Hiển thị (ẩn) thanh ToolBar.
 - + Format Bar: Hiển thị (ẩn) thanh Format.
- Menu Format gồm: Font, Color.
 - + Font: Hiển thị hộp thoại chọn Font, cho phép người dùng định dạng kiểu Font, kích thước, Style,... cho khối văn bản được chọn.
 - + Color: Hiển thị hộp thoại chọn màu cho phép người dùng định dạng màu cho khối văn bản được chọn.

CHƯƠNG 4. TẬP TIN VÀ THƯ MỤC

Mục tiêu:

Sau khi học xong chương này, sinh viên có khả năng:

- Sử dụng được các lớp, luồng để xử lý tập tin và thư mục: Directory, DirectoryInfo, File, Path, StreamReader, StreamWriter.
- Áp dụng các lớp, luồng trên để hoàn chỉnh ứng dụng Notepad, Wordpad, Window Explorer: Di chuyển, sao chép, tạo File và thư mục.

4.1. Giới thiệu

Để cất giữ thông tin từ một phiên làm việc, ta có thể lưu thông tin trên File và thư mục. Các thao tác chương trình thường tập trung vào việc tạo, lựa chọn, di chuyển và xóa một thư mục; đọc danh sách file của một thư mục hoặc thư mục con, tạo, sao chép, di chuyển và xóa file....Chương này sẽ giúp chúng ta thực hiện các tác vụ này.

Để sử dụng được các thư viện lớp xử lý tập tin và thư mục, chúng ta phải tải thư viện System.IO vào chương trình. Cúp pháp như sau:

```
using System.IO
```

4.2. Lớp Directory

4.2.1. Giới thiệu

Directory là một lớp tĩnh chứa các phương thức xử lý thư mục. Vì là lớp tĩnh nên khi sử dụng các phương thức của lớp này, chúng ta không cần khởi tạo một thê hiện của nó.

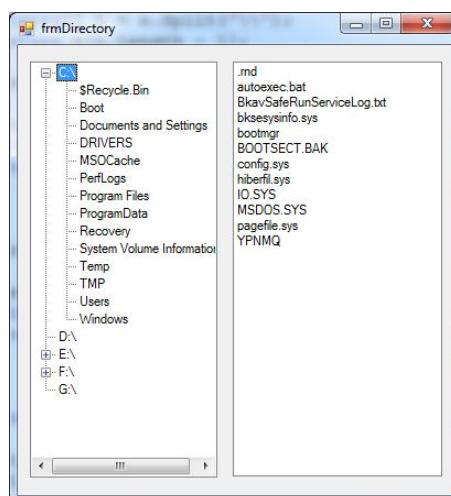
4.2.2. Một số thuộc tính và phương thức cơ bản

- Phương thức CreateDirectory (dirname as String): Tạo ra một thư mục, phương thức trả về giá trị đối tượng kiểu DirectoryInfo (sẽ khảo sát sau) chứa thông tin mô tả thuộc tính của thư mục vừa tạo.
- Phương thức Exists (dirname as String): Trả về giá trị True hoặc False, kiểm tra thư mục dirname có tồn tại hay không.
- Phương thức Delete (dirname as String, del as Boolean): Xóa thư mục dirname. Nếu del = True thì xóa các thư mục và file con của Dirname; ngược lại không thực hiện được. Nếu không có tham số Del thì mặc định là True (xóa thư mục và file con).
- Phương thức move (thư mục 1, thư mục 2): Di chuyển thư mục 1 sang thư mục 2. Ví dụ sau cho phép di chuyển thư mục C:\Temp\Samples sang thư

mục gốc: Directory.Move (“C:\Temp\Sample”, C:\Sample”)

- Phương thức SetCurrentDirectory (dir as String): Thiết lập thư mục dir làm thư mục hiện hành.
- Phương thức GetCurrentDirectory(): Lấy về thư mục hiện hành.
- Phương thức GetFiles (dir as String): Lấy về danh sách các file con của thư mục dir.
- Phương thức GetDirectories (dir as String): Lấy về danh sách các thư mục con của thư mục dir.
- Phương thức GetLogicalDrives(): Trả về danh sách các ổ đĩa logic của hệ thống.
- Phương thức GetCreationTime (dir as String): Trả về thời gian tạo thư mục dir.
- Phương thức GetLastAccessTime (dir as String): Trả về thời gian truy cập thư mục dir lần cuối cùng.
- Phương thức GetLastWriteTime(dir as String): Trả về thời gian chỉnh sửa dir lần cuối cùng.
- Phương thức GetParent (dir as String) as DirectoryInfo: Lấy thông tin thư mục cha của thư mục dir. Phương thức trả về là đối tượng DirectoryInfo.
- Phương thức GetDirectoryRoot (dir as String): Trả về thư mục gốc của thư mục dir.

Ví dụ 4.1: minh họa một số thuộc tính và phương thức của lớp Directory. Chương trình bao gồm một treeview (tr) và một listview. Mặc định, chương trình sẽ hiển thị các ổ đĩa hệ thống trên treeview, khi người dùng chọn vào ổ đĩa hoặc thư mục trên treeview, các thư mục con sẽ hiển thị tiếp trong treeview, các tập tin con sẽ hiển thị bên listview (tương tự như ứng dụng Windows Explorer).



Hình 4.1 Minh họa Directory

Source code:

```
void ht_dr()
{
    string[] dr = Directory.GetLogicalDrives();
    for (int i = 0; i < dr.Length; ++i)
        tr.Nodes.Add(dr[i]);
}
private void frmDirectory_Load(object sender, EventArgs e)
{
    ht_dr();
}
string layten(string s) //lấy tên thư mục từ đường dẫn tuyệt
đối
{
    string[] t = s.Split('\\');
    return t[t.Length - 1];
}
void ht_dir(TreeNode tn)
{
    tn.Nodes.Clear();
    string[] d = Directory.GetDirectories(tn.FullPath);
    for (int i = 0; i < d.Length; ++i)
        tr.SelectedNode.Nodes.Add(layten(d[i]));
}
void ht_file(TreeNode tn)
{
    lsv.Items.Clear();
    string[] f = Directory.GetFiles(tn.FullPath);
    for (int i = 0; i < f.Length; ++i)
        lsv.Items.Add(layten(f[i])); //thiết lập thuộc tính view
        của lsv là List
}
private void tr_AfterSelect(object sender, TreeViewEventArgs e)
{
    ht_dir(tr.SelectedNode);
    ht_file(tr.SelectedNode);
}
```

4.3. Lớp DirectoryInfo

4.3.1. Giới thiệu

Trong quá trình chương trình của chúng ta thao tác xử lý thư mục sẽ có lúc bạn cần biết thông tin về thư mục (như ngày tháng và thời gian cập nhật sau cùng...). Để giúp chương trình xác định những thuộc tính thư mục, bên cạnh sự hỗ trợ của lớp tĩnh Directory, C# còn hỗ trợ lớp DirectoryInfo cung cấp các thuộc tính cũng như phương thức mà ta có thể sử dụng để lấy hoặc đặt thuộc tính cho thư mục.

4.3.2. Một số thuộc tính và phương thức cơ bản

- Khai báo và khởi tạo: DirectoryInfo dir = new DirectoryInfo (String dirname).

- Thuộc tính Fullname: Trả về tên đầy đủ của dirname.
- Thuộc tính CreationTime: Trả về hoặc khởi tạo thời gian tạo dirname.
- Thuộc tính LastAccessTime: Trả về hoặc khởi tạo thời gian truy cập dirname lần cuối cùng.
- Thuộc tính LastWriteTime: Trả về hoặc khởi tạo thời gian chỉnh sửa dirname lần cuối cùng.
- Bên cạnh lớp Directory hỗ trợ việc lấy file và thư mục con. Lớp DirectoryInfo cũng hỗ trợ hai phương thức Getfiles và GetDirectories để lấy về file và thư mục con.

4.4. Lớp File (tập tin)

4.4.1. Giới thiệu

Tùy thuộc vào sự xử lý của chương trình, sẽ có lúc ta muốn thực hiện việc xử lý đặc biệt dựa vào những thuộc tính file. Chẳng hạn đọc tất cả những file vừa được tạo cách đây 2 ngày,... C# hỗ trợ lớp File (lớp tĩnh) để ta thực hiện những công việc trên.

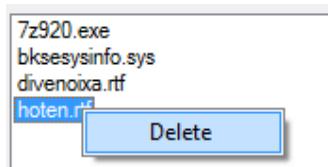
4.4.2. Một số phương thức cơ bản

- Phương thức Delete (String filename): xoá filename.
- Phương thức Move (String oldname, String newname): di chuyển file oldname đến thư mục newname.
- Phương thức Copy (String source, String Destination): sao chép file source sang thư mục Destination.
- Phương thức GetCreationTime(String filename): trả về thời gian (DateTime) tạo ra filename.
- Phương thức GetLastAccessTime(String filename): trả về thời gian (DateTime) truy cập filename lần cuối cùng.
- Phương thức GetLastWriteTime (String filename): trả về thời gian (DateTime) ghi sửa filename lần cuối cùng.
- Phương thức SetCreationTime(String filename, DateTime time): thiết lập thời gian tạo File là Time.
- Phương thức SetLastAccessTime(String filename, DateTime time): thiết lập thời gian truy cập file lần cuối là time.
- Phương thức SetLastWriteTime (String filename, DateTime time): thiết lập thời gian ghi sửa file lần cuối là time.

□Chú ý: ngoài lớp File, ta có thể sử dụng lớp FileInfo để hiển thị các thuộc tính file (tham khảo thêm trong MSDN).

Ví dụ 4.2: minh họa phương thức lớp Delete của lớp File. Chương trình kế thừa ví dụ 4.2.1, khi người dùng nhấn chuột phải vào listview, menu ngữ cảnh sẽ hiển thị

chức năng Delete, khi chọn Delete chương trình sẽ xóa tập tin vừa chọn. Chúng ta bổ sung control contextmenu và gắn vào listview.



Hình 4.2 Minh họa phương thức Delete của lớp File

Source code:

```
private void cmnDel_Click(object sender, EventArgs e)
{
    File.Delete(tr.SelectedNode.FullPath + "\\\" +
    lsv.SelectedItems[0].Text );
}
```

4.5. Lớp Path (Đường dẫn)

4.5.1. Giới thiệu

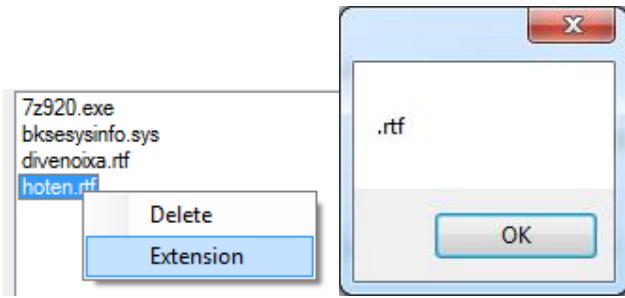
Khi chúng ta làm việc với file và thư mục sẽ có lúc ta cần phân tích thông tin đường dẫn. Ví dụ, nếu chương trình yêu cầu người dùng nhập vào tên file, ta có thể trước hết phân tích dữ liệu người dùng nhập vào để xác định xem người dùng chỉ rõ một tên đường dẫn hay chưa, kiểm tra tên mở rộng, tên file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

Để thao tác xử lý thông tin đường dẫn thư mục, ta có thể sử dụng lớp Path. Lớp Path là một lớp tĩnh cung cấp các phương thức cho phép phân tích đường dẫn.

4.5.2. Một số phương thức cơ bản

- Phương thức GetDirectoryName (String filename): lấy về đường dẫn filename nhưng không có tên file.
- Phương thức GetExtension (String filename): lấy về phần mở rộng của filename.
- Phương thức Getfilename (String filename): Lấy về tên filename.
- Phương thức thwcsGetFilenameWithoutExtension (String filename): lấy về tên file nhưng không có phần mở rộng.
- Phương thức GetFullPath (String filename): Lấy về đường dẫn đầy đủ của filename.
- Phương thức GetPathRoot (String filename): Lấy về thư mục gốc của filename.

Ví dụ 4.2.3: minh họa cách sử dụng phương thức GetExtension của lớp File. Ví dụ này kế thừa ví dụ 4.2.1. Khi người dùng nhấn chuột phải vào tập tin trên listview, chương trình sẽ hiển thị menu người cảnh báo phép chọn Extension.



Hình 4.3 Minh họa phương thức GetExtension

Source code:

```
private void cmnExt_Click(object sender, EventArgs e)
{
    MessageBox.Show(
        Path.GetExtension(lsv.SelectedItems[0].Text));
}
```

4.6. Sử dụng luồng với FileStream

4.6.1. Giới thiệu

File chứa dữ liệu bất kể đó là dữ liệu gì đơn giản chúng được xem như một chuỗi các byte. Một khái niệm khác để chỉ dữ liệu của file là luồng (hay Stream). Ta có thể mở một Stream, đọc dữ liệu và sau đó đóng stream lại.

Để đọc thông tin những file văn bản (text file), sử dụng những phương thức của lớp File. Một chương trình có thể đọc dữ liệu vào Stream, đưa dữ liệu vào bộ đệm và gửi đi trên mạng. Để đơn giản hóa những thao tác xử lý file văn bản, ta có thể sử dụng các lớp StreamWriter và StreamReader để ghi và đọc dữ liệu từ luồng.

Sau khi kết thúc chương này, kết hợp với các chương trước Winform, Common Dialog, chúng ta có thể dễ dàng tạo được ứng dụng tương tự Notepad, Wordpad.

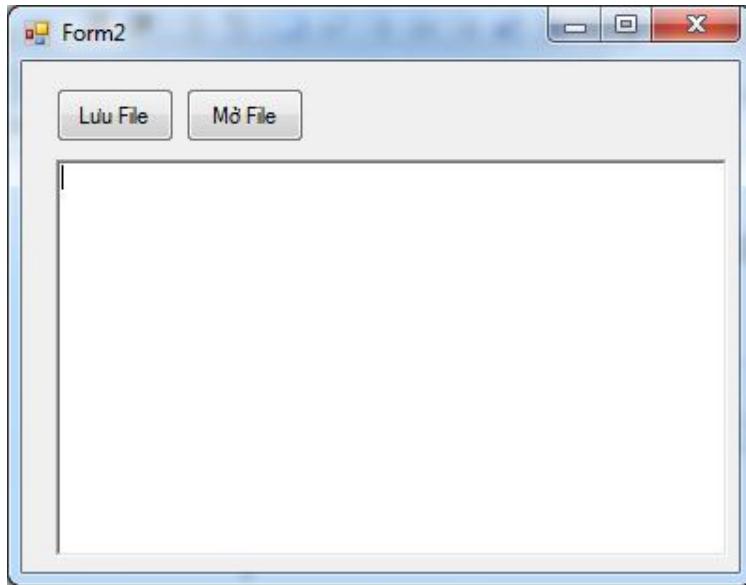
4.6.2. StreamWriter: Ghi dữ liệu vào File

- Khai báo và khởi tạo: StreamWriter Targetfile = new StreamWriter (String filename).
- Phương thức write (String text): ghi text vào filename.
- Phương thức Close(): đóng file.

4.6.3. StreamReader: Đọc file từ nguồn

- Khai báo và khởi tạo: StreamReader sourcefile = new StreamReader (String filename).
- Phương thức Readline(): đọc từng dòng dữ liệu text trong filename.
- Phương thức ReadToEnd(): trả về String bao gồm toàn bộ nội dung của filename.
- Phương thức Close(): đóng filename.

Ví dụ 4.2.4: minh họa cách lưu và mở file, file này được lưu trong cùng thư mục ứng dụng (trong thư mục Debug). Ví dụ gồm một RichTextbox (tên là rtf) và hai Button Lưu và Mở File



Hình 4.4 Giao diện ví dụ lưu – mở File

Lưu File:

```
StreamWriter f = new StreamWriter("baitap.rtf");
f.WriteLine(rtf.Text);
f.Close();
```

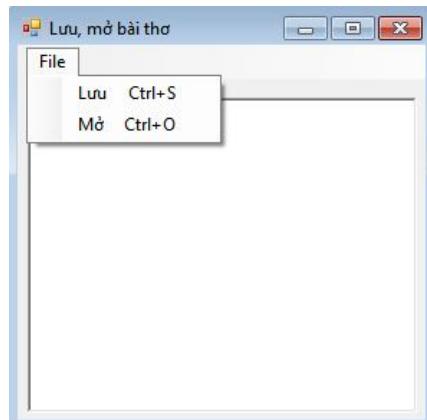
Mở File:

```
StreamReader f = new StreamReader("baitap.rtf");
rtf.Text = f.ReadToEnd();
f.Close();
```

BÀI TẬP CHƯƠNG IV

1. Thiết kế Form lưu, mở bài thơ

a. Giao diện

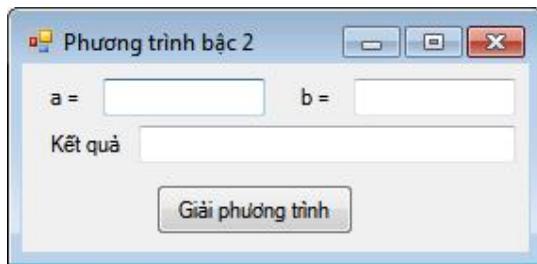


b. Yêu cầu

- Lưu File: Lưu bài thơ vào File
- Mở File: Mở bài thơ đã lưu trong File

2. Thiết kế Form giải, lưu Phương trình bậc 2

a. Giao diện

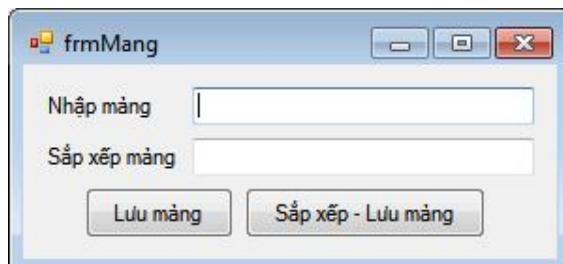


b. Yêu cầu: chọn nút “Giải phương trình”:

- Hiển thị kết quả lên textbox kết quả
- Lưu vào file như sau: 3 hệ số trên một dòng, cách nhau phím Tab
- Kết quả lưu ở dòng thứ hai

3. Thiết kế Form lưu mảng

a. Giao diện

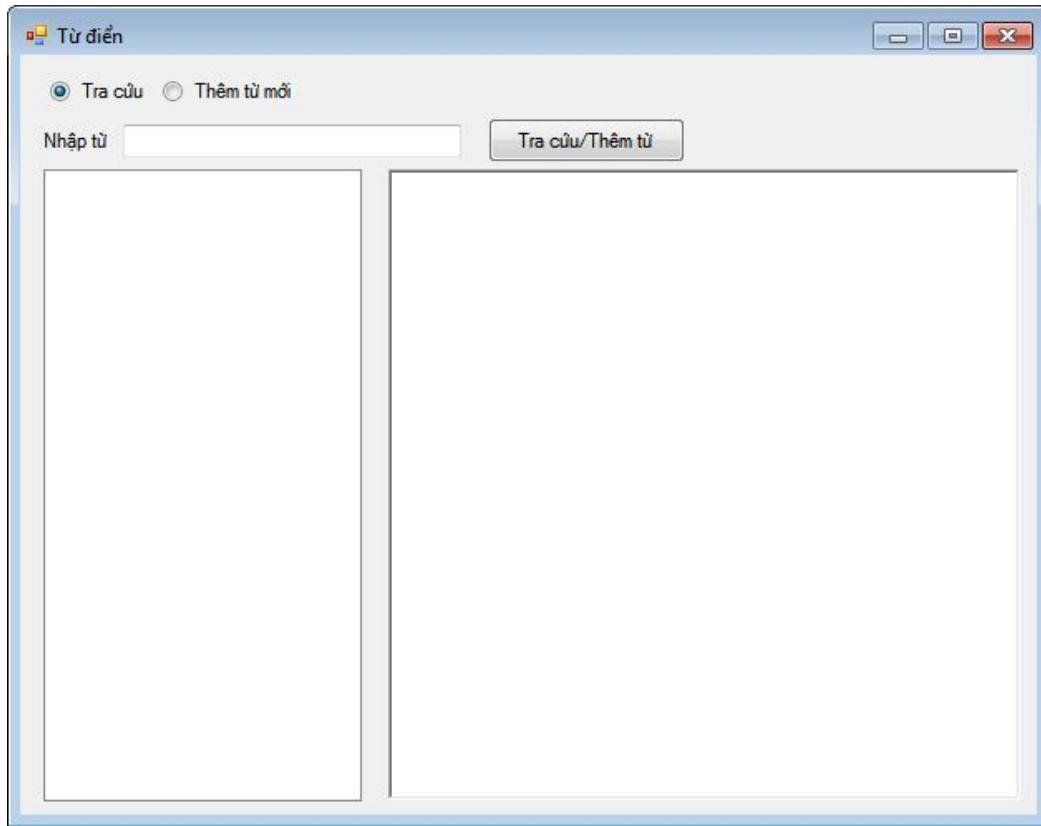


b. Yêu cầu

- Lưu mảng: Lưu mảng vừa nhập vào dòng đầu tiên của File
- Sắp xếp và lưu mảng: Sắp xếp mảng, hiển thị lên textbox sắp xếp, lưu mảng sắp xếp vào file ở dòng tiếp theo

4. Thiết kế Form minh họa từ điển.

a. Giao diện



b. Yêu cầu

- Formload: hiển thị các từ tiếng Anh được sắp xếp thứ tự
- Nếu radio “Tra cứu” đang chọn, gõ từ trên textbox
 - + Hiển thị dấu chọn lên listbox với từ gần giống nhất với từ đang gõ
 - + Nhấn phím Enter: dịch từ đang chọn lên textbox bên phải
- Nếu radio “Thêm từ” đang chọn:
 - + Textbox “từ”: gõ từ tiếng Anh
 - + Textbox “kết quả”: gõ nghĩa tiếng Việt
 - + Nút “thêm mới”: lưu từ tiếng Anh và nghĩa tiếng Việt vào File.

BÀI TẬP NÂNG CAO

1. Hoàn thiện ứng dụng NotePad: xây dựng được các chức năng sau:
 - a. Mở được tập tin dạng TXT và hiển thị trên màn hình soạn thảo.
 - b. Lưu nội dung trên màn hình soạn thảo vào tập tin.
2. Hoàn thiện ứng dụng WordPad: xây dựng được các chức năng sau:
 - a. Mở được tập tin dạng RTF và hiển thị trên màn hình soạn thảo
 - b. Lưu nội dung trên màn hình soạn thảo xuống File dạng RTF
3. Hoàn thiện ứng dụng Windows Explorer: xây dựng các chức năng sau:
 - a. Xác định các ổ đĩa logic trong máy tính.

- b. Hiển thị được các tập tin và thư mục con tương ứng khi người dùng click chọn vào thư mục cha.
- c. Di chuyển, sao chép tập tin, thư mục.
- d. Tạo mới, xóa, đổi tên tập tin, thư mục

CHƯƠNG 5. LẬP TRÌNH CƠ SỞ DỮ LIỆU VỚI ADO.NET

Mục tiêu:

Sau khi học xong chương này, sinh viên có khả năng:

- Liệt kê được các thành phần chính trong kiến trúc của ADO.NET.
- Nêu được các ưu điểm của ADO.NET và những điểm khác so với ADO.
- Mô tả được chức năng của các đối tượng (lớp) Connection, DataAdapter, DataCommand, DataSet, DataReader, DataTable của ADO.NET.
- Sử dụng được các Provider cho từng hệ cơ sở dữ liệu: MS Access, SQL Server 2000, SQL Server 2005, SQL Server 2008, ...

5.1. Giới thiệu ADO.Net

5.1.1. Giới thiệu

ADO.NET là một trong 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.

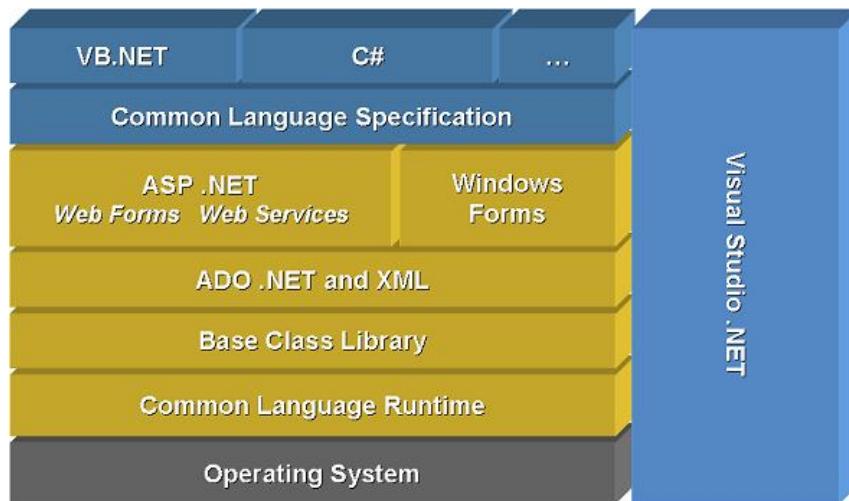
ADO.NET có các đặc điểm sau:

- Là một bộ các thư viện hướng đối tượng (OOP) cho phép tương tác với dữ liệu nguồn. Thông thường thì dữ liệu nguồn là một cơ sở dữ liệu (database), nhưng cũng có thể là tập tin text, excel hoặc XML.
- Làm việc trên 2 namespace cơ sở dữ liệu là System.data.sqlClient và System.data.olebd, một cho SQLServer, và một cho các cơ sở dữ liệu được trình bày thông qua một giao diện OLEDB. Nếu cơ sở dữ liệu được chọn là một bộ phận của OLE DB, bạn có thể dễ dàng kết nối với nó từ .NET, chỉ cần dùng các lớp OLE DB và kết nối thông qua các driver cơ sở dữ liệu hiện hành của bạn.
- Là một tập các lớp nằm trong bộ thư viện lớp cơ sở của .NET Framework, cho phép các ứng dụng windows (như C#, VB.NET) hay ứng dụng Web (như ASP.NET) thao tác dễ dàng với các nguồn dữ liệu.
- Được thiết kế với dạng dữ liệu “ngắt kết nối”, nghĩa là chúng ta có thể lấy cả một cấu trúc phức tạp của dữ liệu từ database, sau đó ngắt kết nối với database rồi mới thực hiện các thao tác cần thiết. Đây là một sự tiến bộ về mặt thiết kế bởi vì thiết kế ADO trước đây luôn cần duy trì một kết nối trong quá trình thao tác dữ liệu.
- Được thiết kế nhằm tăng tốc độ truy cập và thao tác dữ liệu trong môi trường đa lớp (đa tầng). Hai thành phần chính của ADO.NET là đối tượng Dataset (thuộc lớp không kết nối) và trình điều khiển cơ sở dữ liệu .NET thuộc lớp kết nối.

- Là thành phần nội tại (có sẵn) trong .NET Framework, do vậy dễ dàng khi phát triển bằng nhiều ngôn ngữ khác nhau.

Mục tiêu của ADO.NET là:

- Cung cấp các lớp để thao tác CSDL trong cả hai môi trường là phi kết nối (Disconnected data) và kết nối (Connected data).
- Tích hợp chặt chẽ với XML (Extensible Markup Language).
- Tương tác với nhiều nguồn dữ liệu thông qua mô tả dữ liệu chung.
- Tối ưu truy cập nguồn dữ liệu (OLE DB & SQL server).
- Làm việc trên môi trường Internet (môi trường phi kết nối).



Hình 5.1 Vị trí của ADO.NET trong .NET Framework

5.1.2. Sự khác nhau giữa ADO.NET và ADO

Trước ADO.NET, Microsoft đã có là một bộ thư viện để xử lý các thao tác liên quan đến dữ liệu, có tính linh hoạt, dễ sử dụng và được tích hợp trong các ngôn ngữ như Visual Basic, ASP 3.0 đó là ADO.

Có thể coi ADO.NET là một thế hệ tiếp theo của ADO kế thừa tất cả những ưu điểm của, đồng thời với ý tưởng thiết kế hoàn toàn mới ADO.NET có một diện mạo khác hẳn so với tiền thân của nó. Một vài đặc điểm nổi bật của ADO.NET mà ADO không có như sau:

- ADO.NET được thiết kế hoàn toàn dựa vào XML vì XML là chuẩn trao đổi dữ liệu tiên bộ và tốt nhất trên môi trường Internet hiện nay.
- ADO.NET được thiết kế hoàn toàn hướng đối tượng: đây là đặc điểm chi phối toàn bộ các sản phẩm Microsoft .NET.
- Hai đặc điểm trên là hai đặc điểm cơ bản, và nổi trội của ADO.NET mà ADO không có. Nay giờ chúng ta sẽ so sánh chi tiết hơn về từng khía cạnh của ADO và ADO.NET.

Bảng 5.1 So sánh đặc điểm ADO và ADO.NET

Đặc điểm	ADO	ADO.NET
Dữ liệu xử lý được đưa vào bộ nhớ.	Recordset: Tương đương một bảng dữ liệu trong database.	Dataset: tương đương một database.
Duyệt dữ liệu	Recordset chỉ cho phép duyệt tuần tự, từng dòng một.	Dataset: cho phép duyệt “tự do, ngẫu nhiên”, truy cập thẳng tới bảng, dòng, cột mong muốn.
Dữ liệu ngắt kết nối.	Recordset cũng có thể ngắt kết nối nhưng tư tưởng thiết kế ban đầu của Recordset là hướng kết nối, do đó việc ngắt kết nối cũng không được hỗ trợ tốt nhất.	Dataset được thiết kế với tư tưởng ban đầu là “ngắt kết nối” và hỗ trợ mạnh mẽ việc “ngắt kết nối”.
Khả năng vượt tường lửa.	Khi trao đổi dữ liệu với qua Internet, thường sử dụng chuẩn COM, chuẩn COM rất khó vượt qua được tường lửa. Do vậy khả năng trao đổi dữ liệu ADO qua Internet thường có nhiều hạn chế.	ADO.NET trao đổi dữ liệu qua Internet rất dễ dàng vì ADO.NET được thiết kế theo chuẩn XML, là chuẩn dữ liệu chính được sử dụng để trao đổi trên Internet.
Lập trình.	Sử dụng đối tượng Connection để thực hiện chuyển giao câu lệnh.	Sử dụng đặc tính của XML, tự mô tả dữ liệu, đọc và ghi dữ liệu dễ dàng.

ADO vẫn tồn tại song song với ADO.NET. Mặc dù các ứng dụng mới nhất của .NET được viết bằng cách sử dụng ADO.NET, ADO vẫn còn có sẵn cho các lập trình viên .NET thông qua chuẩn COM.

5.1.3. Các trình điều khiển cơ sở dữ liệu trong ADO.NET

Khi lấy dữ liệu từ nguồn dữ liệu (dữ liệu quan hệ, tập tin văn bản, thông tin trong email,...), ta cần quản lý và sử dụng trình điều khiển (provider).

.NET Data Providers là một tập các đối tượng phục vụ cho việc trao đổi dữ liệu giữa Data Source (dữ liệu nguồn) và đối tượng DataSet. Nó chia ra gồm 2 loại tập đối tượng: một tập các đối tượng chịu trách nhiệm quản lý các kết nối (connections) tới DataSource (dữ liệu nguồn) và một tập các đối tượng còn lại chịu trách nhiệm xử lý dữ liệu.

.NET Framework hỗ trợ hai trình cung cấp dữ liệu là SQL Server .NET Data Provider (dành cho phiên bản SQL Server 7.0 của Microsoft trở lên) và OLEDB .NET Data Provider (dành cho các hệ quản trị cơ sở dữ liệu khác) để truy cập vào cơ sở dữ liệu.

Để sử dụng 2 loại .NET Data Provider này chúng ta phải import hai Namespace: System.Data.SqlClient và System.Data.OleDb cho SQL Server .NET Data Provider và OLEDB Data Provider.

SQL Server .NET Data Provider có các đặc điểm:

- Dùng ngữ thức riêng để truy cập cơ sở dữ liệu.
- Truy xuất dữ liệu sẽ nhanh hơn và hiệu quả hơn do không phải thông qua lớp OLEDB Provider hay ODBC.
- Chỉ được dùng với hệ quản trị cơ sở dữ liệu SQL Server 7.0 trở lên.
- Được Microsoft hỗ trợ khá hoàn chỉnh.

OLE DB .NET Data Provider có các đặc điểm:

- Phải thông qua 2 lớp vì thế sẽ chậm hơn.
- Thực hiện được các dịch vụ “Connection Pool”.
- Có thể truy cập vào mọi Datasource có hỗ trợ OLEDB Provider thích hợp.

Mặc dù, hai trình điều khiển trên thuộc hai namespace khác nhau nhưng chúng cung cấp các chức năng tương tự nhau khi làm việc với cơ sở dữ liệu.

Chúng ta sẽ tham khảo sự tương ứng các đối tượng thuộc hai trình điều khiển SQL Server .NET Data Provider và OLEDB.NET Data Provider như sau:

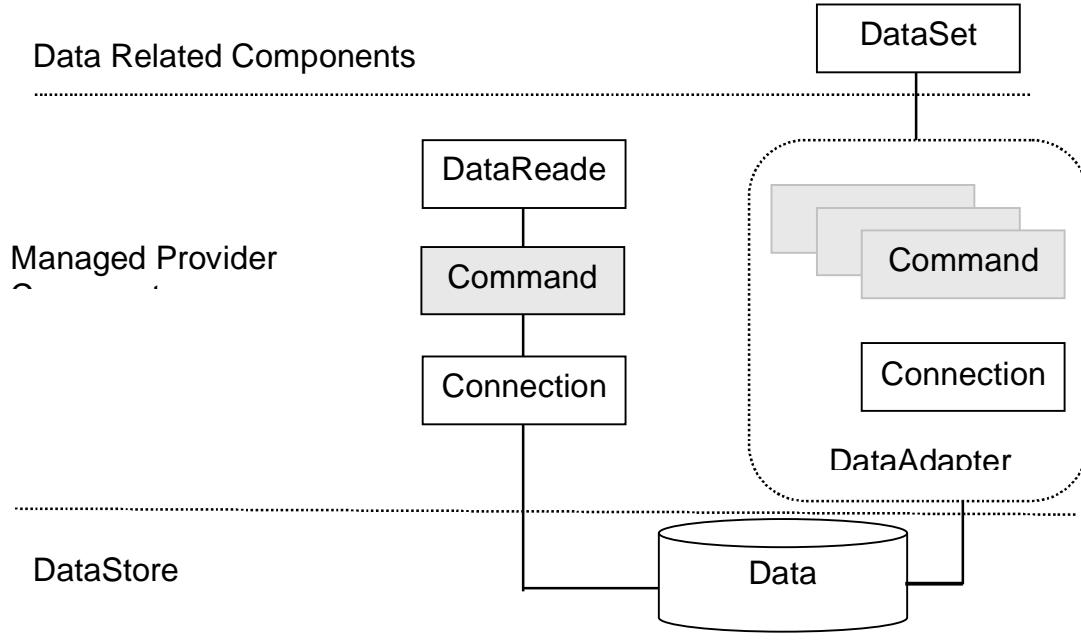
Bảng 5.2 Các loại trình điều khiển của SQLServer và OLEDB

SQL Server .NET Data Provider	OleDb .NET Data Provider
SqlConnection.	OleDbConnection.
SqlCommand.	OleDbCommand.
SqlDataAdapter.	OleDbDataAdapter.
SqlDataReader.	OleDbDataReader.
SqlParameter.	OleDbParameter.

ADO.NET bao gồm nhiều đối tượng có thể dùng với dữ liệu. Chương này giới thiệu một vài đối tượng chính. Trong những phần sau chúng ta sẽ khám phá thêm nhiều đối tượng ADO.NET và cách chúng được sử dụng trong mỗi trường hợp riêng.

Các thành phần chính của ADO.NET:

- Connection
- Command
- Datareader
- DataAdapter
- DataSet



Các tầng kiến trúc của ADO.NET

SqlConnection

Để tương tác với database, bạn phải có một kết nối tới nó. Kết nối giúp xác định database server, database name, user name, password, và các tham số cần thiết để kết nối tới database. Một đối tượng connection được dùng bởi đối tượng command vì thế chúng sẽ biết database nào để thực thi lệnh.

SqlCommand

Quá trình tương tác với database cần phải biết hành động nào chúng ta muốn xảy ra. Điều này được thực hiện bởi đối tượng command. Chúng ta dùng đối tượng command để gửi một câu lệnh SQL tới database. Một đối tượng command dùng một đối tượng connection để xác định database nào sẽ được truy xuất. Bạn có thể dùng một đối tượng command riêng lẻ để thực thi lệnh trực tiếp, hoặc để gắn một tham chiếu của đối tượng command cho một SqlDataAdapter – đối tượng giữ các command sẽ làm việc trên một nhóm dữ liệu như sẽ đề cập tới trong phần dưới.

SqlDataReader

Nhiều thao tác dữ liệu đòi hỏi bạn chỉ lấy một luồng dữ liệu để đọc. Đối tượng data reader cho phép bạn lấy được kết quả của một câu lệnh SELECT từ một đối tượng command. Để tăng hiệu suất, dữ liệu trả về từ một data reader là một luồng dữ liệu fast forward-only. Có nghĩa là bạn chỉ có thể lấy dữ liệu từ luồng theo một thứ tự nhất định. Mặc dù điều này có lợi về mặt tốc độ, nhưng nếu bạn cần phải thao tác dữ liệu, thì một DataSet sẽ là một đối tượng tốt hơn để làm việc.

DataSet

Đối tượng DataSet là một thể hiện của dữ liệu trong bộ nhớ. Chúng chứa nhiều đối tượng DataTable, bên trong DataTable lại có nhiều column và row, giống như các

database table thông thường. Bạn thậm chí có thể định nghĩa dữ liệu giữa các table để tạo các quan hệ parent-child. DataSet được thiết kế đặc biệt để giúp quản lý dữ liệu trong bộ nhớ và để hỗ trợ các thao tác không cần kết nối (disconnected) trên dữ liệu. DataSet là một đối tượng được dùng bởi tất cả Data Provider, đó là lý do tại sao nó không có một Data Provider prefix trong tên gọi.

SqlDataAdapter

Đôi lúc dữ liệu mà bạn làm việc là read-only và bạn ít khi cần thay đổi dữ liệu nguồn. Vài trường hợp cần lưu trữ tạm dữ liệu trong bộ nhớ để hạn chế truy xuất đến database. Data adapter làm điều này dễ dàng bằng cách giúp bạn quản lý dữ liệu trong chế độ ngắt kết nối. Data adapter sẽ đổ vào DataSet khi đọc dữ liệu và thực hiện thay đổi dữ liệu một lượt vào database.

DataAdapter chứa một tham chiếu đến đối tượng connection và mở/đóng kết nối tự động khi đọc và ghi dữ liệu vào database. Hơn nữa, data adapter chứa đối tượng command cho những thao tác SELECT, INSERT, UPDATE và DELETE trên dữ liệu. Bạn sẽ có một data adapter được định nghĩa cho mỗi table trong một DataSet và nó sẽ quản lý các giao tiếp với database cho bạn. Tất cả những gì bạn cần làm là chỉ cho data adapter khi nào nạp hoặc ghi vào database.

ADO.NET là một kỹ thuật .NET để thao tác với nguồn dữ liệu. Chúng ta có một vài Data Provider, cho phép giao tiếp với các nguồn dữ liệu khác nhau, dựa trên giao thức mà chúng dùng hoặc kiểu database. Không cần quan tâm đến điều này, với mỗi Data Provider được sử dụng, bạn sẽ dùng các đối tượng tương tự nhau để thao tác với dữ liệu. Đối tượng SqlConnection cho phép bạn quản lý một kết nối đến nguồn dữ liệu. SqlCommand cho phép bạn gửi lệnh đến dữ liệu. Để đọc dữ liệu nhanh theo cơ chế forward-only, sử dụng SqlDataReader. Nếu bạn muốn làm việc với dữ liệu đã ngắt kết nối, dùng một DataSet và hiện thực việc đọc và ghi đến dữ liệu nguồn bằng một SqlDataAdapter.

5.2. Đối tượng Connection

Đối tượng Connection dùng để tạo một kết nối giữa ứng dụng với cơ sở dữ liệu. Kết nối đến cơ sở dữ liệu là thao tác đầu tiên cần phải thực hiện trước khi muốn truy vấn hay cập nhật dữ liệu (thêm, sửa, xóa).



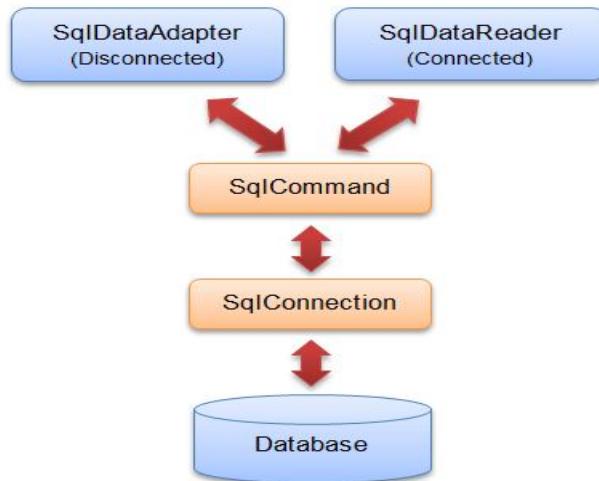
Hình 5.2. Connection.

Nếu dùng Data Provider SQL Server thì phải khởi tạo đối tượng của lớp SqlConnection, khi đó phải khai báo namespace System.Data.SqlClient. Còn nếu dùng Data Provider for OLEDB thì phải khởi tạo đối tượng OleDbConnection, khi đó phải

khai báo namespace: System.Data.OleDb.

5.2.1. SqlConnection

Mô hình sau cho ta thấy vị trí của SqlConnection tương tác với các thành phần khác trong ADO.NET:



Hình 5.3. Vị trí của SqlConnection.

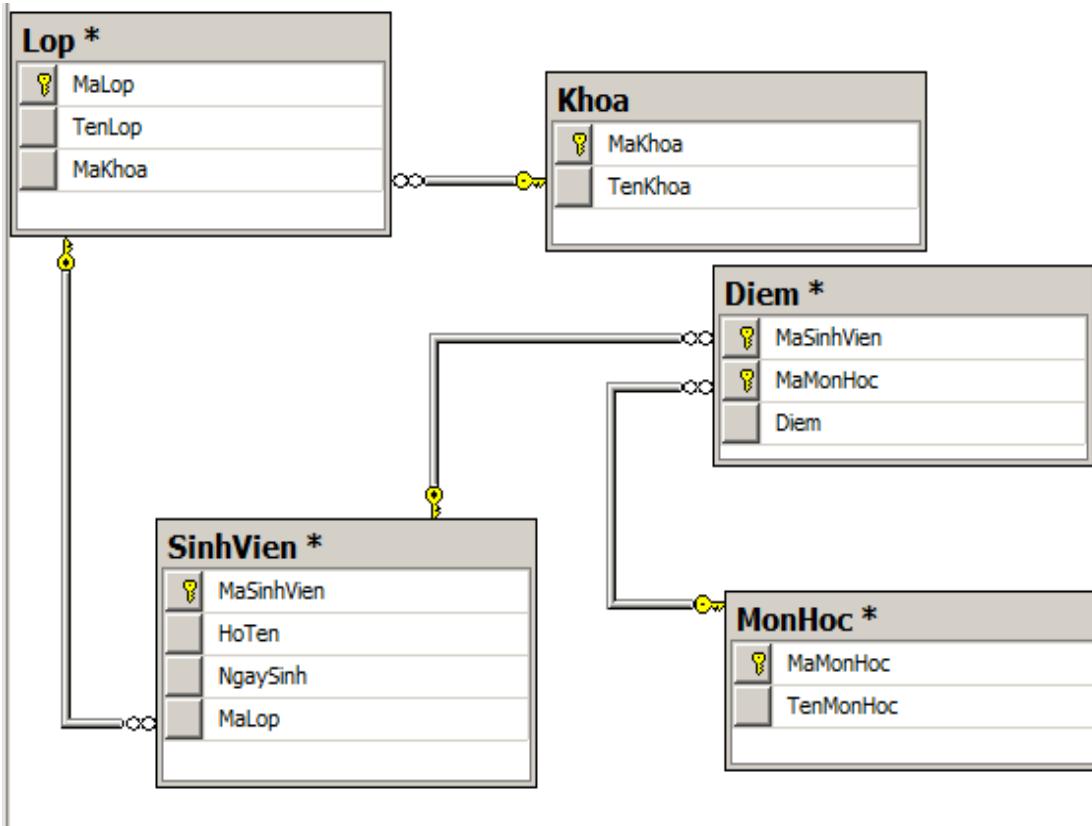
Tạo một đối tượng SqlConnection như sau:

Một đối tượng SqlConnection giống như các đối tượng khác trong C#. Chúng ta chỉ cần khai báo một thê hiện của SqlConnection, như dưới đây:

```
// Create Connection  
SqlConnection conn = new SqlConnection();
```

☞ **Lưu ý:** để tiện sử dụng các ví dụ cho chương này và chương sau, chúng tôi chỉ sử dụng một cơ sở dữ liệu quản lý sinh viên (qlsv) duy nhất bao gồm các table sau:

- Khoa (MaKhoa, TenKhoa)
- Lop (MaLop, TenLop, MaKhoa)
- SinhVien (MaSV, HoTen, NgaySinh, MaLop)
- MonHoc (MaMonHoc, TenMonHoc)
- Diem (MaSV, MaMonHoc, Diem)



Đối tượng SqlConnection trên sử dụng constructor với một tham số kiểu string. Tham số này được gọi là chuỗi kết nối (connection string). Các giá trị của nó được mô tả như sau:

Tên chuỗi tham số kết nối	Mô tả
Data Source	Xác định máy chủ. Có thể là máy cục bộ, tên miền máy, hoặc địa chỉ IP.
Initial Catalog	Tên Database.
Integrated Security	Thiết lập để SSPI để làm cho kết nối với những người sử dụng đăng nhập vào Windows.
User ID	Tên người dùng cấu hình trong SQL Server.
Password	Mật khẩu phù hợp với SQL Server User ID.

Integrated Security sẽ bảo mật khi bạn làm việc trên một máy đơn. Tuy nhiên, bạn sẽ thường xuyên cần phải định rõ mức bảo mật dựa trên SQL Server User ID với quyền hạn được xác định cho ứng dụng bạn sử dụng.

Ví dụ 5.1: kết nối cơ sở dữ liệu bằng quyền windows và Sqld

```
// Create Connection
SqlConnection conn = new SqlConnection("Data
Source=User\\SQLEXPRESS;Initial Catalog=QLSINHVIEN;User
ID=sa;Password=sa2008");
```

Ngoài ra chúng ta còn có thể kết nối bằng quyền đăng nhập của User Windows như sau:

```
// Create Connection
SqlConnection conn = new SqlConnection("Data
Source=User\\SQLEXPRESS;Initial
Catalog=QLSINHVIEN;Integrated Security=True");
```

Lưu ý rằng Data Source được gán cho DatabaseServer để chỉ ra rằng chúng ta có thể định danh một database trên một máy khác, thông qua mạng LAN, hoặc qua Internet. Ngoài ra, User ID và Password được thay thế cho tham số Integrated Security.

Mục đích của việc tạo một đối tượng SqlConnection là để các mã lệnh ADO.NET khác có thể làm việc được với database. Các đối tượng ADO.NET khác, như SqlCommand và SqlDataAdapter dùng một connection như một tham số. Quá trình sử dụng SqlConnection gồm các bước sau:

- Tạo một SqlConnection.
- Mở connection.
- Truyền connection cho các đối tượng ADO.NET khác.
- Thực hiện các thao tác database với các đối tượng ADO.NET này.
- Đóng connection.

Một số thuộc tính của connection

Thuộc tính	Mô tả
State	Tình trạng kết nối của Connection với các giá trị.
Broken	Kết nối với nguồn dữ liệu đã bị ngắt. Tình trạng này chỉ xảy ra sau khi đã kết nối
Closed	Kết nối đã đóng.
Connecting	Đang kết nối với nguồn dữ liệu.
Executing	Kết nối đang thực hiện một lệnh.
Fetching	Kết nối đang truy xuất dữ liệu.
Open	Kết nối đang mở.

5.2.2. OleDbConnection

Tương tự như SqlConnection đối tượng OleDbConnection chúng ta chỉ cần thay đổi Provider trong chuỗi connection string. **Ví dụ 5.2** sẽ minh họa cách thực hiện này:

```

// Create Connection
OleDbConnection cnn = new
OleDbConnection("Provider=Microsoft.Jet.OleDb.4.0; Data
Source=QLSINHVIEN.mdb");

```

Các bước còn lại tương tự như SqlConnection.

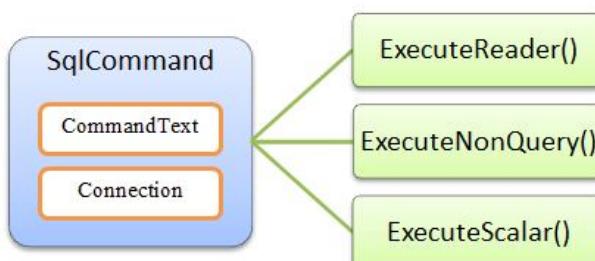
5.3. Đối tượng Command

5.3.1. Giới thiệu

Sau khi đã làm quen cách kết nối cơ sở dữ liệu SQLServer bằng đối tượng SqlConnection hay cơ sở dữ liệu Access ta nhận thấy tập lệnh đã khai báo tương tự nhau.

Khi làm việc với đối tượng Command, chúng ta chọn đối tượng SqlConnection và cơ sở dữ liệu SqlServer để tìm hiểu cách sử dụng chúng. Đối với cơ sở dữ liệu Access thì ta cũng có những đối tượng với những thuộc tính và phương thức tương tự.

Đối tượng SqlCommand cho phép chúng ta chọn kiểu tương tác mà chúng ta muốn thực hiện với database. Ví dụ, chúng ta có thể thực hiện các lệnh select, insert, modify, và delete các dòng trong một bảng của Database.



Hình 5.4. Đối tượng SqlCommand

5.3.2. Khai báo và khởi tạo đối tượng Command

Tương tự như các đối tượng C# khác, bạn tạo đối tượng SqlCommand bằng cách khai báo một thê hiện của nó:

```

// Create SqlCommand
SqlCommand cmd = new SqlCommand("select MaSinhVien from
dbo.SinhVien", cnn);

```

Hoặc

```

// Create SqlCommand
SqlCommand cmd;
string strSQL = "select MaSinhVien from dbo.SinhVien";
cmd = new SqlCommand();
cmd.CommandText = strSQL;
cmd.Connection = cnn;

```

Đối với dữ liệu là Access

```
// Create OleDbCommand  
  
OleDbCommand cmd = new OleDbCommand("select MaSinhVien from  
dbo.SinhVien", cnn);
```

Truy vấn dữ liệu (Querying Data)

Khi dùng một lệnh SQL SELECT, bạn lấy được một dữ liệu từ database để hiển thị. Để làm được điều này với SqlCommand, bạn cần dùng phương thức ExecuteReader() để trả về một đối tượng SqlDataReader.

Các ví dụ sau sẽ minh họa các thao tác: select, thêm, xóa và sửa của đối tượng Command.

Ví dụ 5.3: truy vấn select

```
// 1. Instantiate a new command with a query and connection  
SqlCommand cmd = new SqlCommand("select MaSinhVien from  
SinhVien", cnn);  
  
// 2. Call Execute reader to get query results  
SqlDataReader rdr = cmd.ExecuteReader();
```

Ví dụ 5.4: Chèn dữ liệu (Inserting Data)

```
// prepare command string  
string insertString =  
@"insert into dbo.SinhVien (MaSinhVien, HoTen, NgaySinh,  
MaLop ) values ( '07702451','Nguyễn Văn A','2013-03-29  
03:39:23','03DHTH')";  
  
// 1. Instantiate a new command with a query and connection  
SqlCommand cmd = new SqlCommand(insertString, cnn);  
// 2. Call ExecuteNonQuery to send command  
cmd.ExecuteNonQuery();
```

Ví dụ 5.5: Cập nhật dữ liệu (Updating Data)

```
// prepare command string  
string updateString = @"update dbo.SinhVien set  
MaLop='04DHTH' where MaSinhVien='07702451"';  
  
// 1. Instantiate a new command with command text only  
SqlCommand cmd = new SqlCommand(updateString);  
// 2. Set the Connection property  
cmd.Connection = cnn;  
// 3. Call ExecuteNonQuery to send command  
cmd.ExecuteNonQuery();
```

Ví dụ 5.6: xóa dữ liệu (Deleting Data)

```
// prepare command string  
string deleteString = @"delete from dbo.SinhVien where  
MaSinhVien='07702451"';  
// 1. Instantiate a new command  
SqlCommand cmd = new SqlCommand();  
// 2. Set the CommandText property  
cmd.CommandText = deleteString;  
// 3. Set the Connection property  
cmd.Connection = cnn;  
// 4. Call ExecuteNonQuery to send command  
cmd.ExecuteNonQuery();
```

Ví dụ 5.7: Lấy một giá trị đơn

```
// 1. Instantiate a new command  
SqlCommand cmd = new SqlCommand("select count(*) from  
dbo.SinhVien", cnn);  
// 2. Call ExecuteNonQuery to send command  
int count = (int)cmd.ExecuteScalar();
```

Đối tượng SqlCommand cho phép bạn truy vấn và gửi lệnh đến một database. Nó có các phương thức sử dụng cho các lệnh khác nhau. Phương thức ExecuteReader() trả về một đối tượng SqlDataReader để hiển thị kết quả của câu truy vấn. Cho các lệnh insert, update và delete, bạn dùng phương thức ExecuteNonQuery(). Nếu bạn chỉ cần một giá trị đơn từ một câu truy vấn, phương thức ExecuteScalar() là lựa chọn tốt nhất.

5.3.3. Một số thuộc tính và phương thức cơ bản

Lệnh SQL được gán trong thuộc tính CommandText của đối tượng Command sẽ được thực thi bằng một trong các phương thức được chỉ ra ở bảng dưới đây

Thuộc tính	Mô tả
CommandText	Chuỗi SQL hay tên đối tượng.
CommandType	Loại đối tượng cung cấp trong thuộc tính CommandText. Chẳng hạn, bạn khai báo thủ tục nội tại của SQL Server cho thuộc tính CommandText thì giá trị khai báo cho thuộc tính CommandType là StoredProcedure.
Connection	Đối tượng Connection tương ứng. Chẳng hạn, khi bạn sử dụng SqlCommand thì thuộc tính này có thể gán là đối tượng SqlConnection.
CommandTimeOut	Thời gian chờ tính bằng giây trước khi kết thúc thực thi.

Phương thức	Mô tả
-------------	-------

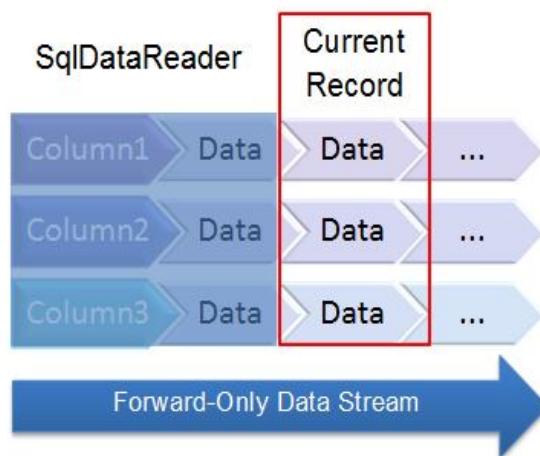
ExecuteNonQuery	Thực thi truy vấn hành động.
ExecuteReader	Thực thi một query và trả về đối tượng DataReader để có thể truy cập tập kết quả của query đó.
ExecuteScalar	Thực thi một query và trả về giá trị của cột đầu tiên trong dòng đầu tiên của tập kết quả.
ExecuteXmlReader	Chỉ có cho data provider SQL Server. Trả về một đối tượng XmlReader dùng để truy xuất tập dữ liệu. Tham khảo thông tin về XmlReader trong MSDN.

ExecuteReader là phương thức quan trọng nhất trong các phương thức kể trên. Phương thức này trả về một đối tượng DataReader giúp truy xuất đến các dòng dữ liệu trả về bởi query.

5.4. Đối tượng DataReader

5.4.1. Giới thiệu

Một SqlDataReader là đối tượng phù hợp để đọc dữ liệu một cách hiệu quả nhất. Như tên gọi, chúng ta không thể dùng nó để ghi dữ liệu. DataReader cho phép lấy các dòng và cột dữ liệu của dữ liệu trả về khi thực thi một query. Việc truy xuất dòng được định nghĩa bởi interface IDataRecord.



Hình 5.5. SqlDataReader

Trong phần này phương thức ExecuteReader của đối tượng SqlCommand thuộc phần trước, chúng ta đã biết cách khai báo để nắm giữ tập dữ liệu từ cơ sở dữ liệu nguồn do phương thức ExecuteReader trả về cho đối tượng SqlDataReader.

Như vậy, đối tượng SqlDataReader là bộ đọc dùng để đọc dữ liệu trực tiếp từ cơ sở dữ liệu nguồn thông qua phương thức ExecuteReader của đối tượng SqlCommand.

Tuy nhiên, chúng ta chỉ nên sử dụng đối tượng SqlDataReader khi tập dữ liệu có số lượng mẫu tin vừa phải và không có nhu cầu xử lý trên tập dữ liệu đó.

Tạo một đối tượng SqlDataReader

Để có được một thể hiện của SqlDataReader chúng ta phải gọi phương thức

ExecuteReader() của một đối tượng SqlCommand, như sau:

```
//Create SqlDataReader  
  
SqlDataReader rdr = cmd.ExecuteReader();
```

Phương thức ExecuteReader() của đối tượng SqlCommand, trả về một thể hiện của SqlDataReader.

Đọc dữ liệu

Để đọc dữ liệu, bạn phải lấy dữ liệu từ một bảng từng dòng một (row-by-row). Mỗi lần một dòng được đọc, dòng trước đó sẽ không còn hiệu lực. Để đọc lại dòng đó, bạn cần phải tạo một thể hiện mới của SqlDataReader và đọc xuyên qua luồng dữ liệu một lần nữa.

Phương pháp điển hình để đọc từ luồng dữ liệu trả về bởi SqlDataReader là lặp qua mỗi dòng với một vòng lặp while.

Ví dụ 5.8: minh họa cách đọc dữ liệu của DataReader

```
while (rdr.Read())  
{  
  
    // get the results of each column  
    string maSV = (string)rdr["MaSinhVien"];  
    string hoten = (string)rdr["HoTen"];  
    string malop = (string)rdr["MaLop"];  
  
    // print out the results  
    Console.WriteLine("{0,-25}", maSV);  
    Console.WriteLine("{0,-20}", hoten);  
    Console.WriteLine("{0,-25}", malop);  
    Console.WriteLine();  
}
```

☞ **Lưu ý:** phương thức Read() của SqlDataReader, rdr, trong điều kiện của vòng lặp while trong đoạn code trên. Giá trị trả về của Read() là kiểu bool và trả về true đến khi nào vẫn còn dòng để đọc. Sau khi dòng cuối được đọc trong luồng dữ liệu, Read() trả về false.

Đối tượng SqlDataReader cho phép chúng ta đọc nhanh dữ liệu theo cách forward-only. Chúng ta lấy dữ liệu bằng cách đọc từng dòng từ luồng dữ liệu. Gọi phương thức Close() của SqlDataReader để đảm bảo sẽ không có tài nguyên nào thất thoát.

5.4.2. Một số thuộc tính và phương thức cơ bản

Thuộc tính	Mô tả
------------	-------

IsClosed	Trả về true/false: DataReader đã đóng hay chưa.
FieldCount	Trả về số cột mà đối tượng SqlDataReader đang nắm giữ.
HasRows	Trả về True/False ứng với mẫu tin tồn tại trong DataReader.
Item (i)	Cho phép ta lấy giá trị của cột theo chỉ mục.
Item (column name)	Cho phép ta lấy giá trị của cột theo tên cột.
Phương thức	Mô tả
Read()	Đọc từng mẫu tin đang nắm giữ.
Close()	Đóng đối tượng SqlDataReader.
GetString(i)	Trả về giá trị dạng chuỗi của cột dữ liệu chỉ định.
GetInt32(i)	Trả về giá trị nguyên.
GetName(i)	Trả về giá trị dạng chuỗi là tên cột của cột thứ i.
GetOrdinal(name)	Trả về giá trị là thứ tự của cột với tên chỉ định.
Item(i/name)	Trả về giá trị kiểu Object của cột thứ i hoặc tên cột.

5.5. Đối tượng DataAdapter

5.5.1. Giới thiệu

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

SqlDataAdapter sẽ được dùng để quản lý các kết nối với nguồn dữ liệu và cho chúng ta chế độ làm việc disconnected. SqlDataAdapter mở một kết nối chỉ khi cần thiết và đóng nó ngay sau khi tác vụ được hoàn thành.

DataAdapter là một bộ gồm bốn đối tượng Command :

- SelectCommand : cho phép lấy thông tin từ nguồn dữ liệu về.
- InsertCommand : cho phép thêm dữ liệu vào bảng trong nguồn dữ liệu.
- UpdateCommand : cho phép sửa đổi dữ liệu trên bảng trong nguồn dữ liệu.
- DeleteCommand : cho phép hủy bỏ dữ liệu trên bảng trong nguồn dữ liệu.

Nếu người sử dụng thay đổi dữ liệu trên đối tượng DataSet, chúng ta có thể cập nhật dữ liệu đã được thay đổi vào dữ liệu nguồn bằng phương thức Update.



Hình 5.6. Đối tượng DataAdapter

Tạo một SqlDataAdapter

SqlDataAdapter chứa các lệnh SQL và đối tượng connection để đọc và ghi dữ liệu. Bạn khởi tạo nó với câu SQL select và đối tượng connection:

```
SqlDataAdapter daSinhVien = new SqlDataAdapter("select  
MaSinhVien,MaLop from dbo.SinhVien", cnn);
```

Dòng mã trên tạo một đối tượng SqlDataAdapter, daSinhVien. Câu SQL SELECT dữ liệu nào sẽ được đọc vào DataSet. Đối tượng connection, cnn, nên được khởi tạo từ trước, nhưng không được mở. Đó là công việc của SqlDataAdapter để mở và đóng connection khi phương thức Fill() và Update() được gọi.

Đỗ dữ liệu vào DataSet

Để đổ dữ liệu vào DataSet bạn cần dùng phương thức Fill() của SqlDataAdapter, như sau:

```
daSinhVien.Fill(dsSinhVien, "SinhVien");
```

Phương thức Fill(), trong dòng trên lấy hai tham số: một DataSet và một tên bảng. DataSet phải được tạo trước khi bạn đổ dữ liệu vào nó. Tham số thứ hai là tên của bảng sẽ được tạo trong DataSet. Bạn có thể đặt bất kỳ tên gì cho bảng. Thông thường, tôi sẽ để tên bảng trùng với tên gốc của nó trong database. Tuy nhiên, nếu câu select của SqlDataAdapter chứa một lệnh join, bạn sẽ cần phải đặt một tên rõ ràng khác cho bảng.

Cập nhật thay đổi

Sau khi thay đổi được thực hiện trên dữ liệu, bạn sẽ cần ghi lại vào database. Dòng mã sau cho thấy cách dùng phương thức Update của SqlDataAdapter để cập nhật các thay đổi vào database.

```
daSinhVien.Update(dsSinhVien, "SinhVien");
```

Phương thức Update() trên được gọi trên thẻ hiện của SqlDataAdapter có tham số đầu tiên là chính đối tượng gọi phương thức. Tham số thứ hai của phương thức Update() chỉ ra bảng nào trong DataSet sẽ được cập nhật. Bảng chứa một danh sách các dòng dữ liệu đã bị thay đổi và các property Insert, Update, Delete của SqlDataAdapter chứa các lệnh SQL dùng để thực hiện thay đổi database.

5.5.2. Một số thuộc tính và phương thức cơ bản

Thuộc tính	Mô tả
DeleteCommand	Cho phép chúng ta gán hay lấy chuỗi SQL dạng Delete tương ứng với bảng dữ liệu khai báo để xóa dữ liệu trong dữ liệu nguồn ứng với dữ liệu trong đối tượng DataSet hay DataTable.
InsertCommand	Cho phép gán hay lấy chuỗi SQL dạng Insert tương ứng với bảng dữ liệu được khai báo khi đọc dữ liệu từ dữ liệu nguồn vào đối

	tượng DataSet hay DataTable.
SelectCommand	Cho phép ta gán hay lấy chuỗi Sql dạng Select tương ứng với bảng dữ liệu đã khai báo để đọc dữ liệu từ dữ liệu nguồn cho đối tượng DataSet hay DataTable.
UpdateCommand	Cho phép ta gán hay lấy chuỗi Sql dạng Update tương ứng với bảng dữ liệu khai báo để cập nhật dữ liệu vào dữ liệu nguồn từ đối tượng DataSet hay DataTable.
TableMappings	Trả về một tập hợp ánh xạ bảng trong cơ sở dữ liệu và DataTable.
Phương thức	Mô tả
Fill	Đổ dữ liệu nguồn vào đối tượng DataSet hay DataTable.
Update	Cập nhật dữ liệu thay đổi kể từ lần cập nhật cuối cùng từ đối tượng DataSet (hay đối tượng DataTable) vào dữ liệu nguồn.
Sự kiện	Mô tả
RowUpdated	Xảy ra khi một bản ghi đã được cập nhật vào cơ sở dữ liệu.
RowUpdating	Sự kiện này xảy ra khi một bản ghi đang cập nhật vào cơ sở dữ liệu.

5.5.3. SqlDataAdapter control và đối tượng SqlDataAdapter

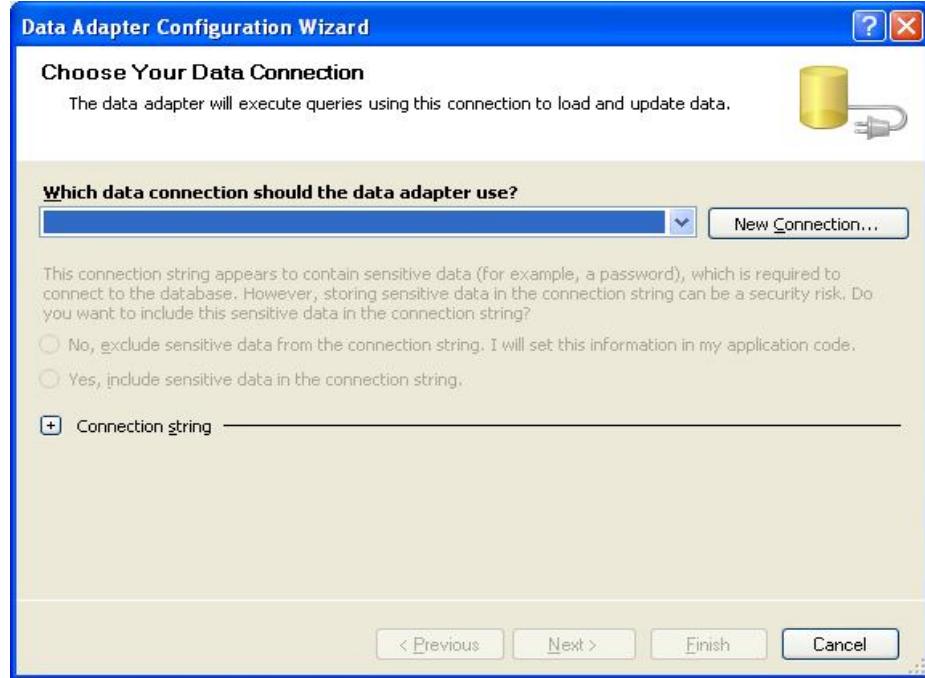
Chúng ta có hai cách để sử dụng SqlDataAdapter, một là sử dụng control SqlDataAdapter từ thanh công cụ, hai là khai báo và sử dụng đối tượng SqlDataAdapter.

SqlDataAdapter control

Adapter Configuration Wizard là một công cụ mạnh mẽ để phát triển ứng dụng cơ sở dữ liệu.

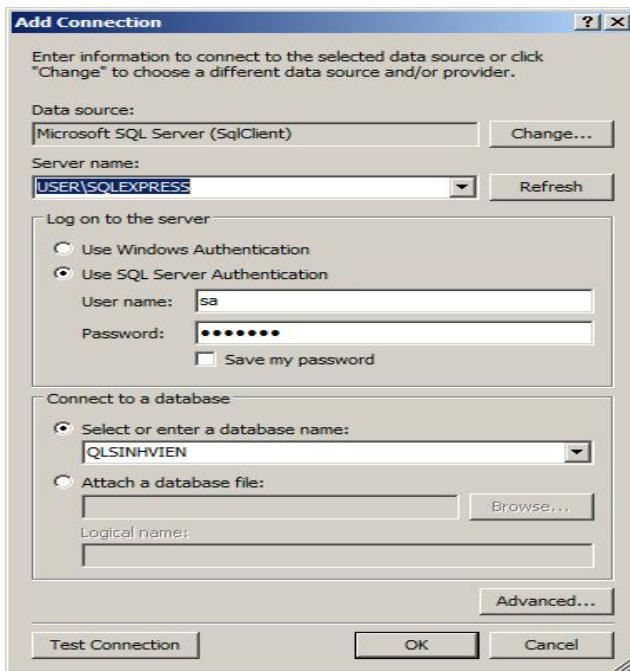
Trong phần này, chúng ta sẽ tạo ra dữ liệu SqlDataAdapter, khi đọc dữ liệu từ một nguồn dữ liệu SQLServer, Chúng ta sẽ dễ dàng phát triển các ứng dụng cơ sở dữ liệu bằng cách sử dụng cấu hình Adapter Configuration Wizard.

Để khai báo và sử dụng điều khiển SqlDataAdapter, trước tiên ta kéo điều khiển này vào Form từ ngăn Data của ToolBox. Nếu trong ToolBox chưa có SqlDataAdapter ta click chuột phải vào Toolbox rồi chọn Choose Items, sau đó chọn SqlDataAdapter rồi nhấn OK. kế đến chọn khai báo mới hay sử dụng lại kết nối cơ sở dữ liệu đang có trong phần “Choose your Data Connection”.



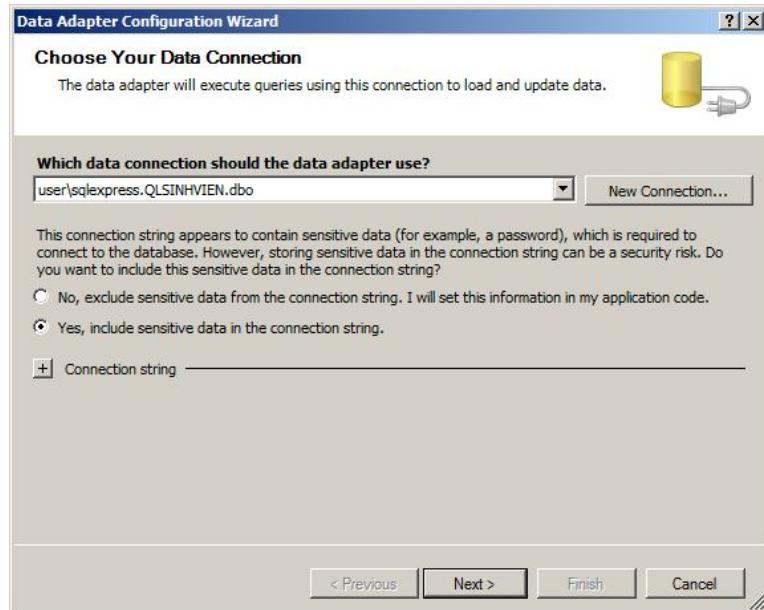
Hình 5.7. Data Connection

Trong trường hợp chọn nút New Connection, ta chọn trình điều khiển cơ sở dữ liệu SQL Server, ứng với cơ sở dữ liệu SQL Server cùng với tham số khác như: Tên Server, tên cơ sở dữ liệu, tài khoản, Password.



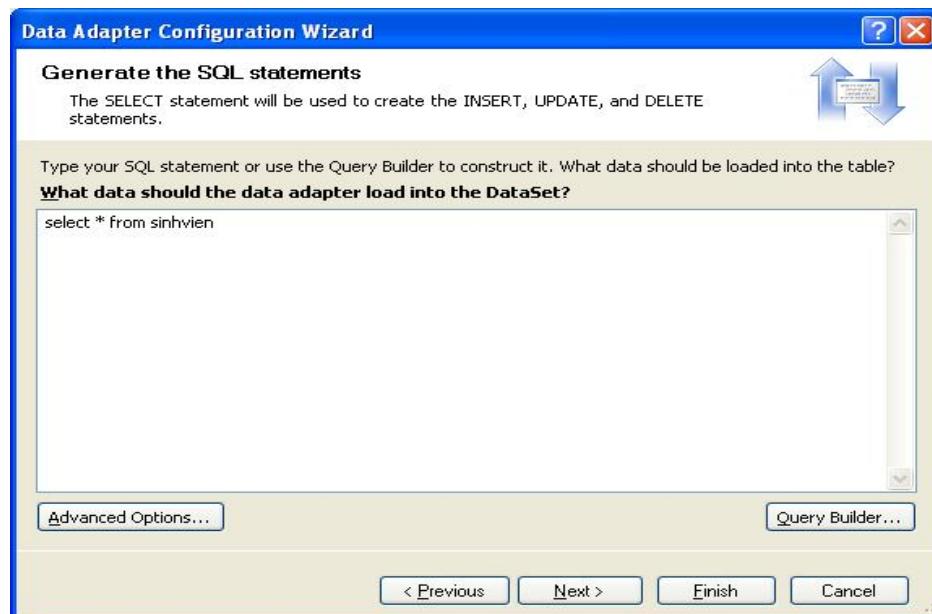
Hình 5.8. Add Connection

Ta chọn next đối với hộp thoại sau.



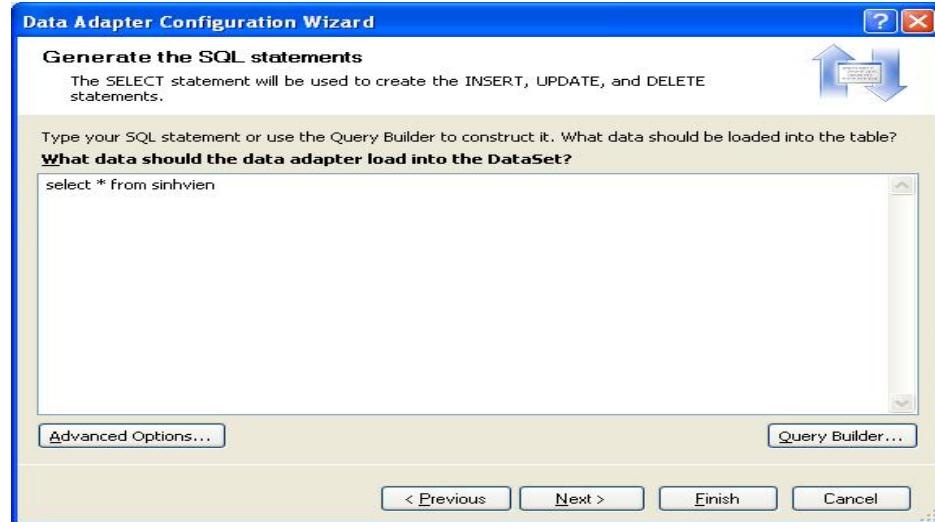
Hình 5.9. Data Connection

Ké đến, chọn loại câu truy vấn trong phần “Choose a Query Type”, đối với trường hợp này chúng ta chọn mặc định “Use SQL Statement”. Bằng cách khai báo phát biểu SQL trong phần “Generate the SQL Statements” hay sử dụng tiện ích hỗ trợ “Query Builder”.



Hình 5.10. Data Adapter

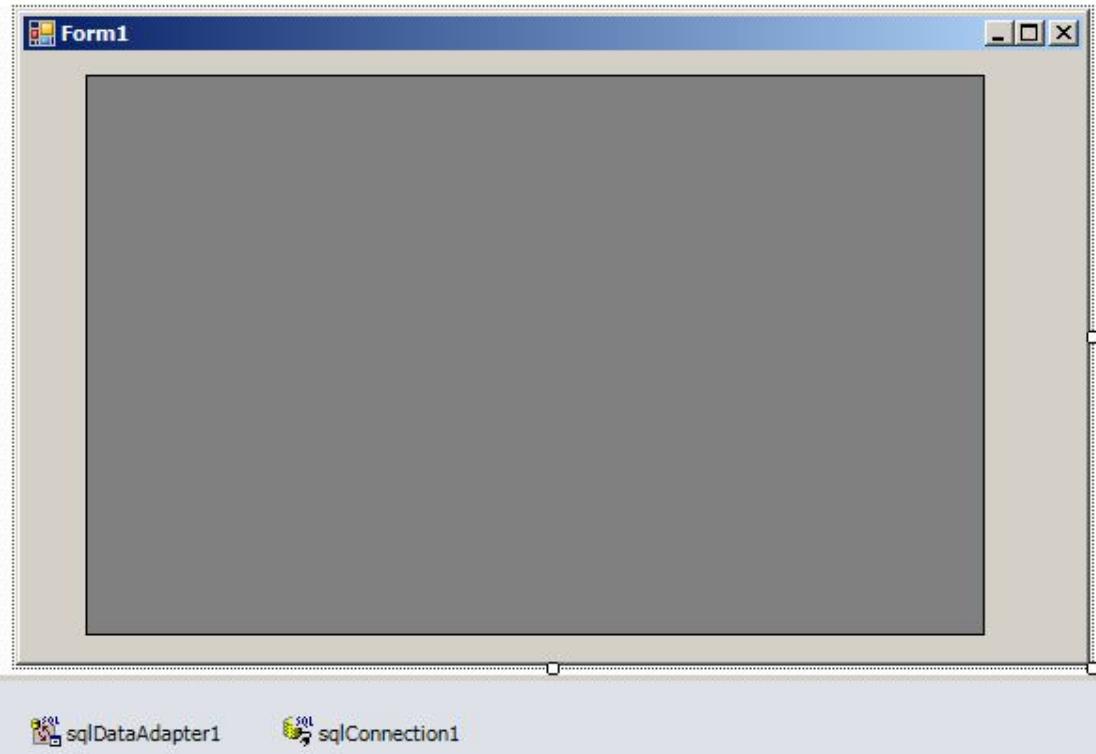
Ké đến, chọn loại câu truy vấn trong phần “Choose a Query Type”, đối với trường hợp này chúng ta chọn mặc định “Use SQL Statement”. Bằng cách khai báo phát biểu SQL trong phần “Generate the SQL Statements” hay sử dụng tiện ích hỗ trợ “Query Builder”.



Hình 5.11. Data Adapter Configuration

Sau đó ta chọn Finish ở màn hình kế tiếp để kết thúc các tùy chọn đối với SqlDataAdapter.

Kết thúc cấu hình điều khiển SqlDataAdapter trở lại giao diện Form, lập tức điều khiển SqlDataAdapter và SqlConnection sẽ xuất hiện trên Form.



Hình 5.12. Form SqlDataAdapter và SqlConnection

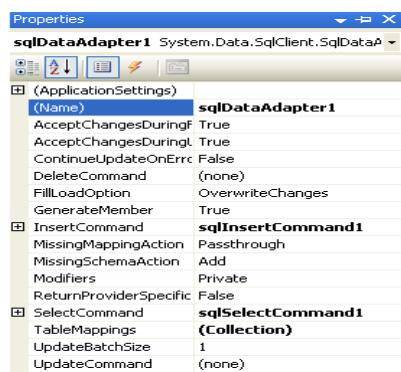
Sau đó ta sử dụng phương thức Fill() của SqlDataAdapter để điền dữ liệu vào DataSet.

```

protected void FillDBGrid()
{
    DataSet ds = new DataSet();
    sqlDataAdapter1.Fill(dsSinhVien);
    dataGrid1.DataSource = ds.Tables[0];
}

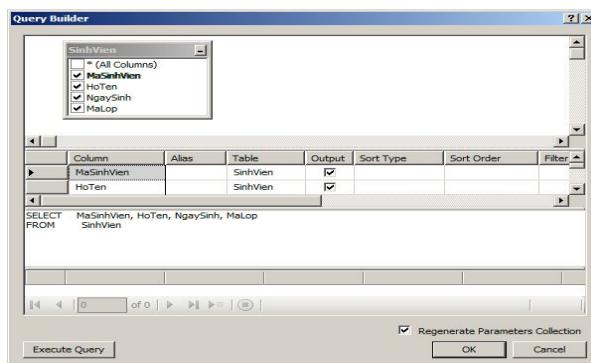
```

Bây giờ chúng ta có một DataAdapter trên mẫu, chúng ta hãy xem xét các thuộc tính thành phần SqlDataAdapter. Chúng ta có thể xem các thuộc tính của nó bằng cách kích chuột phải vào adapter và chọn mục trình đơn Properties. Nó bao gồm InsertCommand, DeleteCommand, SelectCommand, UpdateCommand như hình bên dưới:



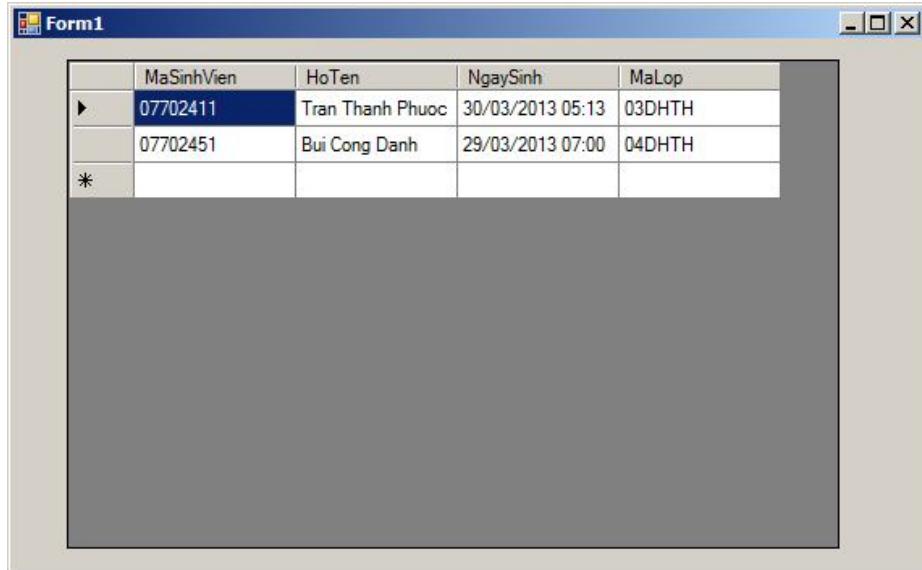
Hình 5.13. Thuộc tính SqlDataAdapter

Như chúng ta cũng thấy trong hình dưới đây, có thể thiết lập CommandText, CommandType, kết nối, và như vậy bằng cách sử dụng hộp thoại thuộc tính. Nếu bạn kích đúp vào CommandText, nó bật lên Query Builder nơi bạn có thể xây dựng lại truy vấn của chúng ta.



Hình 5.14. Query Builder

Sau đó Built và chạy Project chúng ta có kết quả sau:



Hình 5.15. Minh họa Query Builder

5.6. Dataset

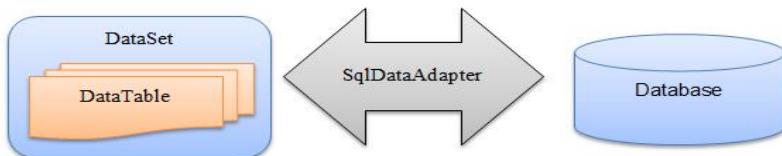
5.6.1. Giới thiệu

DataSet là một trong hai thành phần chính của ADO.NET, nó được thiết kế nhằm tăng tốc độ truy cập và thao tác dữ liệu trong môi trường ứng dụng đa lớp (n-tier).

Một DataSet là một đối tượng chứa dữ liệu trong bộ nhớ và có thể gồm nhiều bảng. DataSet chỉ chứa dữ liệu chứ không tương tác với nguồn dữ liệu.

DataSet được xem như một container, dùng để lưu các đối tượng DataTable, DataView và DataRelation cùng với lược đồ quan hệ của chúng. Khi có tập dữ liệu lớn hoặc cần thao tác trên tập dữ liệu đó, bạn nên sử dụng đối tượng DataSet để nắm giữ chúng thay vì sử dụng đối tượng SqlDataReader.

Đối tượng DataSet sử dụng bộ nhớ truy cập nhanh (Cache) của máy tính để lưu trữ dữ liệu mà nó nắm giữ, chính vì vậy nếu dung lượng dữ liệu càng lớn thì đối tượng này chiếm giữ bộ nhớ càng nhiều.



Hình 5.16. DataSet

Tạo đối tượng DataSet

```
DataSet dsSinhVien = new DataSet();
```

Constructor của DataSet không yêu cầu tham số. Tuy nhiên có một overload chấp nhận một chuỗi đại diện cho tên của DataSet, được dùng nếu chúng ta cần serialize dữ liệu thành XML. Bây giờ, chúng ta có một DataSet rỗng và cần một SqlDataAdapter

để nạp dữ liệu cho nó.

Đổ dữ liệu vào DataSet

Để đổ dữ liệu vào DataSet bạn cần dùng phương thức Fill() của SqlDataAdapter, như sau:

```
daSinhVien.Fill(dsSinhVien, "SinhVien");
```

Phương thức Fill(), trong dòng trên lấy hai tham số: một DataSet và một tên bảng. DataSet phải được tạo trước khi bạn đổ dữ liệu vào nó. Tham số thứ hai là tên của bảng sẽ được tạo trong DataSet. Chúng ta có thể đặt bất kì tên gì cho bảng. Thông thường, chúng ta nên để tên bảng trùng với tên gốc của nó trong database. Tuy nhiên, nếu câu select của SqlDataAdapter chứa một lệnh join, chúng ta cần phải đặt một tên rõ ràng khác cho bảng.

Sử dụng DataSet

Một DataSet sẽ gắn dữ liệu vào DataGrid của ASP.NET và Windows form. Đây là một ví dụ sẽ gán DataSet cho một Windows forms DataGrid:

```
dataGridView1.DataSource = dsSinhVien.Tables[0];
```

Cập nhật thay đổi

Sau khi thay đổi được thực hiện trên dữ liệu, bạn sẽ cần ghi lại vào database. Dòng mã sau cho thấy cách dùng phương thức Update của SqlDataAdapter để cập nhật các thay đổi vào database.

```
daSinhVien.Update(dsSinhVien, "SinhVien");
```

Phương thức Update() trên được gọi trên thê hiện của SqlDataAdapter có tham số đầu tiên là chính đối tượng gọi phương thức. Tham số thứ hai của phương thức Update() chỉ ra bảng nào trong DataSet sẽ được cập nhật.

5.6.2. Một số thuộc tính và phương thức cơ bản

Thuộc tính	Mô tả
DataSetName	Cho phép gán hay lấy chuỗi ứng với tên của đối tượng DataSet.
HasErrors	Trả về giá trị Boolean, cho biết một trong những đối tượng DataTable bên trong DataSet phát sinh lỗi.
HasChanges	Trả về true nếu có sự thay đổi dữ liệu trong đối tượng Dataset. Bằng cách dựa vào thuộc tính này, ta có thể kiểm soát người sử dụng có thay đổi dữ liệu trong đối tượng DataSet hay không.
Tables	Trong đối tượng DataSet có thể có một hoặc nhiều đối tượng DataTable, thuộc tính Tables trả về một tập đối tượng DataTable.
Relations	Tương tự thuộc tính Table, thuộc tính Relations trả về tập các quan hệ của những cặp đối tượng DataTable trong đối tượng DataSet.

Phương thức	Mô tả
AcceptChanges	Chấp nhận sự thay đổi do người dùng tạo ra trên đối tượng DataSet.
Clear	Loại bỏ tất cả những mẩu tin trong các đối tượng DataTable thuộc đối tượng DataSet. Tuy nhiên, nếu loại bỏ đối tượng DataTable khỏi đối tượng DataSet thì ta dùng phương thức Clear của Table Collection.
RejectChanges	Không chấp nhận sự thay đổi trên DataSet.
GetChanges	Trả về một đối tượng DataSet chứa đựng các đối tượng DataTable, bao gồm những hàng dữ liệu có thay đổi, thêm mới hay xóa. Chẳng hạn, trong trường hợp ta cập nhật sự thay đổi dữ liệu từ DataSet vào dữ liệu nguồn.

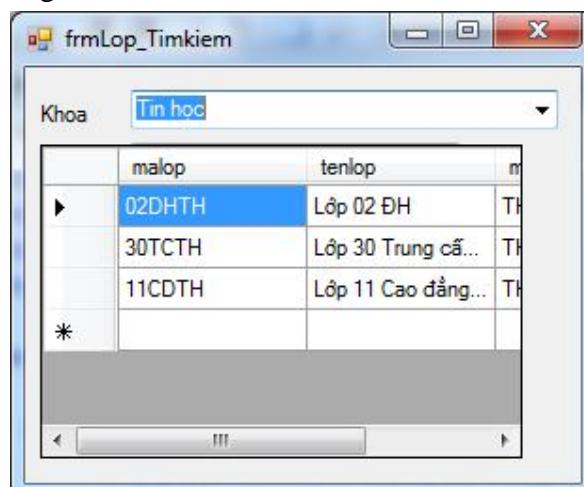
5.7. Hiển thị dữ liệu bằng DataGridView control

Để có thể sử dụng được điều khiển DataGridView, trước tiên ta click chọn biểu tượng DataGridView control trên Toolbox. Sau đó, ta vẽ DataGridView lên Form và đặt tên cho nó trong Properties.

Để hiển thị dữ liệu lên DataGridView, trước tiên ta phải có một biến đối tượng chứa cơ sở dữ liệu như Dataset hoặc DataTable (sẽ trình bày tiếp theo). Sau đó ta chỉ cần gán thuộc tính Datasource cho biến đối tượng đó.

```
dataGridView1.DataSource = dsSinhVien.Tables[0];
```

Ví dụ 5.9: minh họa cách sử dụng đối tượng DataAdapter (data), Dataset (ds), DataGridView Control (grd) và một ComboBox (cbkhoa) chứa danh sách các khoa. Formload sẽ hiển thị tất cả các khoa trong bảng Khoa lên cbkhoa, hiển thị tất cả các lớp tương ứng với khoa lên grd. Khi người dùng chọn một khoa, lớp tương ứng với khoa đó sẽ hiển thị lên grd.



Hình 5.17. Hiển thị dữ liệu trong DataGridView control

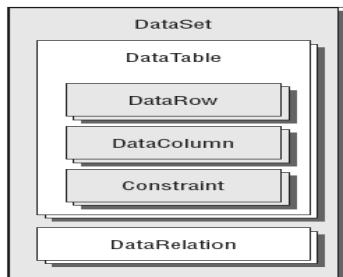
Source code:

```
void ht_khoa()
{
    string s = "select * from khoa";
    data = new SqlDataAdapter(s, cn);
    tb = new DataTable();
    data.Fill(tb);
    cbkhoa.DataSource = tb;
    cbkhoa.DisplayMember = "tenkhoa";
    cbkhoa.ValueMember = "makhoa";
}
void ht_grd()
{
    string s = "select * from lop where makhoa = '" +
    cbkhoa.SelectedValue.ToString() + "'";
    data = new SqlDataAdapter(s, cn);
    tb = new DataTable();
    data.Fill(tb);
    grd.DataSource = tb;
}
private void frmLop_Timkiem_Load(object sender, EventArgs e)
{
    cn = new SqlConnection("initial catalog = qlsv; data
    source = PHUOC-PC\\SQLEXPRESS; integrated security =
    true");
    cn.Open();
    ht_khoa();
    ht_grd();
}
private void cbkhoa_SelectedIndexChanged(object sender,
EventArgs e)
{
    ht_grd();
}
```

5.8. DataTable

5.8.1. Giới thiệu

Đối tượng DataTable là phần tử chính trong đối tượng DataSet. Mỗi đối tượng DataTable sẽ nắm giữ một hay nhiều đối tượng DataRow. Đối tượng DataTable có thể có quan hệ với các đối tượng DataTable khác trong cùng đối tượng DataSet.



Hình 5.18. DataTable

Khi trình bày dữ liệu trên điều khiển DataGridView (hoặc các điều khiển dạng danh sách khác), thay vì sử dụng đối tượng DataSet ứng với thuộc tính Tables[i] thì ta có thể sử dụng trực tiếp đối tượng DataTable. **Ví dụ 5.10** sẽ minh họa cách thực hiện này:

```
DataTable dtb;
dtb = new DataTable();
SqlDataAdapter daSinhVien = new SqlDataAdapter("SELECT *
FROM dbo.SinhVien", cnn);
daSinhVien.Fill(dtb);
dataGridView1.DataSource = dtb;
```

5.8.2. Một số thuộc tính và phương thức cơ bản

Thuộc tính	Mô tả
Dataset	Trả về đối tượng DataSet mà đối tượng DataTable trực thuộc.
HasErrors	Tương tự như thuộc tính HasErrors trong đối tượng DataSet, thuộc tính HasErrors trả về True nếu có phát sinh lỗi trong hàng dữ liệu.
TableName	Trả về tên của bảng dữ liệu, nếu đối tượng DataTable không khai báo tên thì tên của nó chính là tên bảng dữ liệu hay chuỗi SQL.
ChildRelations, ParentRelations	Trả về tập hợp quan hệ cha hay con ứng với hai đối tượng DataTable.
PrimaryKey	Trả về một mảng bao gồm các cột khai báo như khóa chính (Primary key).
Columns	Trả về tập các đối tượng DataColumn của đối tượng DataTable. Ta có thể in ra tên của các cột dữ liệu trong đối tượng DataTable bằng thuộc tính Count của Column Collection.
Rows	Trả về tập đối tượng DataRow của đối tượng DataTable.
Phương thức	Mô tả
AcceptChanges	Chấp nhận sự thay đổi dữ liệu trên đối tượng DataTable.
Clear	Loại bỏ tất cả mẫu tin trong các đối tượng DataTable
RejectChanges	Không chấp nhận sự thay đổi dữ liệu trên đối tượng DataTable.
GetChanges	Trả về một đối tượng DataTable chứa đựng các đối tượng DataRow mà dữ liệu có thay đổi, thêm mới hay xóa.
NewRow	Cho phép thêm mới một DataRow vào đối tượng DataTable
ImportRow	Thêm đối tượng DataRow đang có dữ liệu vào DataTable.

Cho CSDL quản lý sinh viên bao gồm các table:

- Khoa (makhoa, tenkhoa)
- Lop (malop, tenlop)
- Sinhvien (masv, hoten, ngaysinh, malop)
- Monhoc (mamh, tenmh)
- Diem (masv, mamh, diem)

1. Thiết kế Form cho phép Thêm, Xóa, Sửa với hai trường hợp: không kiểm tra và có kiểm tra khóa chính khóa ngoại

a. Form quản lý Khoa

Quản lý Khoa

Mã khoa

Tên khoa

Thêm Xóa Sửa

b. Form quản lý Lớp

frmLop

Mã khoa

Mã lớp

Tên lớp

Thêm Xóa Sửa

c. Form quản lý Sinh viên

Quản lý sinh viên

Mã lớp

Mã sv

Họ tên

Ngày sinh

Thêm Xóa Sửa

d. Form quản lý Môn học

Quản lý môn học

Mã mh |

Tên mh |

Thêm Xóa Sửa

e. Form quản lý Điểm

Quản lý Điểm

Mã sv |

Mã mh |

Điểm |

Thêm Xóa Sửa

2. Sử dụng Combobox, Command, DataReader hiển thị khóa ngoại

a. Form Lớp

Quản lý Lớp

Khoa CNTT

Mã lớp |

Tên lớp |

Thêm Xóa Sửa

b. Form Sinh viên

frmSinhvien_Combo

Lớp 04CDNTH

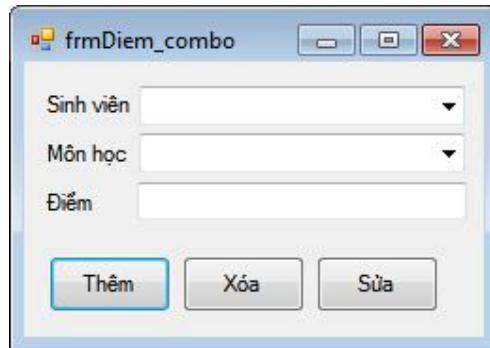
Mã sv |

Họ tên |

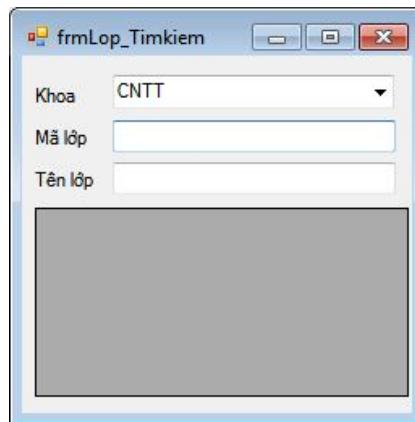
Ngày sinh _/_/_

Thêm Xóa Sửa

c. Form Điểm

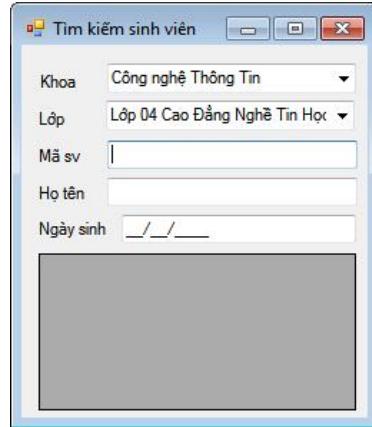


3. Sử dụng Combobox, DataAdapter, Dataset/DataTable hiển thị khóa ngoại (hiển thị tên, giá trị chọn là mã)
 - a. Form Lớp
 - b. Form Sinh viên
 - c. Form Điểm
4. Hiển thị thông tin lên Datagrid, lọc dữ liệu (tìm kiếm)
 - a. Form Lớp



Yêu cầu:

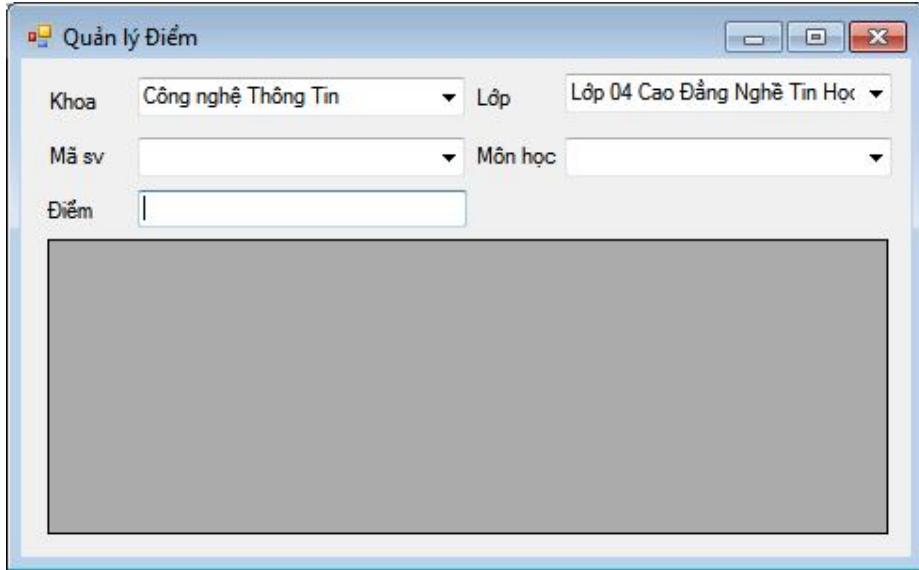
- FormLoad:
 - + Hiển thị Khoa lên Combobox Khoa
 - + Datagrid hiển thị dữ liệu trong bảng Lop
 - + Hiển thị dòng đầu tiên trong Datagrid lên các control
 - Chọn Combobox Khoa
 - + Hiển thị lớp tương ứng với khoa vừa chọn
- b. Form Sinh viên



Yêu cầu

- FormLoad:
 - + Hiển thị Khoa lên Combobox Khoa
 - + Hiển thị Lớp lên Combox Lớp
 - + Datagrid hiển thị dữ liệu trong bảng Sinh viên
 - + Hiển thị dòng đầu tiên trong Datagrid lên các control
- Chọn Combobox Khoa
 - + Hiển thị lớp tương ứng với khoa vừa chọn
- Chọn combobox Lop:
 - + Hiển thị sinh viên thuộc lớp đó

c. Form Điểm



Yêu cầu

- FormLoad:
 - + Hiển thị Khoa lên Combobox Khoa
 - + Hiển thị Lớp lên Combox Lớp

- + Hiển thị Mã sinh viên lên Combobox Sinh viên
- + Hiển thị môn học lên combobox môn học
- + Datagrid hiển thị dữ liệu trong bảng Điểm
- + Hiển thị dòng đầu tiên trong Datagrid lên các control
- Chọn Combobox Khoa
 - + Hiển thị lớp tương ứng với khoa vừa chọn
- Chọn combobox Lop:
 - + Hiển thị sinh viên thuộc lớp đó
- Chọn combobox sinh viên
 - + Hiển thị Điểm của sinh viên đó
- Chọn combobox Môn học
 - + Hiển thị điểm của môn học đó (lưu ý xem có chọn sinh viên hay không?)

5. Form quản lý Khoa hoàn chỉnh (nhập trên textbox)

a. Giao diện



b. Yêu cầu

- **Formload:**
 - + Datagrid: Hiển thị tất cả khoa trong bảng Khoa
 - + Tất cả textbox bị vô hiệu hóa
 - + Các nút Sửa, xóa, Lưu bị vô hiệu hóa
- **Khi chọn vào nút Thêm:**
 - + Các textbox có hiệu lực
 - + Nút Lưu có hiệu lực
 - + Dấu nháy xuất hiện ở textbox Mã khoa.

- Khi chọn vào Datagrid

- + Hiển thị thông tin tương ứng lên các textbox
- + Nút Sửa và Xóa có hiệu lực

- Chọn nút Sửa

- + Nút Lưu có hiệu lực
- + Các textbox có hiệu lực trừ textbox Mã khoa
- + Cho phép sửa các thông tin còn lại

- Khi chọn nút “Lưu”

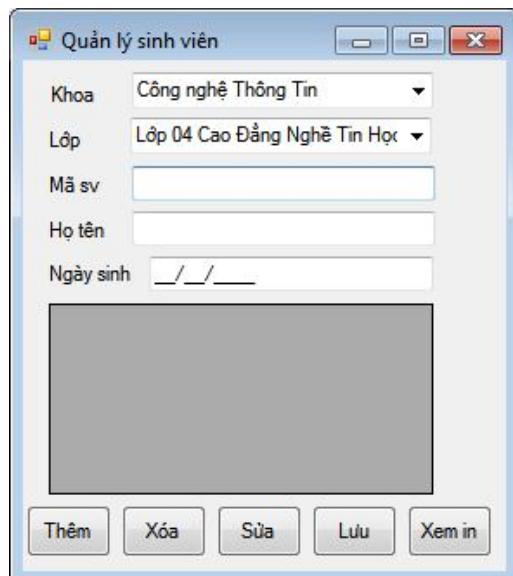
- + Kiểm tra thông tin vừa nhập hoặc sửa cho phù hợp
- + Lưu vào Cơ sở dữ liệu (lưu ý đang lưu Thêm hay Sửa)
- + Thông báo thành công hoặc báo lỗi nếu có
- + Nút Lưu bị vô hiệu hóa

- Khi nhấn nút Xóa.

- + Hiển thị thông báo xác nhận
- + Nếu đồng ý
 - Kiểm tra mã khoa định xóa có tồn tại trong bảng lớp hay không. Nếu có thì hỏi xác nhận “có xóa luôn trong bảng lớp hay không?”. Nếu đồng ý, xóa dữ liệu trong table Lớp tương ứng với mã khoa đang chọn (nếu có); xóa khoa vừa chọn.
 - Nếu mã khoa không tồn tại bên bảng lớp thì xóa bên bảng khoa
- + Hiển thị thông báo nếu xóa thành công hoặc báo lỗi (nếu có)

6. Form quản lý Sinh viên hoàn chỉnh (Nhập trên lưới)

a. Giao diện



b. Yêu cầu

- Formload:

- + Combobox mã khoa: Chứa tên khoa trong bảng khoa
- + Combobox mã lớp: Chứa tên lớp trong bảng lop
- + Datagrid sinh viên: Hiển thị tất cả sinh viên trong bảng sinh viên và chỉ đọc
- + Tất cả textbox, combobox bị vô hiệu hóa
- + Các nút Sửa, xóa, Lưu bị vô hiệu hóa

- Khi chọn vào nút Thêm:

- + Nút Lưu có hiệu lực
- + Cho phép thêm các dòng tiếp theo trên Datagrid
 - **Lưu ý:** không được sửa đổi các dòng trên Datagrid đã có dữ liệu

- Khi chọn vào Datagrid

- + Hiển thị thông tin tương ứng lên các textbox, combobox
- + Nút Sửa và Xóa có hiệu lực

- Chọn nút Sửa

- + Nút Lưu có hiệu lực
- + Cho phép sửa các thông tin trên Datagrid
- + Lưu ý: không cho phép gõ thêm các dòng mới

- Khi chọn nút “Lưu”

- + Kiểm tra thông tin vừa nhập hoặc sửa trên lưới cho phù hợp
- + Lưu vào Cơ sở dữ liệu (lưu ý đang lưu Thêm hay Sửa)
- + Thông báo thành công hoặc báo lỗi nếu có
- + Nút Lưu bị vô hiệu hóa

- Khi nhấn nút Xóa.

- + Hiển thị thông báo xác nhận
- + Nếu đồng ý
 - Xóa dữ liệu trong table điểm tương ứng với mã số sinh viên đang chọn (nếu có) (nhớ cảnh báo nhắc nhở)
 - Xóa dữ liệu trong table sinh viên tương ứng với mã số sinh viên đang chọn
 - Hiển thị thông báo nếu xóa thành công hoặc báo lỗi (nếu có)

7. Làm tương tự cho bài 6 đối với các Table: Lớp, môn học. (nhập trên Textbox)

8. Form quản lý Điểm hoàn chỉnh (nhập trực tiếp trên lưới)

CHƯƠNG 6. CRYSTAL REPORT

Mục tiêu:

Sau khi học xong chương này, sinh viên có khả năng:

- Biết cách thiết kế một Report trong C#.
- Biết cách sử dụng control Crystal Report Viewer để hiển thị Report.
- Biết cách xây dựng ứng dụng tìm kiếm và xem in kết quả vừa tìm kiếm được.

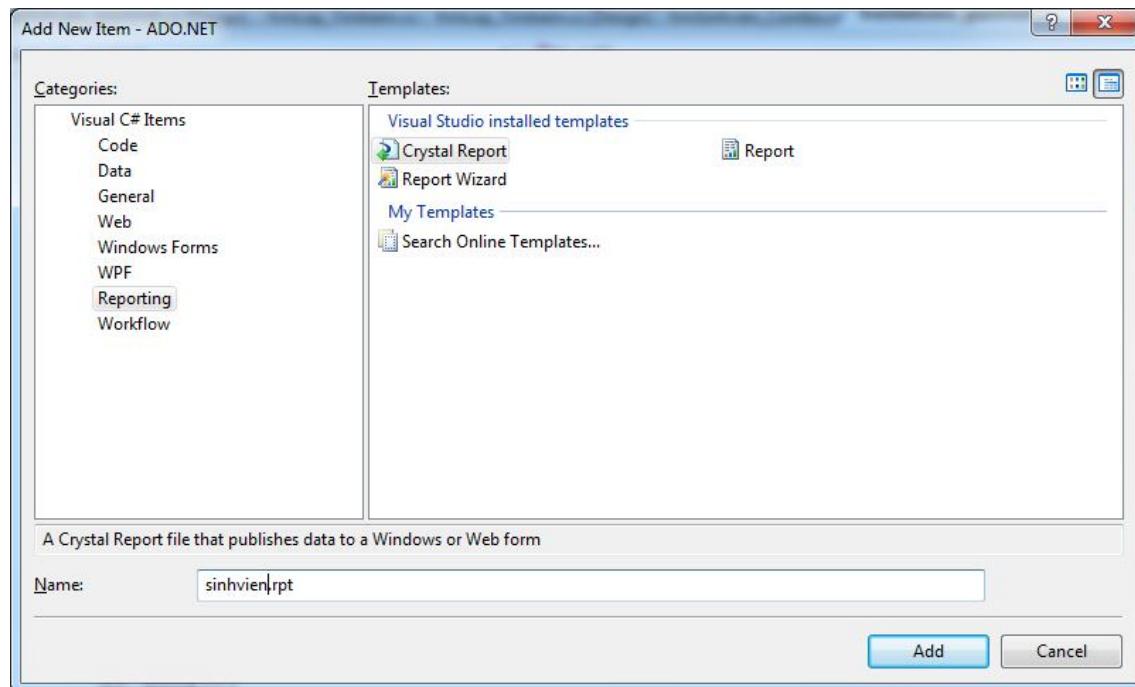
6.1. Giới thiệu

.NET cung cấp Crystal Report như một phần của Project, ta có thể thêm Report bằng cách chọn Project → Add New Item → Crystal Report.

Mỗi đối tượng Crystal Report được tạo ra đều xuất hiện trong danh sách thành phần của dự án, tên mở rộng của chúng là .rpt. Tuy nhiên, ta cũng có thể thiết kế Report từ ứng dụng Crystal Report, sau đó thêm chúng vào Project.

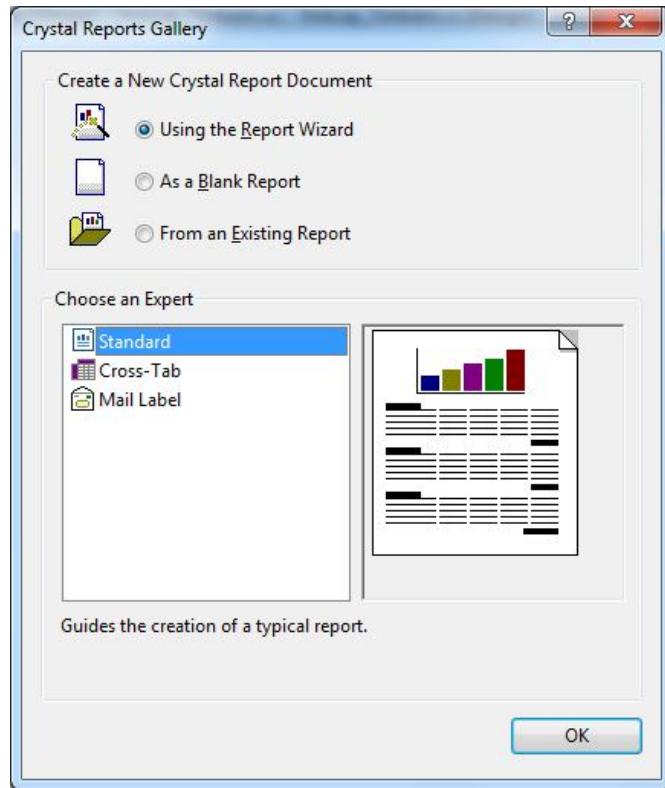
6.2. Thiết kế Report

Vào Project → Add New Item → Crystal Report → Chọn biểu tượng Crystal Report → Đặt tên sinhvien.rpt.



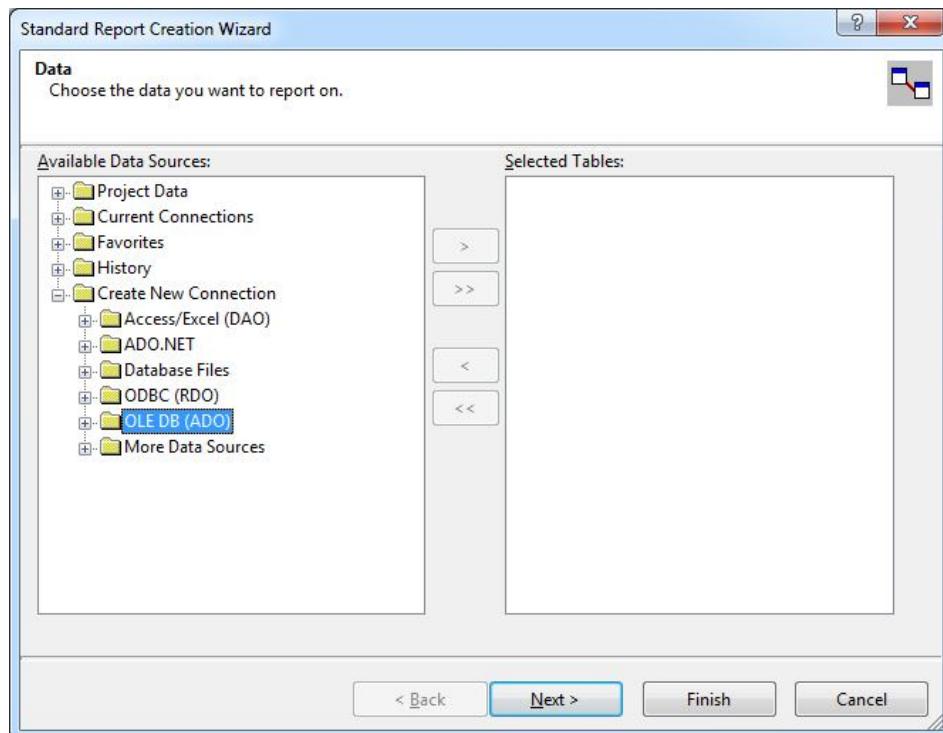
Hình 6.1. Khởi tạo Crystal Report

Chọn mặc định là “Using the Report wizard” rồi chọn vào Standard.



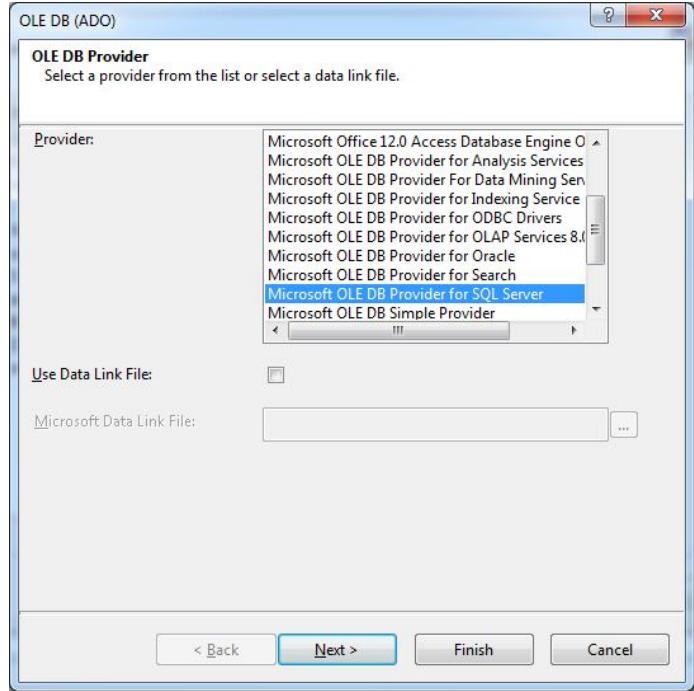
Hình 6.2. Report wizard

Tù ngǎn Data, chọn vào Project Data (nếu đã tạo kết nối cơ sở dữ liệu), hoặc Create New Connection rồi chọn OLE DB (ADO) để tạo mới kết nối cơ sở dữ liệu.



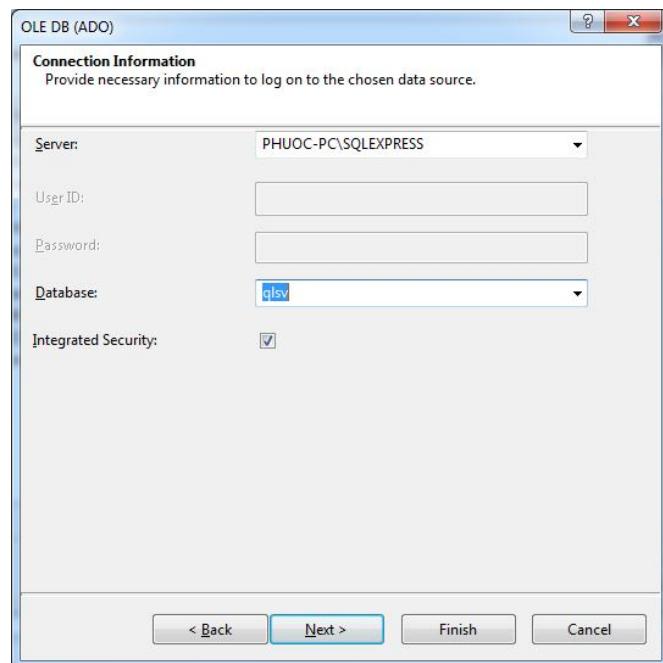
Hình 6.3. Kết nối cơ sở dữ liệu với Report wizard

Chọn vào trình điều khiển cơ sở dữ liệu SQL Server → Chọn Next.



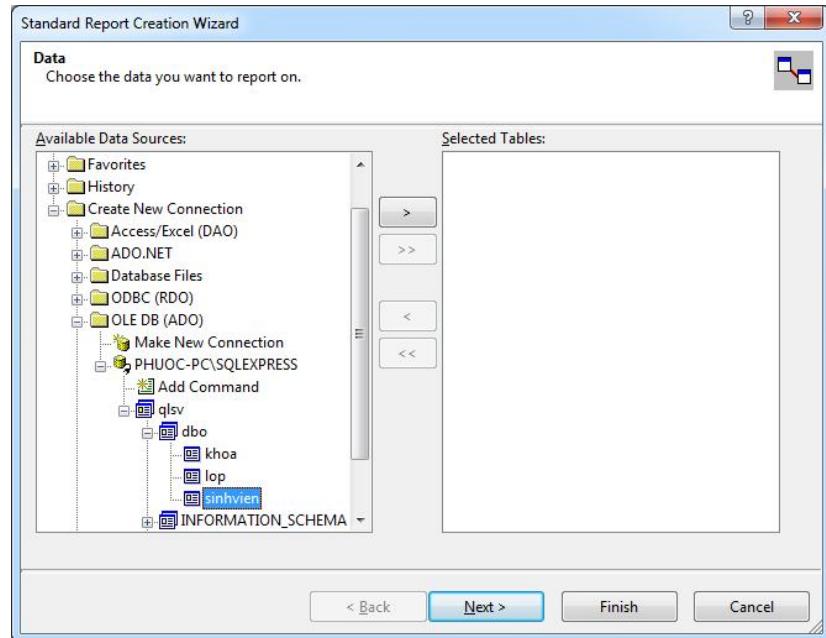
Hình 6.4. Chọn trình kết nối SQL server

Gõ vào các thông số cần thiết cho: Server, UserId, Password, Database → Chọn Finish.



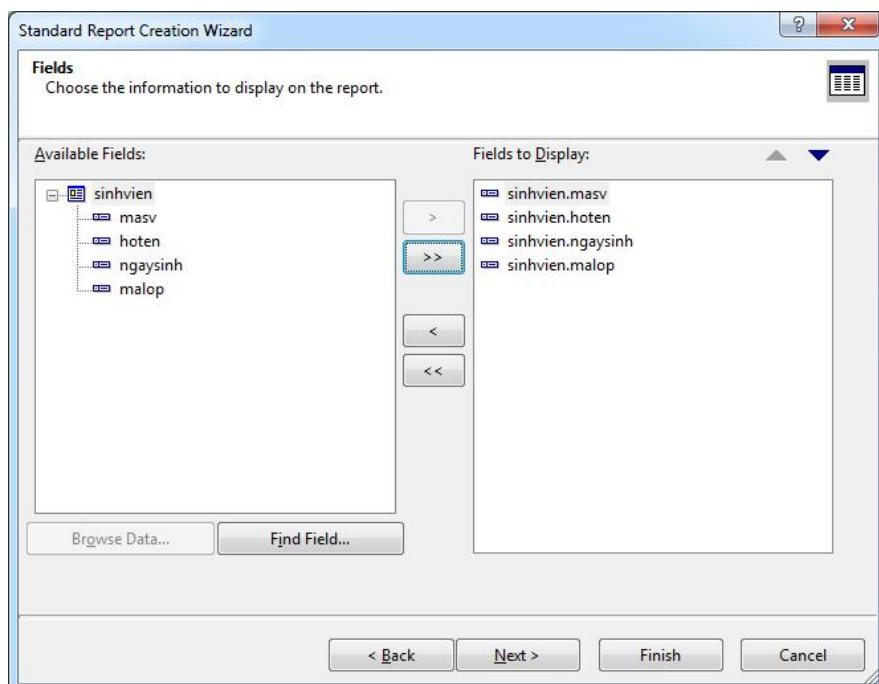
Hình 6.5. Thực hiện các thông số kết nối

Chọn vào Cơ sở dữ liệu cần báo cáo trong Tables hoặc Views, chọn vào Tables hoặc Views rồi chọn nút “>” để chọn tables hoặc Views cần kết xuất báo cáo.



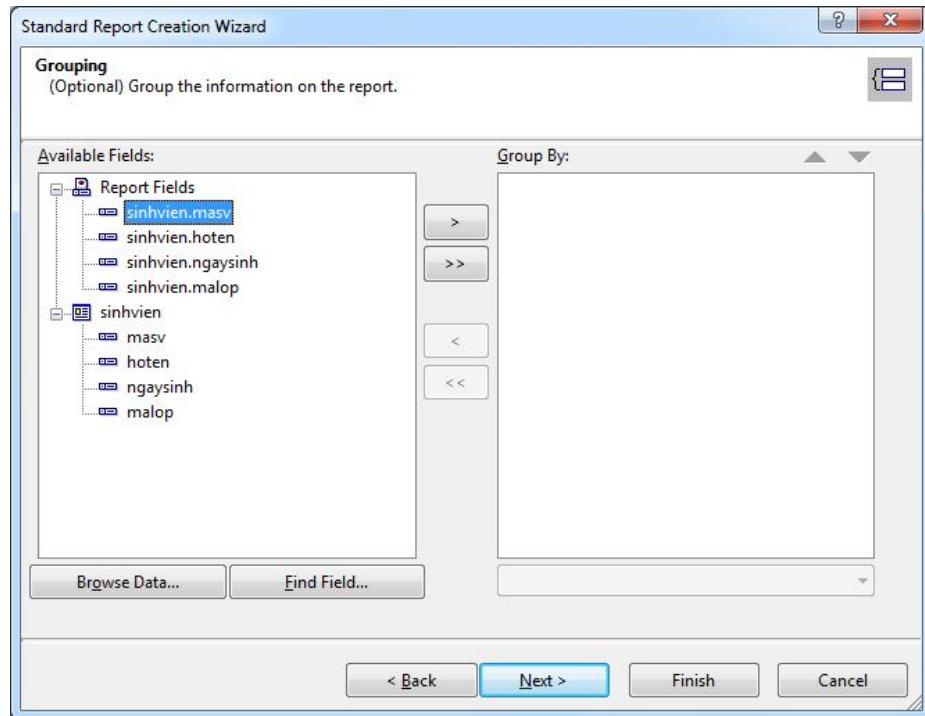
Hình 6.6. Chọn bảng hoặc view cần xuất báo cáo

Chọn những cột dữ liệu cần trình bày trên Report từ ngăn Available Fields, ta nhấn nút “>” hoặc “>>”.



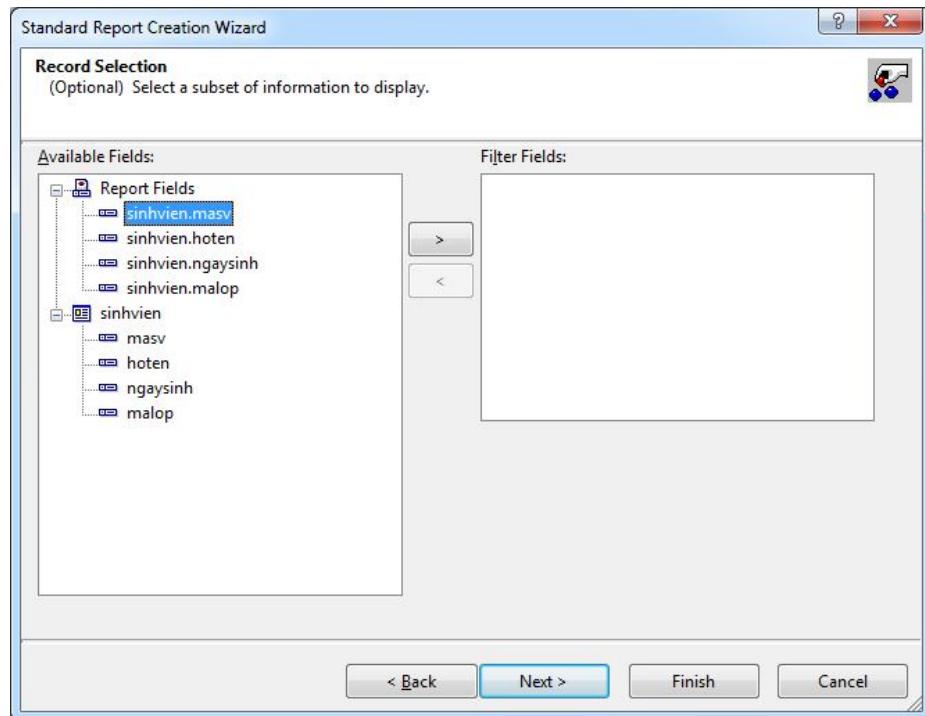
Hình 6.7. Chọn cột cần xuất báo cáo

Chọn Next, ta có thể nhóm dữ liệu theo một cột nào đó.



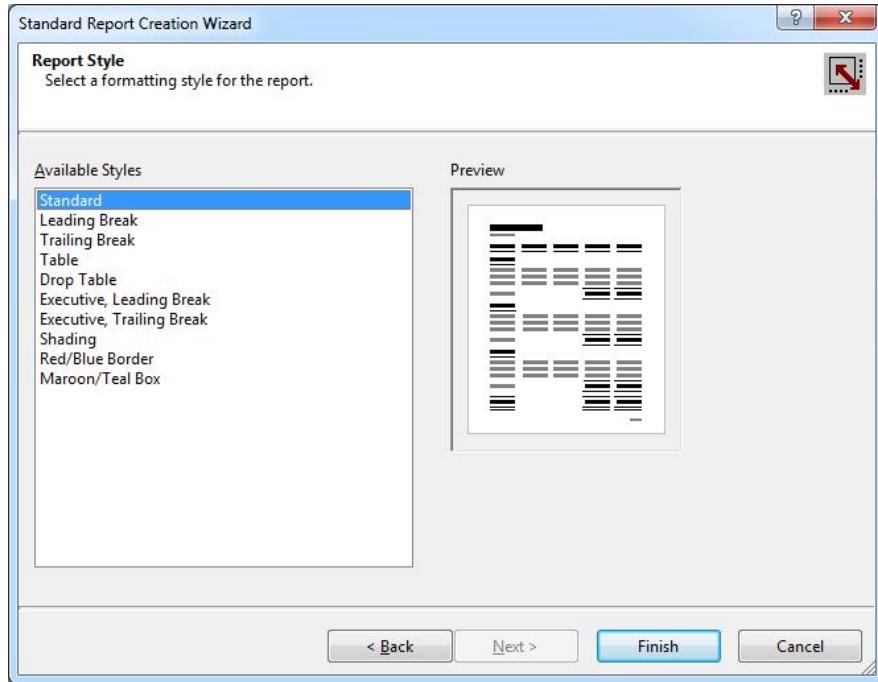
Hình 6.8. Gom nhóm các cột

Chọn Next, ta có thể lọc dữ liệu cho phù hợp với yêu cầu (nên để mặc định).



Hình 6.9. Lọc dữ liệu các cột

Chọn Next, ta có thể khai báo biểu đồ đính kèm theo Report.



Hình 6.10. Chọn biểu đồ cho Report

Cuối cùng, chọn Finish, Report sẽ hiển thị cửa sổ thiết kế, ta có thể chỉnh sửa các tiêu đề, vùng dữ liệu cho phù hợp.

6.3. Tương tác Report từ C#

Trong phần này, chúng ta sẽ tìm hiểu các cách đọc và trình bày dữ liệu trên điều khiển Crystal Report Viewer bằng các hình thức như: Đọc dữ liệu trực tiếp, thông qua đối tượng DataSet hay DataTable, thay đổi quyền truy cập cơ sở dữ liệu,...

Các bước thực hiện như sau.

Tạo file Report (đã có ở phần trước, giả sử là sinhvien.rpt) và thiết kế một form (tên là frmHienthiReport).

Thêm điều khiển Crystal ReportViewer (đặt tên là rv) lên form.

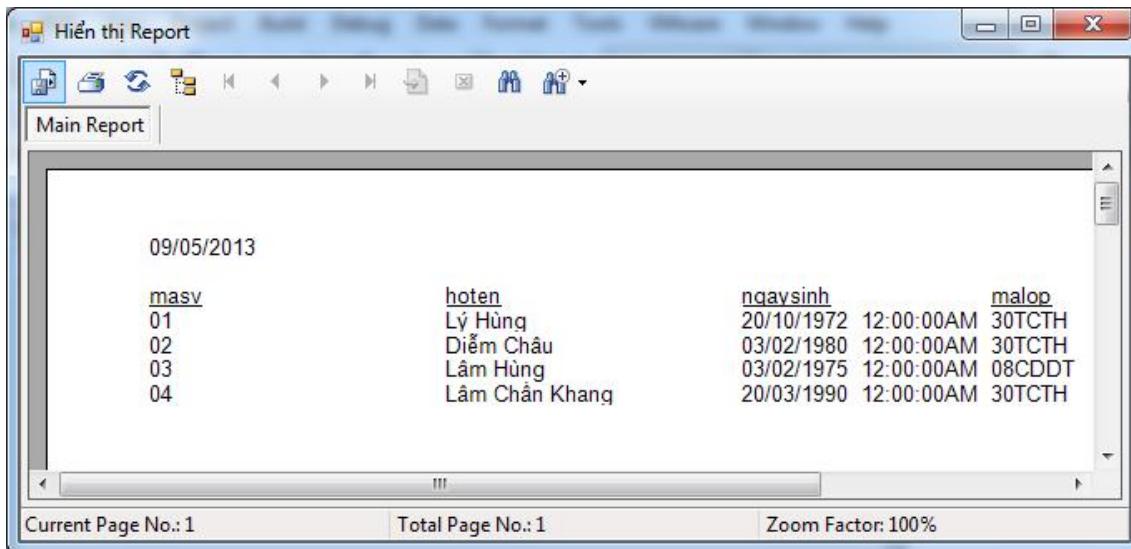
Khai báo đối tượng ReportDocument, sau đó gán đối tượng này vào thuộc tính ReportSource của điều khiển Crystal Report Viewer, **ví dụ 6.1** sẽ minh họa cách làm này (đoạn chương trình này nằm trong sự kiện Form_load).

```
ReportDocument objRep;
objRep = new sinhvien();
rv.ReportSource = objRep;
rv.Refresh();
```

Khi thực thi chương trình, lập tức cửa sổ yêu cầu cung cấp thông tin đăng nhập xuất hiện. Cung cấp các thông tin hợp lệ, lập tức Report sẽ được trình bày.

☞ Lưu ý: trong mỗi form trình bày báo cáo, ta cần khai báo để sử dụng hai namespace như sau.

```
using CrystalDecisions.CrystalReports.Engine;
using CrystalDecisions.Shared;
```



Hình 6.11 Hiển thị Report trên Form

6.4. Cung cấp thông tin đăng nhập

Như trình bày ở phần trên, khi ta triển khai ứng dụng trên một hỗ trợ khác, người sử dụng thông tin đăng nhập khác với thông tin đăng nhập trong khi thiết kế Report, cửa sổ đăng nhập sẽ xuất hiện mỗi khi mở Report.

Để giải quyết vấn đề này, ta sẽ sử dụng đối tượng *ConnectionInfo* và *TableLogOnInfo* để cập nhật thông tin đăng nhập. Kế đến sử dụng phương thức *ApplyLogOnInfo* của đối tượng *ReportDocument* để áp dụng thông tin kết nối cơ sở dữ liệu hiện hành cho Report.

Áp dụng các đối tượng *ConnectionInfo* và *TableLogOnInfo* để cài tiến ví dụ trước. Khi chương trình thực thi sẽ không còn xuất hiện màn hình yêu cầu đăng nhập thông tin.

Ví dụ 6.2: viết lại sự kiện Form_Load của form frmHienthiReport như sau:

```
ReportDocument objrep = new ReportDocument();
ConnectionInfo mycon = new ConnectionInfo();
TableLogOnInfo myinfo = new TableLogOnInfo();
mycon.IntegratedSecurity = true;
mycon.ServerName = "PHUOC-PC\\SQLEXPRESS";
mycon.DatabaseName = "qlsv";
myinfo.ConnectionInfo = mycon;
objrep.Database.Tables[0].ApplyLogOnInfo(myinfo);
rv.ReportSource = objrep;
rv.Refresh();
```

6.5. Điền dữ liệu vào Report từ đối tượng Dataset

Trong những trường hợp trên, Report đều đọc dữ liệu bằng cách kết nối theo hình thức OLE DB (ADO), đồng thời sử dụng đối tượng ConnectionInfo và TableLogonInfo để cập nhật thông tin đăng nhập.

Ngoài ra, ta còn có thể sử dụng đối tượng Dataset (hoặc DataTable) để nắm giữ tập dữ liệu, sau đó sử dụng phương thức SetDataSource để gán đối tượng này vào đối tượng ReportDocument.

Ví dụ 6.3 sẽ minh họa đầy đủ cách đổ dữ liệu từ Dataset vào Report. Chương trình gồm 2 form: form 1 (frmTimkiem) và form 2 (frmHienthiReport) như ở mục 6.3 và 6.4). Form 1 bao gồm một combobox (cblop) dùng để lưu trữ các lớp, một Datagridview (grd) dùng để hiển thị các sinh viên tương ứng với lớp trong cblop, một nút “Xem in” dùng để hiển thị form frmReportHienthi.

☞ **Lưu ý:** Lúc này form frmReportHienthi chỉ hiển thị những sinh viên tương ứng với lớp vừa chọn bên form 1.

Thực thi chương trình, các form hiển thị lần lượt như sau:

	masv	hoten	ngaysinh	malop
▶	01	Lý Hùng	20/10/1972	30TCTH
	02	Điểm Châu	03/02/1980	30TCTH
	04	Lâm Chấn Khang	20/03/1990	30TCTH
*				

Hình 6.12 Minh họa form tìm kiếm

masv	hoten	ngaysinh	malop
01	Lý Hùng	20/10/1972	12:00:00AM 30TCTH
02	Điểm Châu	03/02/1980	12:00:00AM 30TCTH
04	Lâm Chấn Khang	20/03/1990	12:00:00AM 30TCTH

Hình 6.13 Form Report hiển thị các sinh viên tương ứng với lớp được chọn

Source code Form 1:

☞ Lưu ý: biến toàn cục sql (mục đích là để Form 1 và Form 2 cùng hiển) được khai báo trong file Program.cs với cú pháp như sau:

```
public static string sql = "";  
-----  
public partial class frmTimkiem : Form  
{  
    SqlConnection cn;  
    SqlDataAdapter data;  
    DataSet ds;  
    public frmTimkiem()  
    {  
        InitializeComponent();  
    }  
    void ht_lop()  
    {  
        string s = "select * from lop";  
        data = new SqlDataAdapter(s, cn);  
        ds = new DataSet();  
        data.Fill(ds, s);  
        cblop.DataSource = ds.Tables[0];  
        cblop.DisplayMember = "tenlop";  
        cblop.ValueMember = "malop";  
    }  
    void ht_grd()  
    {  
        string s = "select * from sinhvien where malop = '" +  
        cblop.SelectedValue.ToString() + "'";  
        data = new SqlDataAdapter(s, cn);  
        ds = new DataSet();  
        data.Fill(ds, s);  
        grd.DataSource = ds.Tables[0];  
        Program.sql = s;  
    }  
    private void frmTimkiem_Load(object sender, EventArgs e)  
    {  
        cn = new SqlConnection("initial catalog = qlsv; data source =  
        PHUOC-PC\\SQLEXPRESS; integrated security = true");  
        cn.Open();  
        ht_lop();  
        ht_grd();  
    }  
    private void cblop_SelectedIndexChanged(object sender, EventArgs e)  
    {  
        ht_grd();  
    }  
    private void bxemin_Click(object sender, EventArgs e)  
    {  
        frmHienthiReport f = new frmHienthiReport();  
        f.Show();  
    }  
}
```

Source code Form 2:

```

private void frmHienthiReport_Load(object sender, EventArgs e)
{
    ReportDocument objrep = new ReportDocument();
    SqlConnection mycon = new SqlConnection ("initial catalog
= qlsv; data source = PHUOC-PC\SQLEXPRESS; integrated
security = true");
    mycon.Open();
    SqlDataAdapter data = new SqlDataAdapter(Program.sql ,
    mycon);
    DataSet ds = new DataSet();
    data.Fill(ds,Program.sql);
    objrep.SetDataSource(ds.Tables[0]);
    rv.ReportSource = objrep;
    rv.Refresh();
}

```

BÀI TẬP CHƯƠNG VI

Hoàn chỉnh chức năng “xem in” của các bài tập chương 5

1. Form quản lý Khoa hoàn chỉnh (nhập trên textbox)

- a. Giao diện

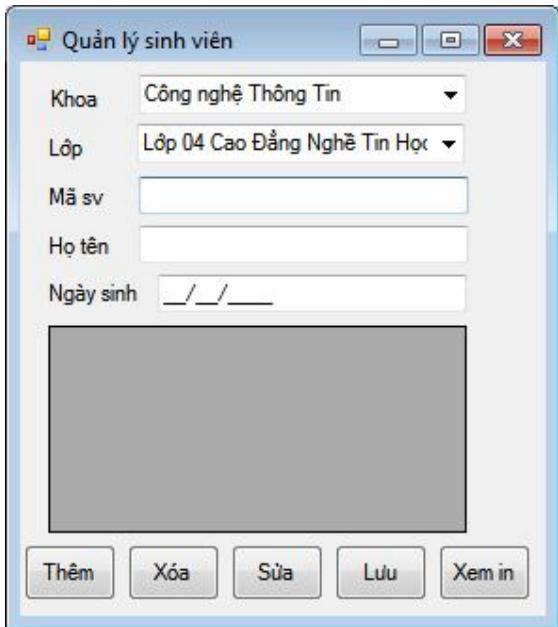


- b. Yêu cầu

- Khi chọn nút “Xem in”: hiển thị form xem in chứa các dữ liệu đang có trên DataGridView.

2. Form quản lý Sinh viên

- a. Giao diện



b. Yêu cầu

- Khi chọn nút “Xem in”: hiển thị form xem in chứa các dữ liệu đang có trên Datagridview.

3. Form quản lý học sinh:

Table **NguoiDung**: quản lý danh sách “người dùng” được phép đăng nhập vào hệ thống, với cấu trúc và dữ liệu tương ứng như sau:

MaND (ID)	TenND (UserName)	MatKhau (Password)	Hoten (FullName)	Diachi (Address)	Email
1	Admin	Admin	Quản trị hệ thống	123 Trương Định	admin@yahoo.com
2	Giaovu	Giaovu	Giáo vụ	140 Lê Trọng Tân	giaovu@yahoo.com
3	Sv01	Sv01	Sinh viên	152/2 Hoàng Văn Thụ	Sv01@yahoo.com

Table **HocSinh**: quản lý danh sách học sinh của chương trình, với các field như sau

STT	Tên field	Kiểu dữ liệu	Ghi chú
	<u>MaHS</u>	Autonumber	Mã học sinh, khóa chính
	Ho	Text	Họ và chữ lót
	Ten	Text	Tên học sinh
	NgaySinh	Date	Ngày sinh
	DiaChi	Text	Địa chỉ
	DienThoai	Text	Điện thoại
	Email	Text	
	TonGiao	Text	Tôn giáo
	DanToc	Text	Dân tộc
	GioiTinh	Integer	Giới tính 0: Nữ 1: Nam
	AnhVan	Integer	Ảnh văn 0: không có 1: có
	PhapVan	Integer	Pháp văn 0: không có 1: có
	HoaVan	Integer	Hoa văn 0: không có 1: có
	HoTenCha	Text	Họ tên cha
	NgheNghiepCha	Text	Nghề nghiệp của Cha
	DiaChiCha	Text	Địa chỉ của cha
	HoTenMe	Text	Họ tên mẹ
	NgheNghiepMe	Text	Nghề nghiệp của mẹ
	DiaChiMe	Text	Địa chỉ của mẹ

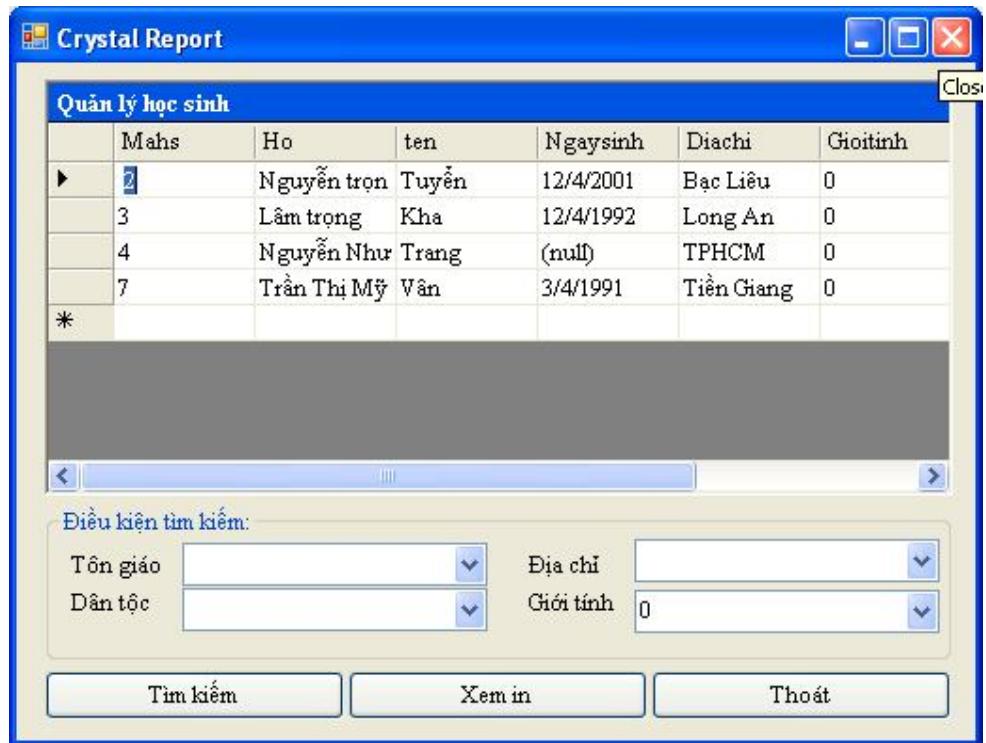
Table DanToc: lưu trữ danh mục dân tộc

MaDT	Ten
K	Kinh
T	Tây
M	Mường
H	H'Mông

Table TonGiao: lưu trữ danh mục tôn giáo

MaTG	Ten
PG	Phật giáo
KG	Không
TC	Thiên chúa giáo
CD	Cao đài
HH	Hòa hảo

4. Giao diện



- Xử lý nút “Tìm Kiếm”: Chọn các tiêu chí tìm kiếm trong “Điều kiện tìm kiếm”.
Hiển thị các mẫu tin theo “Điều kiện tìm kiếm”.
- Xử lý nút “Xem in”: Hiển thị trang in cho các mẫu tin vừa được tìm kiếm.