LẬP TRÌNH TRỰC QUAN C#

GIỚI THIỆU MÔN HỌC

Tên môn: Lập trình trực quan C#

Số tín chỉ: 3 (7LT + 8TH)

Môn học trước:

- Cấu trúc giải thuật,C, Hệ quản trị CSDL

Giờ học(xen kẽ):

- Lý thuyết & Thực hành: Chiều thứ 3,5,7, tiết 6 – 9, phòng A5.602

ĐÁNH GIÁ MÔN HỌC

Điểm chuyên cần: 10%

Điểm kiểm tra giữa kỳ: 20%

Điểm đồ án môn học + thi vấn đáp: 70%

Thang điểm: 10 ***(lấy một chữ số thập phân)***

TÀI LIỆU THAM KHẢO

Sách, giáo trình chính:

- Phạm Hữu Khang, Đoàn Thiện Ngân (2003) Lập trình ứng dụng bằng Visual C#.NET toàn tập(Tập 1, tập 2, tập 3, tập 4) Nhà XB Lao Động Xã Hội.

Sách tham khảo:

- Slide, tài liệu của giáo viên

Quy tắc lớp học

- Điểm danh bất kỳ trong buổi học(không hạn chế số lần)
- Vắng quá 3 buổi→ Cấm thi
- Không nói chuyện, làm việc riêng, dùng điện thoại→ đuổi ra ngoài cảnh cáo, nếu đuổi hẳn → vắng buổi đấy
- Học thầy, học bạn, học hỏi lẫn nhau
- QT1: Không phải giáo viên cái gì cũng biết
- QT2: Cái gì không biết hỏi giáo viên, nếu giáo viên không biết thì xem lại quy tắc 1

NỘI DUNG CHÍNH

- Tổng quan về .NET Framework
- Giới thiệu ngôn ngữ C# và các khái niệm cơ bản
- Lập trình hướng đối tượng với C#
- Lập trình Winforms với C# (Windows Forms và Windows Controls)
- Lập trình Cơ sở dữ liệu (Entity Framework)

TỔNG QUAN VỀ

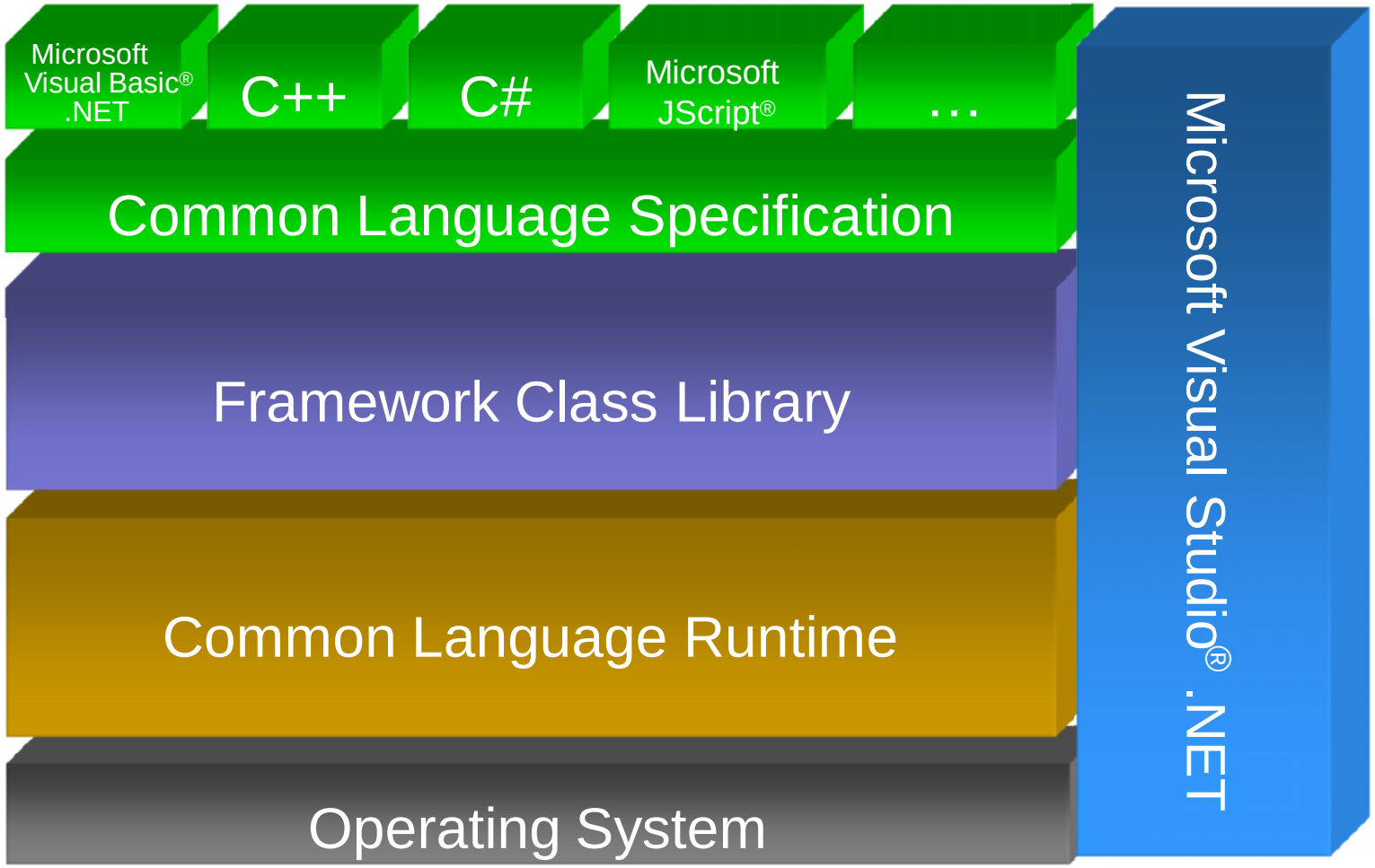
.NET FRAMEWORK

GIỚI THIỆU

- Microsoft .NET gồm 2 thành phần chính:
  - Framework: cung cấp những gì cần thiết và căn bản nhất
  - Integrated Development Environment – IDE: cung cấp môi trường giúp triển khai dễ dàng và nhanh chóng các ứng dụng trên nền tảng .NET

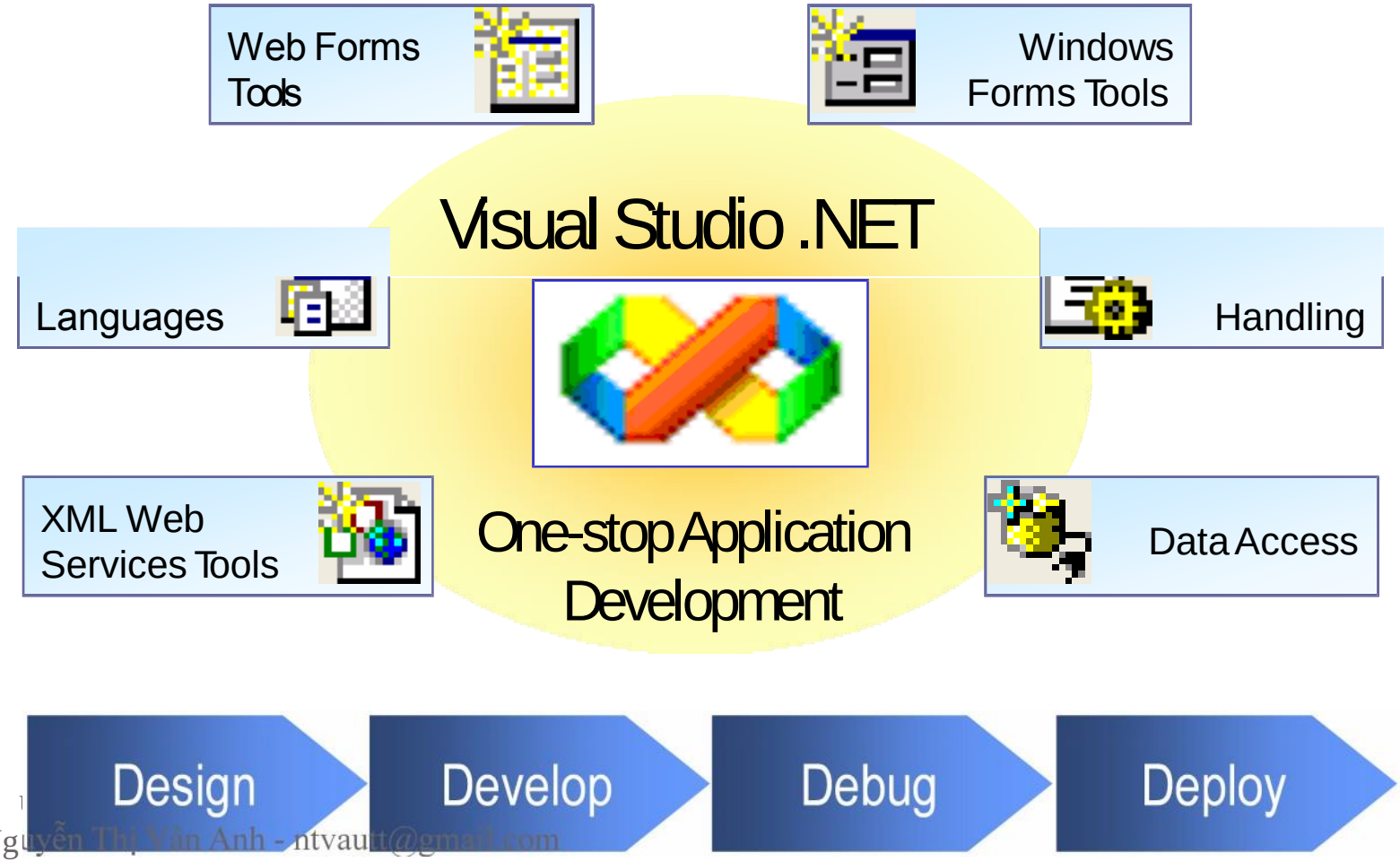
KIẾN TRÚC .NET

- Microsoft **.NET Framework** Architecture



KIẾN TRÚC .NET

- Các đặc điểm của VS .NET



Nguyễn Thị Vân Anh - ntvautt@gmail.com

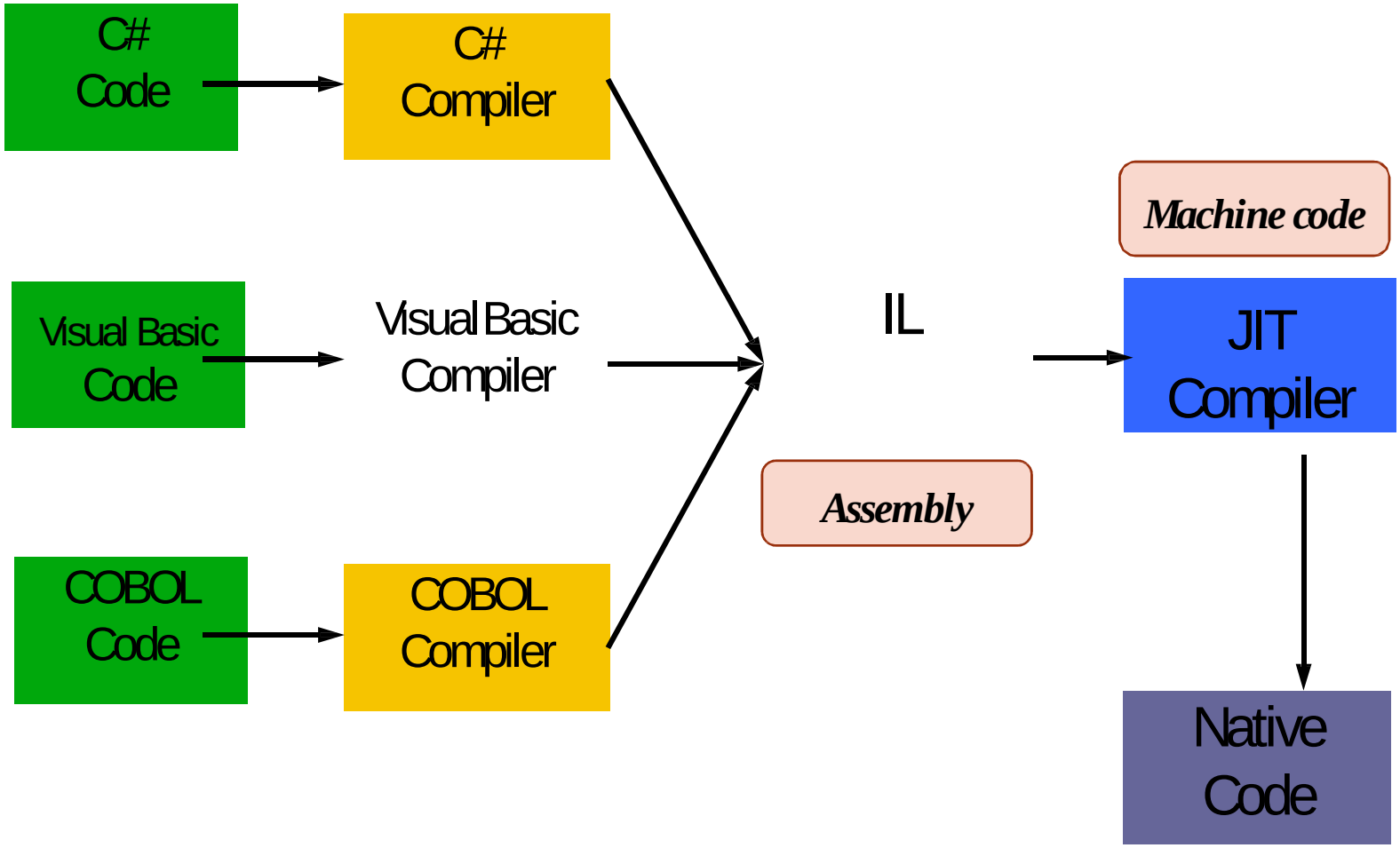
KIẾN TRÚC .NET

- Các đặc điểm của VS .NET

1. Hổ trợ lập trình đa ngôn ngữ.
2. Độc lập với hệ điều hành (Platform).
3. Xây dựng ứng dụng nhanh chóng và dễ dàng.
4. Hổ trợ xây dựng ứng dụng cho nhiều thiết bị .
5. Môi trường thiết kế trực quan .
6. Hướng đến các ứng dụng trên Internet (Webservice, WAP…)

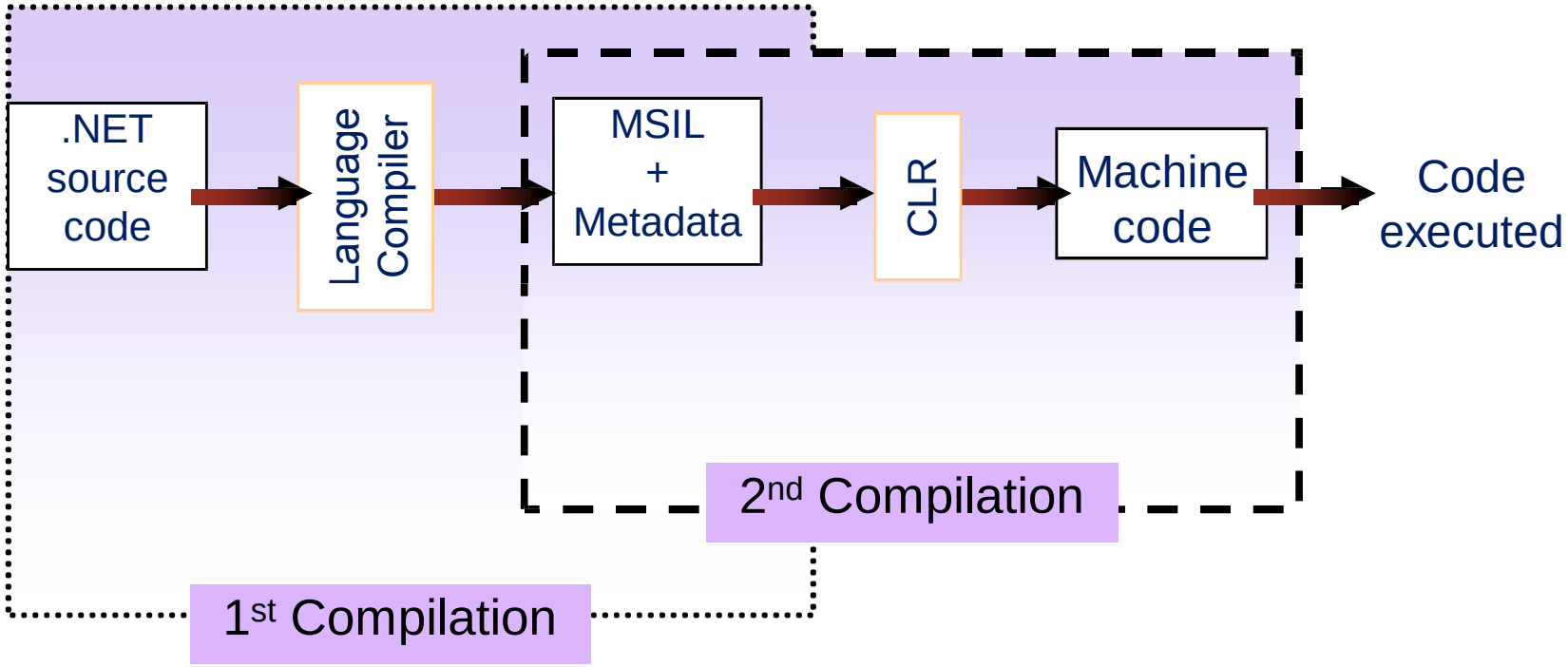
KIẾN TRÚC .NET

- Thực thi một chương trình .NET

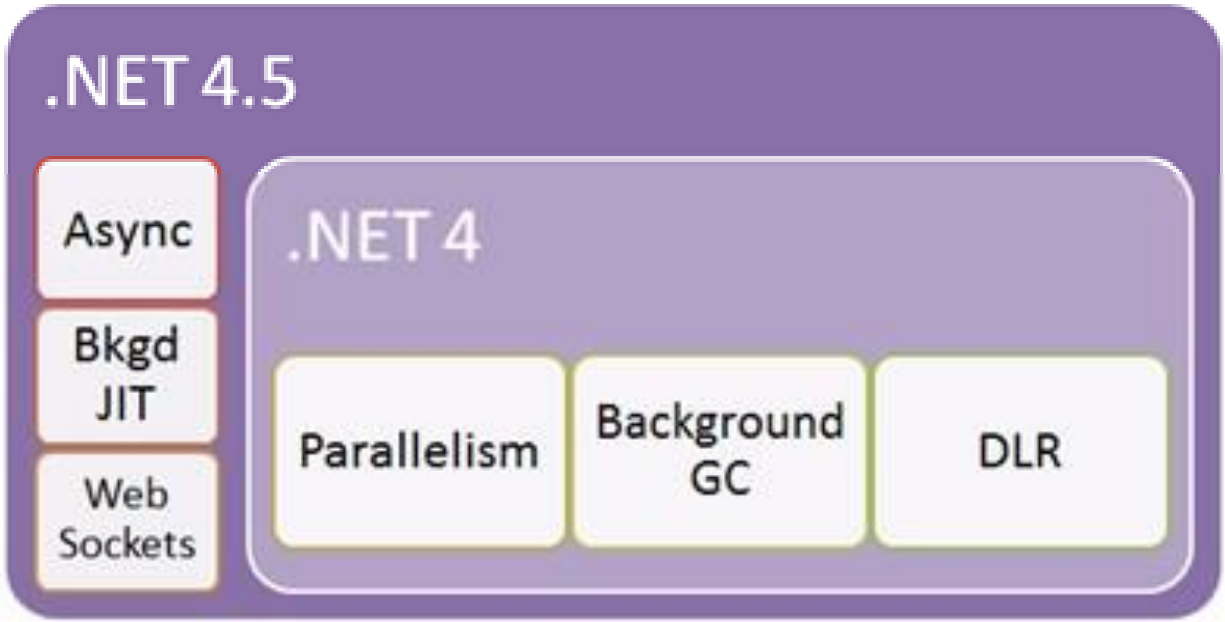


KIẾN TRÚC .NET

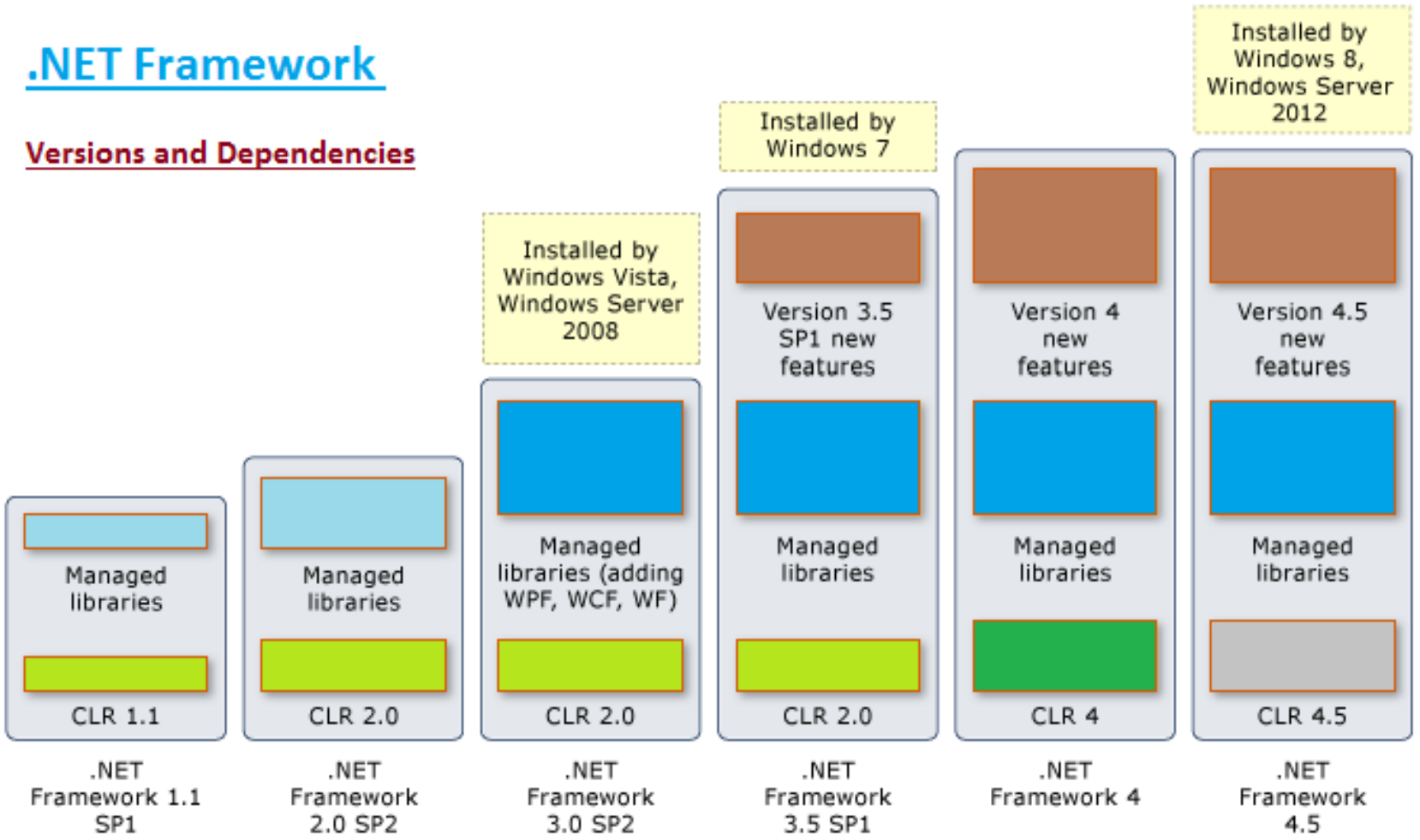
- Thực thi một chương trình .NET



KIẾN TRÚC .NET



KIẾN TRÚC .NET



TỔNG QUAN VỀ C#

GIỚI THIỆU

- Được phát triển bởi Microsoft, do nhóm Ander Hejlsberg và Scott Wiltamuth sáng tác
- Là ngôn ngữ lập trình trực quan, hướng sự kiện, hướng đối tượng
- Dựa trên ý tưởng của các ngôn ngữ: C, C++, Java, Visual Basic
- Hỗ trợ đầy đủ bởi .NET Platform
- Có trình Complier hiệu quả nhất trong .NET family
- Hạn chế khi muốn sử dụng con trỏ

GIỚI THIỆU

- Các ưu điểm của C#
  - Là ngôn ngữ đơn giản
  - Là ngôn ngữ hiện đại
  - Là ngôn ngữ hướng đối tượng
  - Là ngôn ngữ mạnh mẽ và mềm dẻo
  - Là ngôn ngữ có ít từ khóa
  - Là ngôn ngữ hướng module
  - Là ngôn ngữ phổ biến!!! ☺

GIỚI THIỆU

- Để thành thạo C# thì phải làm song song 2 việc
  - Học các tính năng của ngôn ngữ C#
  - Nghiên cứu về các class được cung cấp sẵn trong .NET Framework (Base Class Library – BCL)

VÍ DỤ

- Chương trình C#

```
using System;
public class SampleCSharp
{
    public static void Main(string[] args)
        {
          Console.WriteLine("Hello world");
        }
}
```

Namespace

- Tác dụng:
  - Tránh sự trùng lặp khi đặt tên lớp
  - Quản lý mã dễ dàng
  - Giảm bớt sự phức tạp khi chạy với các ứng dụng khác
  - Namespace **System** bao gồm tất cả các thư viện để tương tác với hệ thống
- Cú pháp

```
namespace Tên_Namespace
{
        //Khai báo các lớp…
 }
```

Namespace

- Có thể khai báo các namespace, class… bên trong namespace khác

Ví dụ 1 : namespace Sample

```
{
    public class A
    {
    }
    public class B
    {
    }
}
```

Ví dụ 2 :

```
namespace Sample_2
{
        public class A
        {
        }
        namspace Sample_3
        {
        }
}
```

Namespace

- Các namespace hoàn toàn **public**
- Không thể *private*, *protected* hay *internal*.

```
...
public namespace Sony //error
{
        ...
}
private namespace Samsung //error
{
        ...
}
...
```

Namespace

- Standard .NET Namespaces

System	Types dealing with intrinsic data, mathematical computations, random number generation, garbage collection, exceptions, attributes
System.Collections System.Collections.Generic	Stock container objects, base types and interfaces used for building customized collections; generics
System.Data System.Data.Odbc System.Data.OracleClient System.Data.OleDb System.Data.SqlClient	ADO.NET for database solutions
System.Diagnostics	Source code debugging and tracing
System.Drawing System.Drawing.Drawing2D System.Drawing.Printing	Types wrapping graphical primitives such as bitmaps, fonts, and icons; printing capabilities

Namespace

- Standard .NET Namespaces

System.IO System.IO.Compression System.IO.Ports	File I/O, buffering, compression, serial ports
System.Net	Network programming, sockets
System.Reflection System.Reflection.Emit	Runtime type discovery, dynamic creation of types
System.Runtime.InteropServices	Interaction with unmanaged code (DLL and COM)
System.Threading	Support for multithreaded applications
System.Web	ASP.NET, XML Web Services
System.Windows.Forms	Windows Forms (GUI for Windows applications)
System.Xml	Interaction with XML data

Namespace

- Ví dụ

```
// Explicitly list the namespaces used by this file.
using System;
using System.Drawing;

class MyApp
{
      public void DisplayLogo()
      {
            // Create a 20_20 pixel bitmap.
            Bitmap companyLogo = new Bitmap(20, 20);
            ...
      }
}
```

Namespace

- Ví dụ

```
// Not listing System.Drawing namespace!
using System;

class MyApp
{
      public void DisplayLogo()
      {
            // Using fully qualified name.
            System.Drawing.Bitmap companyLogo =
                  new System.Drawing.Bitmap(20, 20);
            ...
      }
}
```

Namespace

- Để sử dụng 1 lớp bên ngoài 1 namespace chứa nó, cần phải theo cú pháp sau

namespace. class name

- Ví dụ:

```
namespace Sony
{
  class Television
  {
    ...
  }
  class WalkMan
  {
    ...
    Television MyEntertainment = new Television();
    ...
  }
```

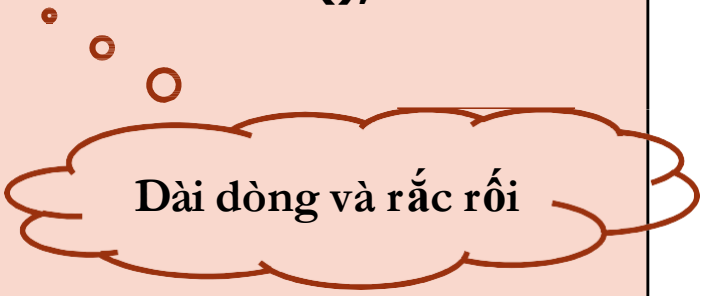
Namespace

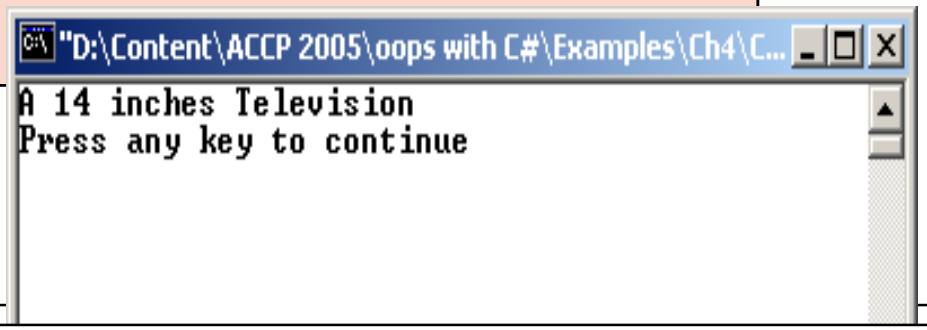
```
using Sony;
using Samsung;
using System;
namespace Sony
{
  namespace Television
  {
    class T14inches
    {
      public T14inches()
      {       Console.WriteLine
("A          14 inches Television");
      }
    }
```

```
    class T21inches
    {
      public T21inches()
      {
        Console.WriteLine
("A 21 inches Television");
      }
    }
  }//end of namespace
  Television
}//end of namespace
Sony
```

Namespace

```
namespace Samsung
{
  class Television
  {
     Sony.Television.T14inches myEntertainment = new
                          Sony.Television.T14inches();
  }
}
class Test
{
    static void Main()
   {
       Samsung.Television myEntertainment = new
                          Samsung.Television();
    }
}
```



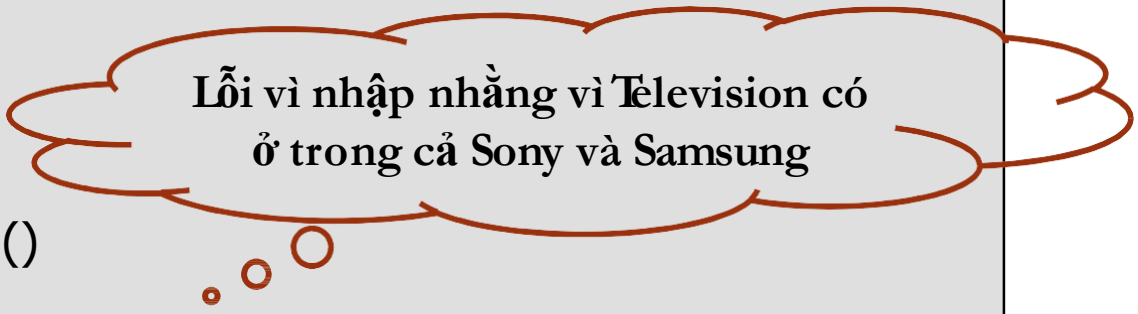


Namespace

- Sử dụng namespace dài dòng gây lúng túng → rút gọn đầy ý nghĩa bằng cách sử dụng các chỉ dẫn namespace khai báo tại phạm vi toàn cục (global scope) trước bất kỳ một khai báo thành viên nào
- Ví dụ

```
using Sony.Television;
T14inches Television = new T14inches();
T21inches Television2 = new T21inches();
```

```
using Sony;
using Samsung;
class Test
{
   static void Main()
   {
            Television MyEntertainment = new Television();
   }
}
```



Namespace

- Bắt buộc phải sử dụng qualified namespace trong trường hợp nhập nhằng này

```
using Sony;
using Samsung;
class Test
{
   static void Main()
   {
            Samsung.Television MyEntertainment = new
                                        Samsung.Television();
   }
}
```

Alias directives

- Các chỉ dẫn bí danh (Alias directives)

```
using T21inches = Sony.Televisions.T21inches;
class Test
{
   static void Main()
   {
            T21inches M = new T21inches();
   }
}
```

- Cú pháp

using alias_name = fully qualified path to the namespace or class

Lớp và đối tượng

- Lớp được coi là 1 kiểu dữ liệu *hay* 1 kiểu dữ liệu trong C# được định nghĩa là 1 lớp
- Các thể hiện riêng của từng lớp (class) gọi là đối tượng (object)
- Hai thành phần chính khi định nghĩa lớp
  - Thuộc tính (tính chất của đối tượng)
  - Phương thức (hành động ứng xử của đối tượng)

Lớp và đối tượng

- Cú pháp định nghĩa lớp

```
class Tên_lớp
 {
        //khai báo các thành phần
        //khai báo các phương thức
 }
```

- Ví dụ

```
class KhachHang
 {
   private int mMaKhachHang;
   private string mTenKhachHang;
   public void In()    {…}
```

Các kiểu dữ liệu cơ bản

- Các kiểu dữ liệu đơn giản: int, float, string , char, bool..
- Các kiểu dữ liệu tham chiếu: đối tượng, lớp.
- Các bổ từ khi sử dụng với biến: private, public, protected
- Các kiểu dữ liệu khác: Array, Struct, Enum...
- Giá trị mặc định cho các kiểu dữ liệu:

Kiểu dữ liệu	Giá trị mặc định
Numeric (int, float, short…)	0
Bool	False
Char	'\0'
Enum	0
Reference	null

Các kiểu dữ liệu cơ bản

C# Data type	Description	Example
object	Base data type for all other types.	object o = null;
string	Declares a variable that can store string values	string s = “hello”;
int	Declares a variable that can store integer values.	int val = 12;
byte	Declares a variable that can store byte integer values.	byte val = 12;
float	Declares a variable, which can store real number values that have integer and decimals parts	float val = 1.23F;
bool	Declares a variable, which can store real number values that have integer and decimals parts	bool val1 = true; bool val2 = false;
char	Declares a variable, which can store char values.	char val = 'h';

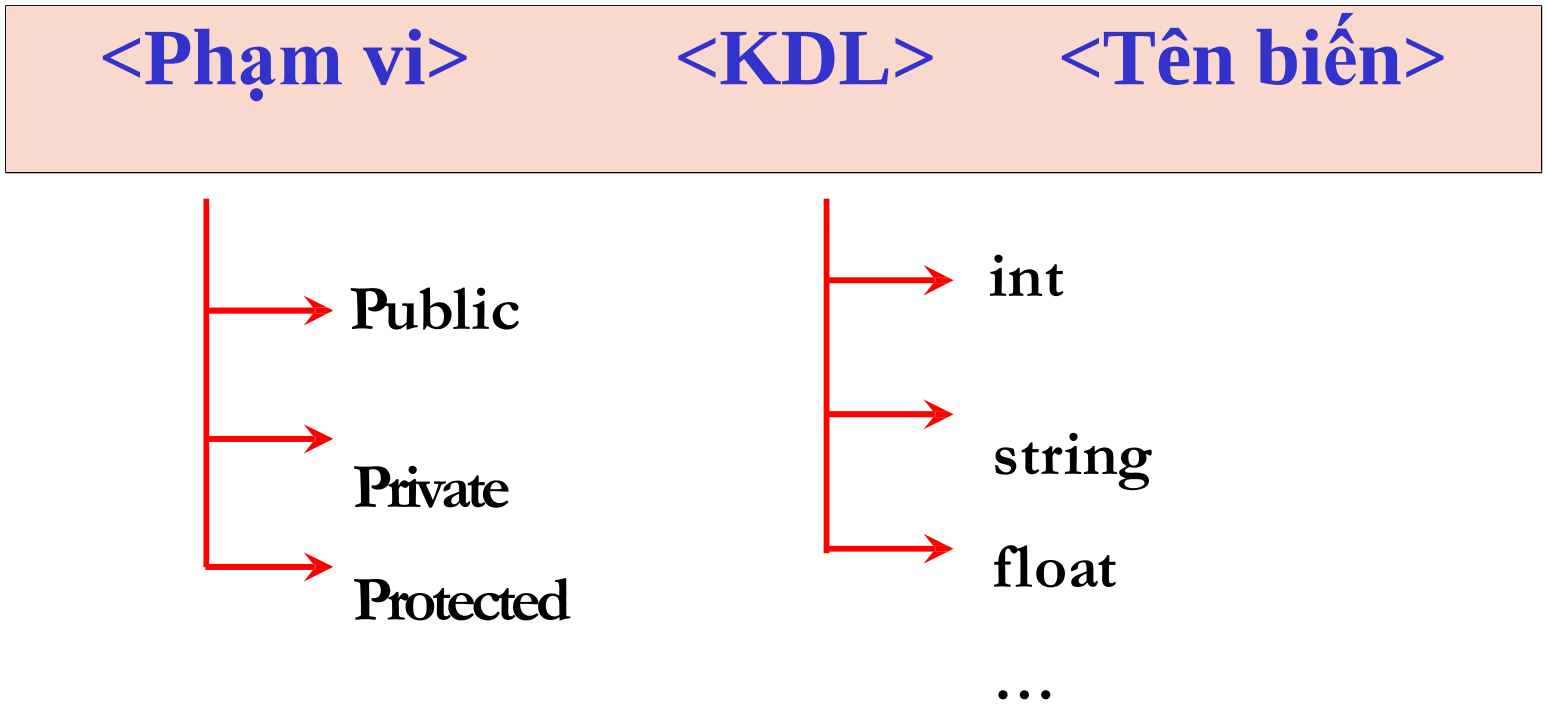
Các kiểu dữ liệu cơ bản

- Common Type System (CTS)

CTS Data Type	VB.NET	C#	Managed C++
System.Byte	Byte	byte	unsigned char
System.SByte	SByte	sbyte	signed char
System.Int16	Short	short	short
System.Int32	Integer	int	int or long
System.Int64	Long	long	__int64
System.UInt16	UShort	ushort	unsigned short
System.UInt32	UInteger	uint	unsigned int or unsigned long
System.UInt64	ULong	ulong	unsigned __int64
System.Single	Single	float	Float
System.Double	Double	double	Double
System.Object	Object	object	Object^
System.Char	Char	char	wchar_t
System.String	String	string	String^
System.Decimal	Decimal	decimal	Decimal
System.Boolean	Boolean	bool	Bool

Khai báo biến, thuộc tính

- Cú pháp khai báo biến trong C#



Khai báo biến, thuộc tính

- Cú pháp khai báo biến trong C#
  - Ví dụ

```
Type  variable_name [=initialization] ;
```

```
int a;
int  b = 3;
int  _999; // hợp lệ
int  123_a; // không hợp lệ
```

Khai báo biến, thuộc tính

- Để sử dụng 1 từ khóa làm tên 1 biến, thêm tiền tố '@' trước tên biến
- Ví dụ

```
using System;
class VariableDemo
{
        public static void Main()
        {
        string @string;
        @string = "string is a keyword but used as a
        variable name    in this example";
        Console.WriteLine (@string);
        }
}
```

Nhập / Xuất đơn giản

- Sử dụng phương thức của lớp Console trong namspace System.
- Các phương thức thường được sử dụng là:
  - Console.WriteLine(): hiển thị kết quả ra màn hình và xuống dòng
  - Console.Write() : hiển thị kết quả ra màn hình
  - Console.ReadLine(): đọc 1 chuỗi ký tự từ bàn phím
  - Console.Read (): đọc 1 ký tự từ bàn phím

Nhập / Xuất đơn giản

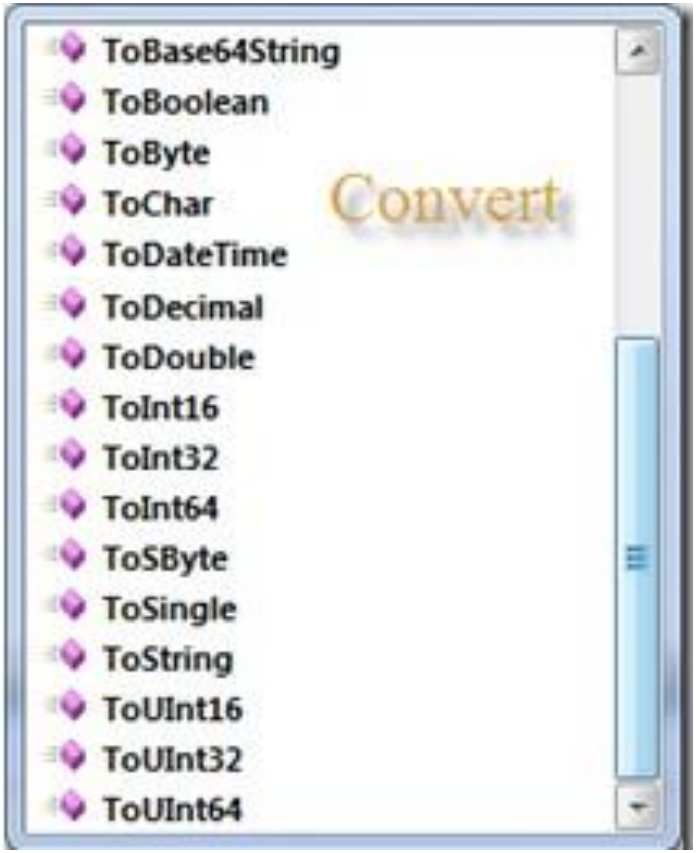
```
using System;
class SampleCSharp
{
    static void Main(string[] args)
            {
              //Nhập vào 1 chuỗi
              string s = Console.ReadLine();

              //Nhập vào số nguyên
              int n = int.Parse(Console.ReadLine());

              //In ra màn hình
              Console.WriteLine("s = {0} va n = {1}", s, i);
            }
}
```

Nhập / Xuất đơn giản

- Hàm Console.Readline() trả về kiểu string.
- Chuyển đổi giá trị để tính toán qua lớp Convert:



Nhập / Xuất đơn giản

- Nhập 1 số trong C#:
- Cú pháp: <KDL số> <Biến số> = <KDL số>.Parse (Console.Readline());

```
Console.WriteLine("Nhap 1 so nguyen:");
int so =  int.Parse(Console.ReadLine());
Console.Write("So vua nhap la: ");
Console.WriteLine(so);
```

Nhập / Xuất đơn giản

- Ví dụ:

```
string name = "";  int age = 0;
//Nhap ten
Console.Write("Name: ");
name = Console.ReadLine();
Console.Write("Age: ");
//Ep kieu ve kieu int
age = Convert.ToInt32(Console.ReadLine());
//Xuat thong tin
Console.WriteLine("Name: {0}, Age: {1}", name, age);
Console.ReadLine();
```

Cấu trúc điều khiển

- Cấu trúc rẽ nhánh **if**
  - Cú pháp

```
if (biểu thức điều kiện)
{
        <Khối lệnh thực hiện khi điều kiện đúng>
}
        [else
{
        <Khối lệnh thực hiện khi điều kiện sai>
}]
```

Tương tự ngôn ngữ C, C++!

Cấu trúc điều khiển

- Cấu trúc rẽ nhánh **if**
  - Toán tử điều kiện **?:**

Biến = <biểu thức điều kiện> ? <Lệnh thực hiện khi điều kiện đúng> : <Lệnh thực hiện khi điều kiện sai>

```
string  ketqua = (i % 2) == 0 ? “chan” : “le” ;
```

Cấu trúc điều khiển

- Cấu trúc lựa chọn **switch**
  - Cú pháp

```
switch (biểu thức)
{
        case <giá trị 1>:
                <Các câu lệnh thực hiện 1>
                <Lệnh nhảy>
        case <giá trị 2>:
                <Các câu lệnh thực hiện 2>
                <Lệnh nhảy>
                        …
        [default:
                <Các câu lệnh thực hiện mặc định>]
}
```

Cấu trúc điều khiển

- Cấu trúc lặp **while**
  - Cú pháp

```
while (biểu thức điều kiện)
{
        <Khối lệnh thực hiện>
}
```

Cấu trúc điều khiển

- Cấu trúc lặp **do…while**
  - Cú pháp

```
do
{
        <Khối lệnh thực hiện>
}
while (biểu thức điều kiện)
```

Cấu trúc điều khiển

- Cấu trúc lặp **for**
  - Cú pháp

```
for ([biểu thức khởi tạo] ; [bt điều kiện] ; [bước lặp])
{
        [<Khối lệnh thực hiện>]
}
```

Cấu trúc điều khiển

- Cấu trúc lặp **foreach**
  - Cú pháp

```
foreach (<kiểu tập hợp>  <tên truy cập thành
ph
{

}
```

```
int[]  intArray = {1,2,3,4,5,6,7};

foreach (int item in intArray)
{
        Console.Write ("{0}", item);
}
```

Mảng

- Một nhóm các giá trị cùng kiểu dữ liệu
- Là kiểu tham chiều và do đó được lưu trữ trong Heap
- Cú pháp khai báo

```
DataType[] ArrayName;
```

```
int[] array1;
```

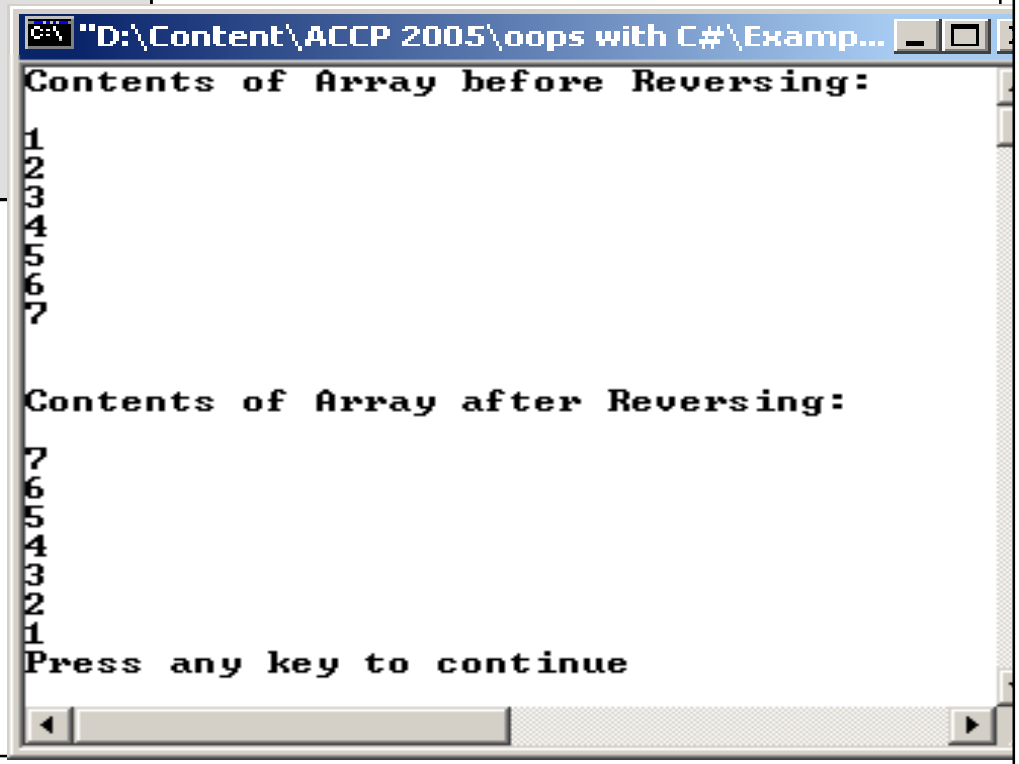
System.Array

```
using System;
class Test
{
   static void Main()
   {
       int[] arrayToReverse= {1,2,3,4,5,6,7};

      Console.WriteLine ("Contents of Array before Reversing:\n");
      displayArray (arrayToReverse);
      Array.Reverse (arrayToReverse);
      Console.WriteLine("\n\nContents of Array after Reversing:\n");
      displayArray (arrayToReverse);
   }
```

System.Array

```
public static void displayArray(Array myArray)
{
    foreach(int arrValue in myArray)
    {
     Console.WriteLine (arrValue);
    }
}
}
```



System.Array

THUỘC TÍNH	MÔ TẢ TÁC DỤNG
IsFixedSize	Kiểm tra xem mảng có độ dài cố định không
IsReadOnly	Kiểm tra xem mảng có readonly không
Length	Lấy ra tổng số phần tử của mảng (số nguyên 32bit)
LongLength	Lấy ra tổng số phần tử của mảng (số nguyên 64bit)
Rank	Lấy ra số chiều của mảng. Ví dụ mảng 1 chiều rank = 1, mảng 2 chiều rank = 2…

System.Array

PHƯƠNG THỨC	MÔ TẢ TÁC DỤNG
BinarySearch	Tìm kiếm nhị phân trong mảng 1 chiều đã được sắp xếp, sử dụng giao tiếp Icomparable
Clear	Thiết lập các phần tử của mảng về giá trị mặc định
Clone	Tạo 1 bản sao của mảng
Copy	Sao chép 1 mảng từ 1 mảng khác
CreateInstance	Tạo 1 thể hiện của lớp mảng
Equals	Kiểm tra xem 2 đối tượng có bằng nhau không

System.Array

PHƯƠNG THỨC	MÔ TẢ TÁC DỤNG
Exist	Kiểm tra xem trong mảng có phần tử nào thỏa mãn điều khiện cho trước không
Finalize	cho phép 1 đối tượng cố giải phóng tài nguyên trước khi GC đòi lại
Find	Tìm kiếm 1 phần tử thỏa mãn điều kiện cho trước và trả về vị trí đầu tiên tìm thấy
FindAll	Trả về tất cả các thành phần thỏa mãn điều kiện cho trước
FindIndex	Tìm 1 phần tử thỏa mãn điều kiện và trả về chỉ số (tính từ 0) của phần tử
FindLast	Tìm kiếm 1 phần tử thỏa mãn điều kiện cho trước và trả về vị trí cuối cùng tìm thấy

System.Array

PHƯƠNG THỨC	MÔ TẢ TÁC DỤNG
FindLastIndex	Tìm kiếm 1 phần tử thỏa mãn điều kiện cho trước và trả về chỉ số cuối cùng tìm thấy
GetLength	Lấy ra số phần tử (kiểu int 32bit) trong 1 chiều cụ thể của mảng
GetLowerBound	Lấy ra chỉ số của phần tử đầu tiên trong 1 chiều cụ thể của mảng
GetUpperBound	Lấy ra chỉ số của phần tử cuối cùng trong 1 chiều cụ thể của mảng
GetValue	Lấy ra giá trị tại vị trí cụ thể trong mảng 1 chiều
IndexOf	Tìm kiếm 1 đối tượng đặc biệt và trả về chỉ số của phần tử đầu tiên

System.Array

PHƯƠNG THỨC	MÔ TẢ TÁC DỤNG
Resize	Thay đổi số phần tử của mảng 1 chiều thành kích thước mới
Reverse	Đảo 1 chuỗi các phần tử trong mảng 1 chiều
SetValue	Thiết lập giá trị cho phần tử ở vị trí cụ thể của mảng 1 chiều
Sort	Sắp xếp mảng 1 chiều...
ToString	Trả về 1 chuỗi biểu diễn đối tượng hiện tại
TrueForAll	Kiểm tra xem mọi phần tử trong mảng có thỏa mãn điều kiện cho trước không

Cấu trúc

- Kiểu người dùng định nghĩa
- Có thể định nghĩa các phương thức bên trong cấu trúc
- Không thể thực thi kế thừa
- Biểu diễn kiểu tham trị

```
…
struct structEx
{
    public int structDataMember;

    public void structMethod1()
    {
        //structMethod1 Implementation
    }
}
…
```

Kiểu liệt kê - enum

```
public class Holiday
{
    public enum WeekDays
    {
     Monday,
     Tuesday,
     Wednesday,
     Thursday,
     Friday
    }
    public void GetWeekDays (String EmpName, WeekDays DayOff)
    {
        //Process WeekDays
    }
    static void Main()
    {
      Holiday myHoliday = new Holiday();
      myHoliday.GetWeekDays ("Richie",Holiday.WeekDays.Wednesday);
    }
}
```

Kiểu liệt kê

- Kiểu liệt kê trong C# có thể kết hợp với các giá trị
- Mặc định, phần tử đầu tiên của kiểu liệt kê được gán giá trị 0 và được tăng cho mỗi thành phần liên tiếp của kiểu liệt kê
- Các giá trị mặc định có thể được ghi đè (gán giá trị mới) trong quá trình khởi tạo

```
public enum WeekDays
{
        Monday=1,
        Tuesday=2,
        Wednesday=3,
        Thursday=4,
        Friday=5
}
```

Boxing và Unboxing

- Trong C# kiểu dữ liệu được chia thành 2 loại là kiểu tham trị (value types) và kiểu tham chiếu (reference types)
- Để chuyển đổi 2 kiểu tham trị và kiểu tham chiếu ta có thuật ngữ Boxing (Bao: tham trị → tham chiếu) và Unboxing (Không bao: tham chiếu → tham trị).

Boxing và Unboxing

```
...
class BoxEx
{
   int objsTaker(object objectX)
   {
    //objsTaker takes an object
    //and processes it here
   }
   object objsConverter()
   {
    //objsConverter does the processing
//and returns an object
   }
}
...
//Implementation of code
int variable1;
variable1 = 5;
BoxEx boxVar = new BoxEx();
boxVar.objsTaker(variable1); //line 1
int convertVar = (int) boxVar.objsConverter(); //line 2
...
```

Dịch và thực thi chương trình C#

- Dịch và thực thi 1 chương trình Console Application trong C#
  - Chọn **File → New → Project** để tạo 1 dự án mới
  - Chọn **Visual C# Projects** từ danh sách bên phải
  - Chọn **Console Application** từ danh sách bên trái
  - Nhập tên dự án vào phần **Name**, ví dụ *Example1*.
  - Trong phần **Solution Explorer** đổi tên *ConsoleApplication1* thành *Chapter1*.
  - Đổi tên dự án thành *Exampe1* và *Class1* thành *HelloWorldDemo.cs*.
  - Lưu tệp và chọn **Build → Build Solution**
  - Chọn **Start without Debugging** từ thực đơn **Debug** để thực thi ứng dụng.

Dịch và thực thi chương trình C#

- Một cách khác để dịch và thực thi Console Application là sử dụng Notepad và DOS
  - Bước 1: Soạn thảo code trong Notepad
  - Bước 2: Lưu tệp với phần mở rộng là **.cs**
  - Bước 3: Chuyển vào DOS và gõ dòng lệnh

```
csc <filename>.cs
```

  - Để chạy tệp C#, gõ tên tệp mà không có phần mở rộng

Quản lý lỗi và ngoại lệ

- Khi các đoạn mã chương trình phát sinh các lỗi. Nếu dùng biến toàn cục để nhận các giá trị lỗi khác nhau từ các phương thức khác nhau thì sẽ không hiệu quả
- Sử dụng các ngoại lệ: ý tưởng đón bắt ngoại lệ bằng việc phân cách mã thực thi và mã xử lý lỗi
- Hai bước:
  - Viết mã chương trình bên trong khối **try**.
  - Viết 1 hoặc nhiều bộ xử lý **catch** ngay sau khối **try**.

```
try
{
        <Khối lệnh>
}
catch (<Ngoại lệ>  <Tên biến>)
{
        //Xử lý ngoại lệ tại đây
```

Quản lý lỗi và ngoại lệ

- Các lỗi khác nhau sẽ đưa ra các ngoại lệ khác nhau

```
try
{
        <Khối lệnh>
}
catch (<Ngoại lệ 1>  <Tên biến 1>)
{
        //Xử lý ngoại lệ 1 tại đây
}
catch (<Ngoại lệ 2>  <Tên biến 2>)
{
        //Xử lý ngoại lệ 2 tại đây
}
.....
```

Quản lý lỗi và ngoại lệ

- Ví dụ:

```
try
{
        result.Text = answer.ToString();
}
catch (System.FormatException ex)
{
        //Xử lý lỗi tại đây
}
```

Quản lý lỗi và ngoại lệ

- Một số ngoại lệ
  - Ngoại lệ tổng quát: System.Exception
  - Ngoại lệ định dạng: System.FormatException
  - Ngoại lệ phép chia 0: System.DivideByZeroException
  - Ngoại lệ tràn số học: OverflowException
  -

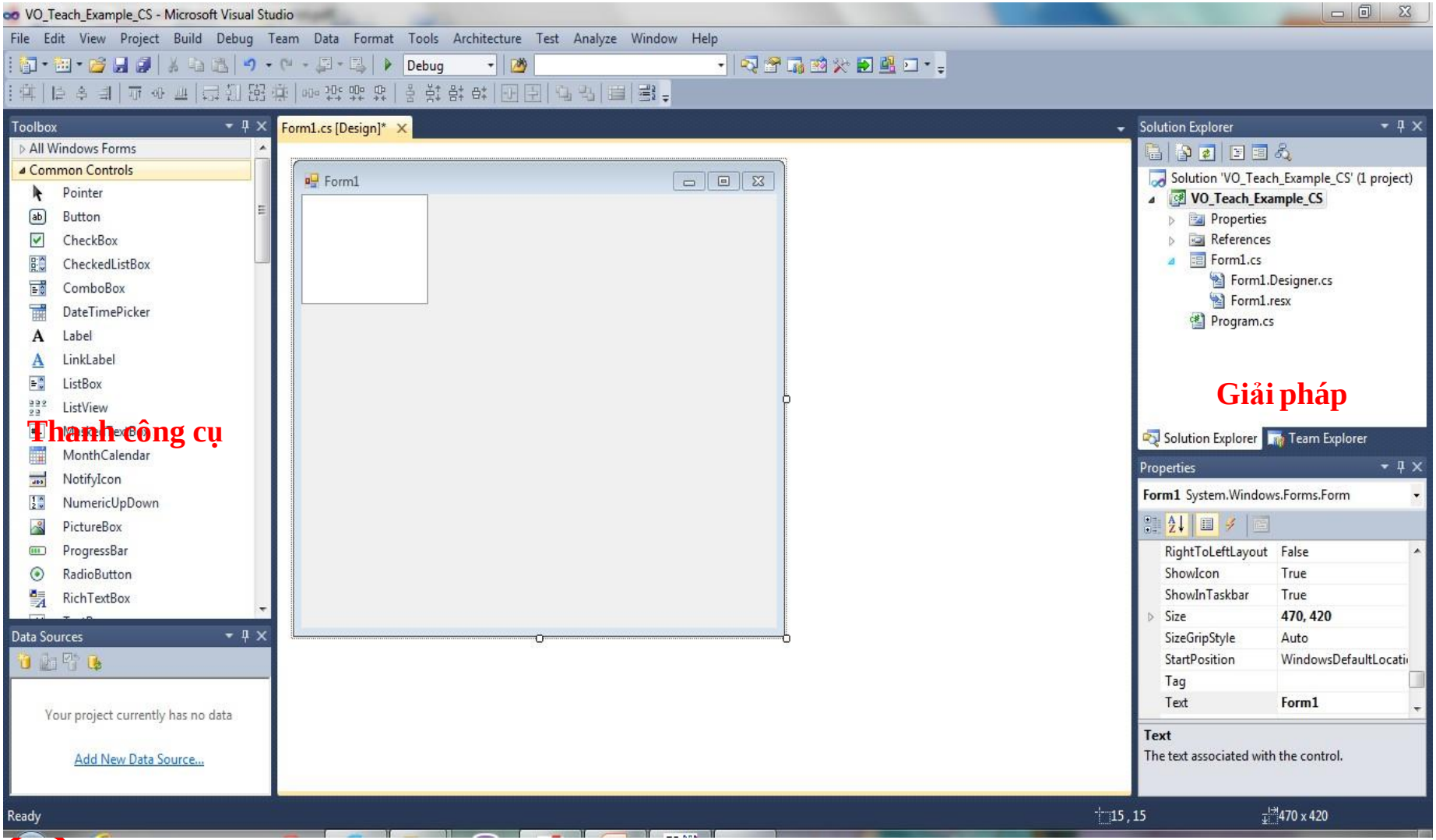
LẬP TRÌNH HĐT TRONG C#

- Hàm tạo trong C# (Constructors)
- Hàm hủy trong C# (Destructors)
- Hoạt động của Bộ thu gom rác (Garbage Collector)
- Nạp chồng phương thức (Method Overloading)
- Nạp chồng toán tử (Operator Overloading)
- Kế thừa trong C# (Inheritance)
- Ghi đè (Overriding)
- Đa hình (Polymorphism)
- Hàm ảo (Virtual Functions)
- Interfaces

LẬP TRÌNH WINFORM

winform

Visual Studio .NET

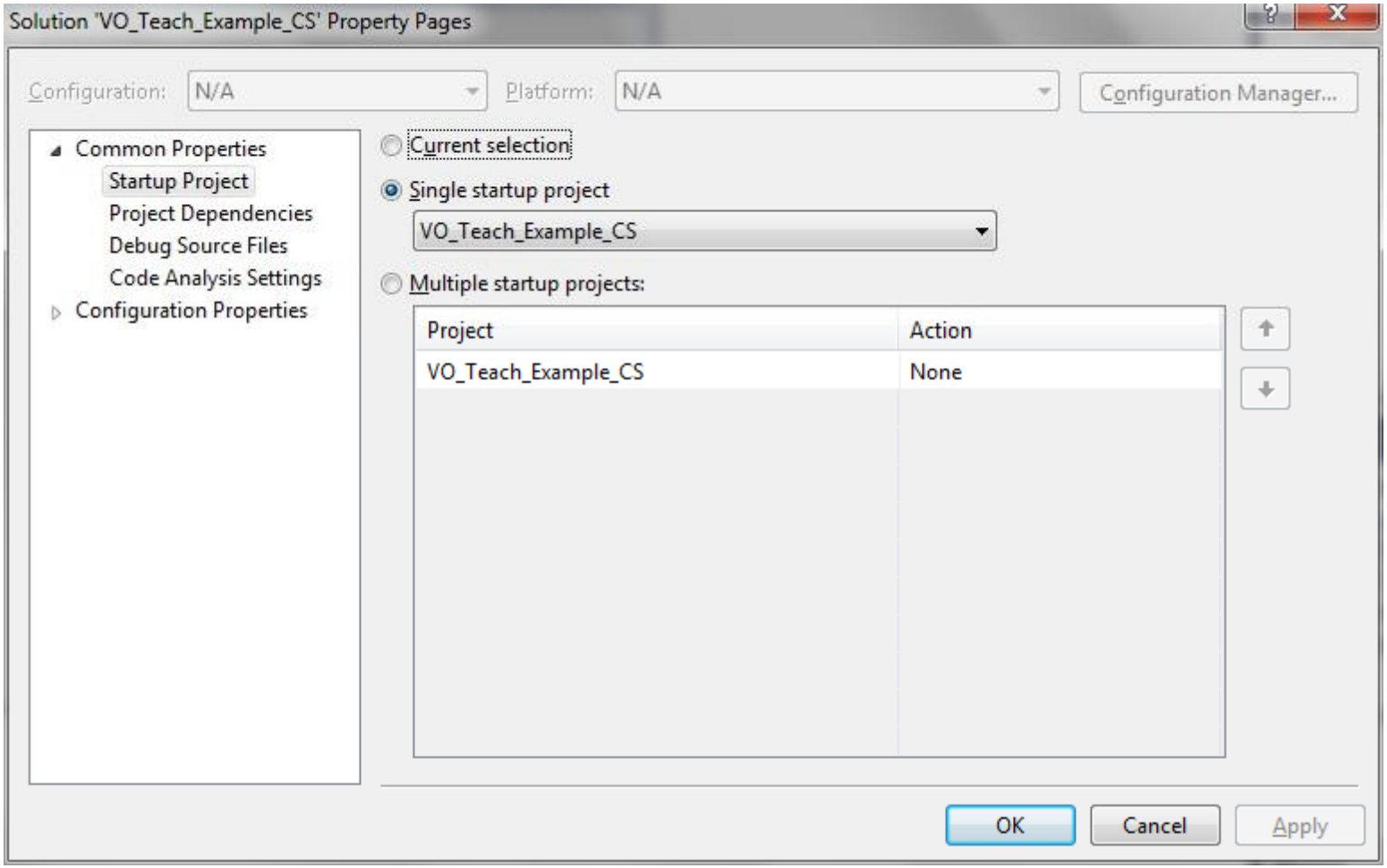


Visual Studio .NET

- Solution (Giải pháp)
  - Là 1 khối công việc nền tảng trong .NET
  - Một Solution có thể chứa nhiều kiểu dự án khác nhau (thường các dự án con này đều liên quan đến 1 ứng dụng cụ thể)
  - Một Solution có thể biên dịch 1 lần chung với những dự án con
  - Solution Explorer: có dạng cấu trúc cây, chứa các dự án, thư mục, tập tin, tài nguyên (resource) và các dữ liệu khác
  - Thêm 1 dự án vào Solution: R-click/ Add/ New Project…
  - Solution Property: R-click / Properties (Alt + Enter)

Visual Studio .NET

- Solution (Giải pháp)

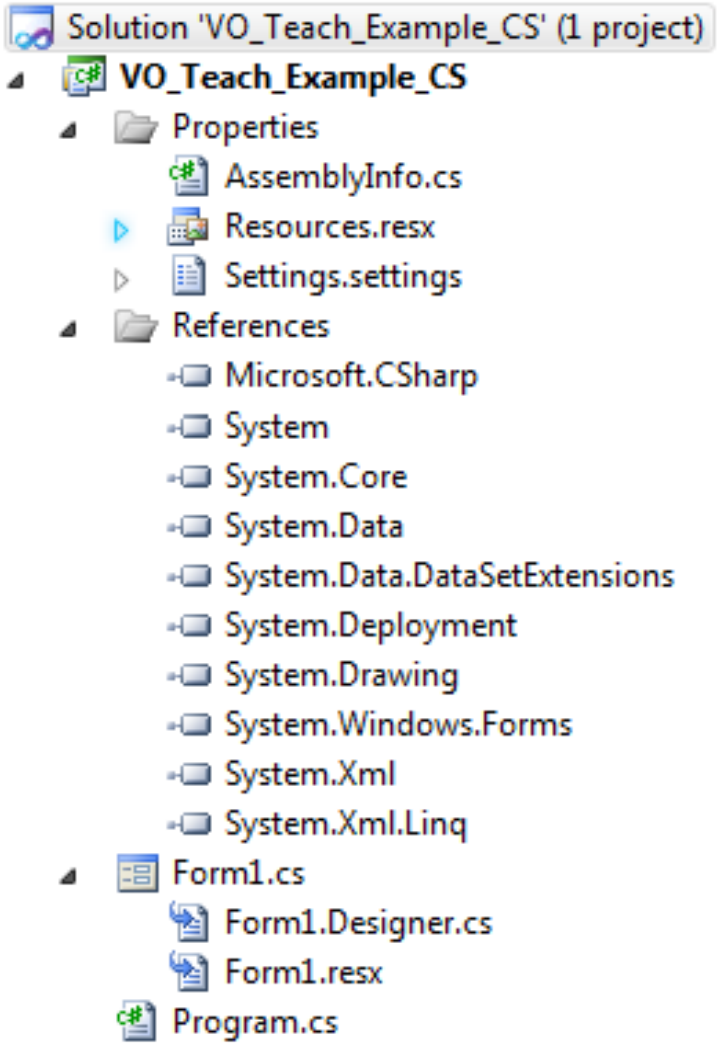


Visual Studio .NET

- Dự án (Project)
  - Solution chứa các dự án, còn các dự án chứa những mục thông tin như:
    - Các thư mục dùng cho việc tổ chức mã nguồn
    - Những tập tin định nghĩa (header file)
    - Những tập tin tài nguyên (resource file)
    - Những tham chiếu đến các dự án khác
    - Các dịch vụ web (web service)
    - Những tập tin hệ thống (system file)
    - Những tập tin mã nguồn của dự án…

Visual Studio .NET

- Dự án (Project)



Lập trình với WINFORM

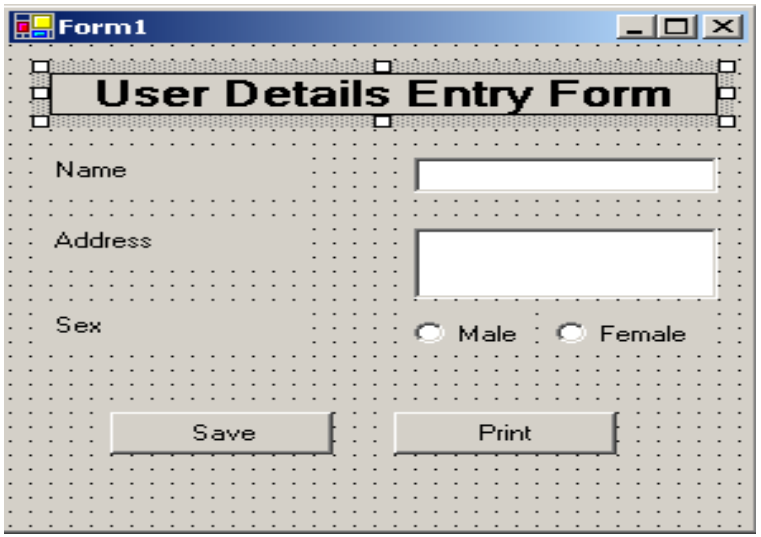
- Sử dụng **System.Windows.Forms**
- Dựa trên nền tảng lập trình HĐT
- Dùng chung giao diện với các ứng dụng khác
- Giao diện winform được kế thừa từ lớp Form gồm nhiều đối tượng Visual Interface Component (gọi là các Controls)

Lập trình với WINFORM

Application

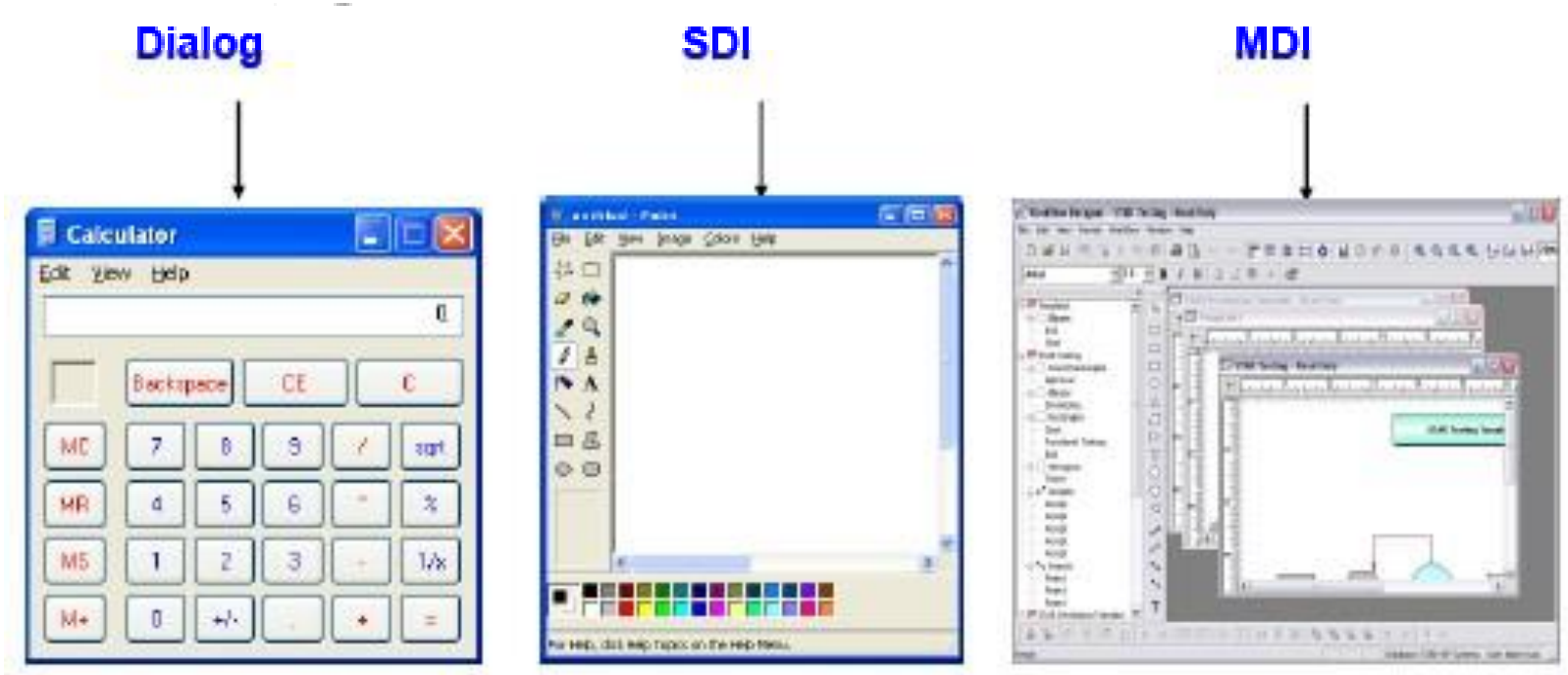
User Interface

Program



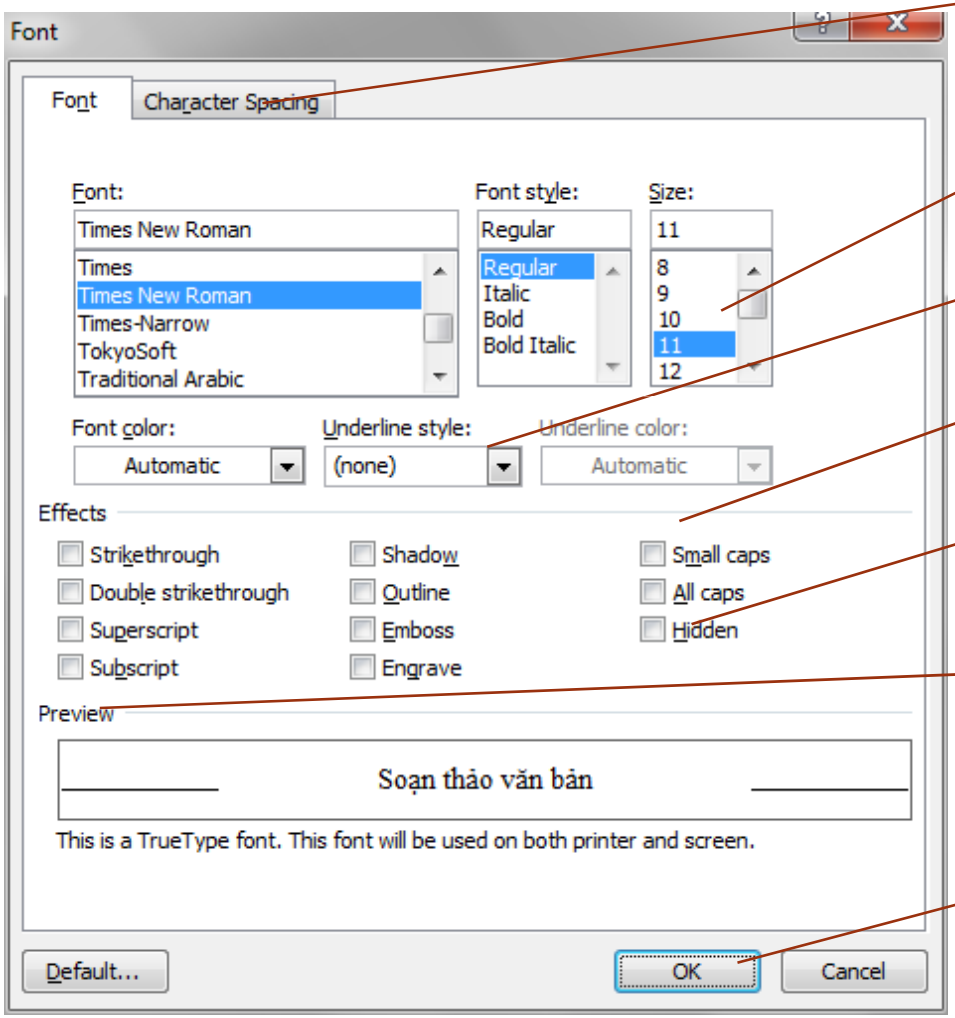
Giao diện đồ họa GUI

- Graphics User Interface – GUI
  - Là 1 vùng chữ nhật trên màn hình
  - Dùng để hiển thị kết quả output và nhận các input từ người dùng
- Các dạng GUI cơ bản



Giao diện đồ họa GUI

- Các dạng Control chuẩn của Windows



Tab Page

List box

Combo box

Group box

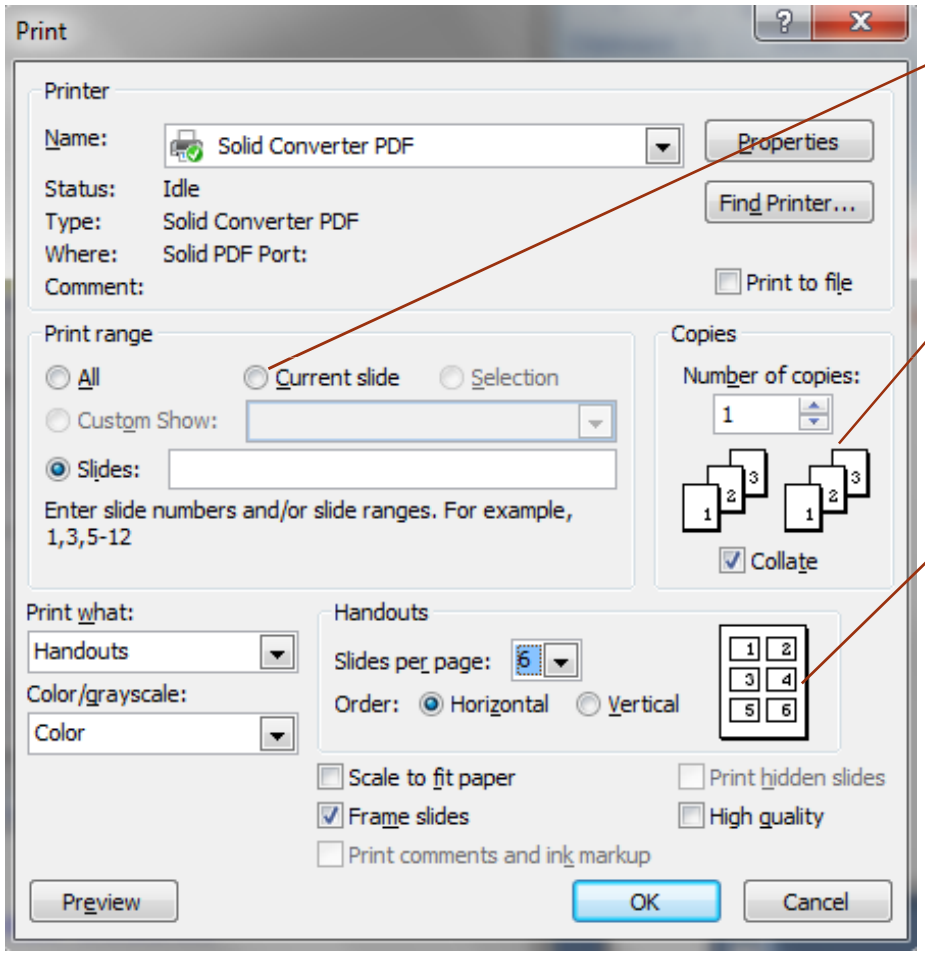
Check box

Label

Button

Giao diện đồ họa GUI

- Các dạng Control chuẩn của Windows



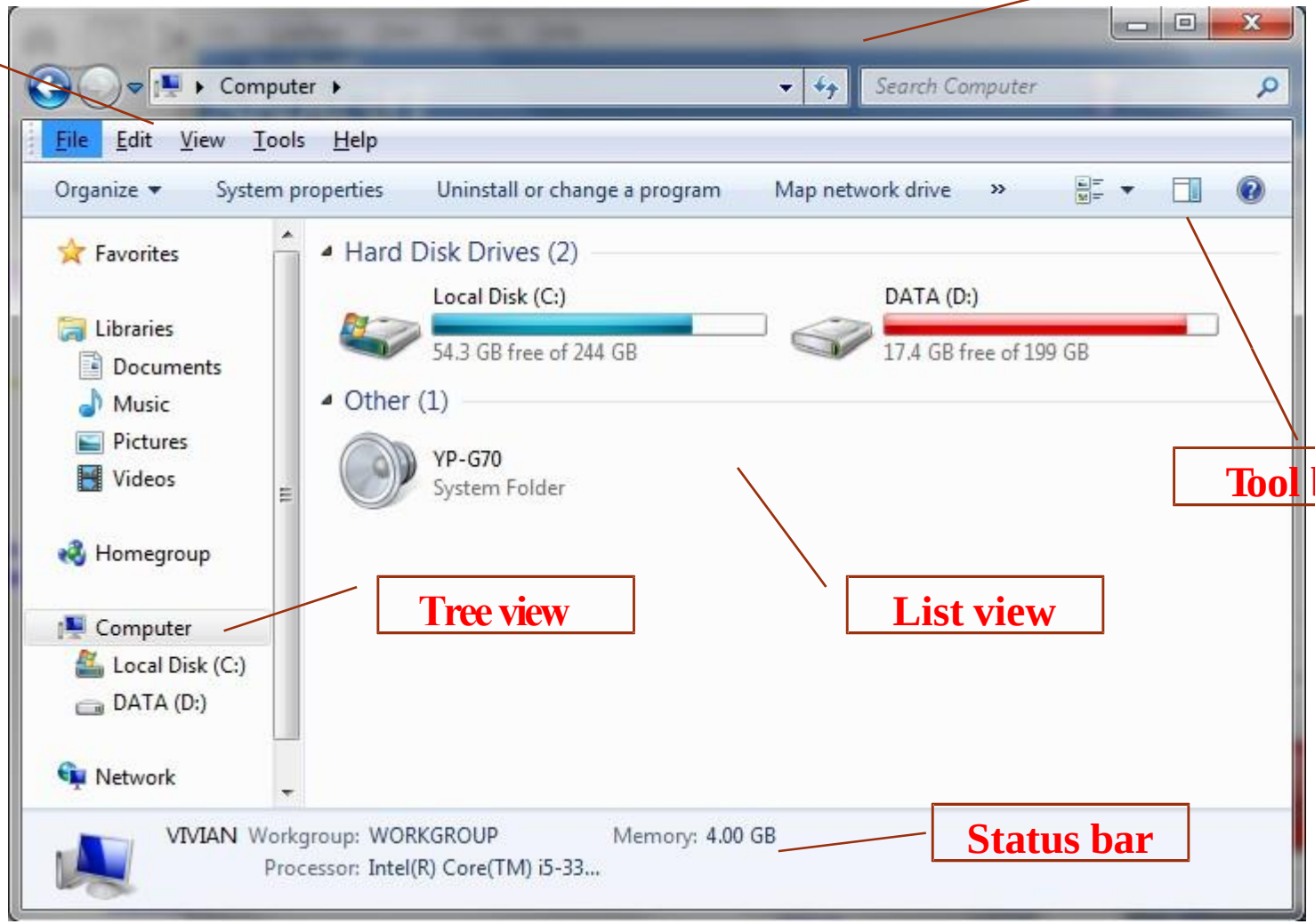
Radio button

NumericUpDown

Picture box

Giao diện đồ họa GUI

- Các dạng Control chuẩn của Windows



Lập trình với WINFORM

- Event Handling: Sự tương tác giữa các đối tượng với ứng dụng

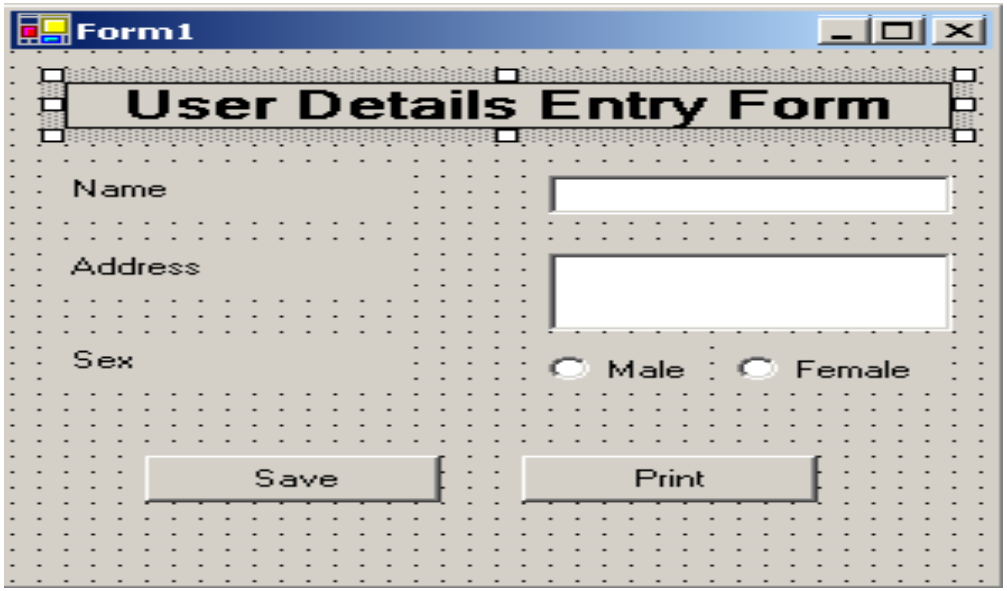
Lập trình với WINFORM

- Event Handling

```
When Save Clicked()
{
 Display Message("Save Button");
}
```

Event

Event Handler



Lập trình với WINFORM

- Một số sự kiện quan trọng với các Controls

SỰ KIỆN	MÔ TẢ TÁC DỤNG
Click	Xảy ra khi chuột trái được ấn. Tùy thuộc vào điều khiển, có thể cũng xảy ra khi ấn 1 phím ví dụ như Enter
DoubleClick	Xảy ra khi chuột trái được click đúp. Không phải tất cả điều khiển đều phản ứng lại sự kiện này
KeyDown	Xảy ra khi 1 phím được ấn lần đầu
KeyPress	Xảy ra khi điều khiển có focus và người dùng ấn và thả 1 phím
KeyUp	Xảy ra khi 1 phím được thả

Lập trình với WINFORM

- Một số sự kiện quan trọng với các Controls

SỰ KIỆN	MÔ TẢ TÁC DỤNG
MouseClick	Xảy ra khi điều khiển được ấn bằng chuột
MouseDoubleClick	Xảy ra khi ấn đúp chuột lên điều khiển
MouseDown	Xảy ra khi trỏ chuột di qua và 1 phím chuột được ấn.
MouseEnter	Xảy ra khi chuột ấn vào điều khiển
MouseHover	Xảy ra khi chuột dừng lại tại điều khiển trong 1 khoảng thời gian

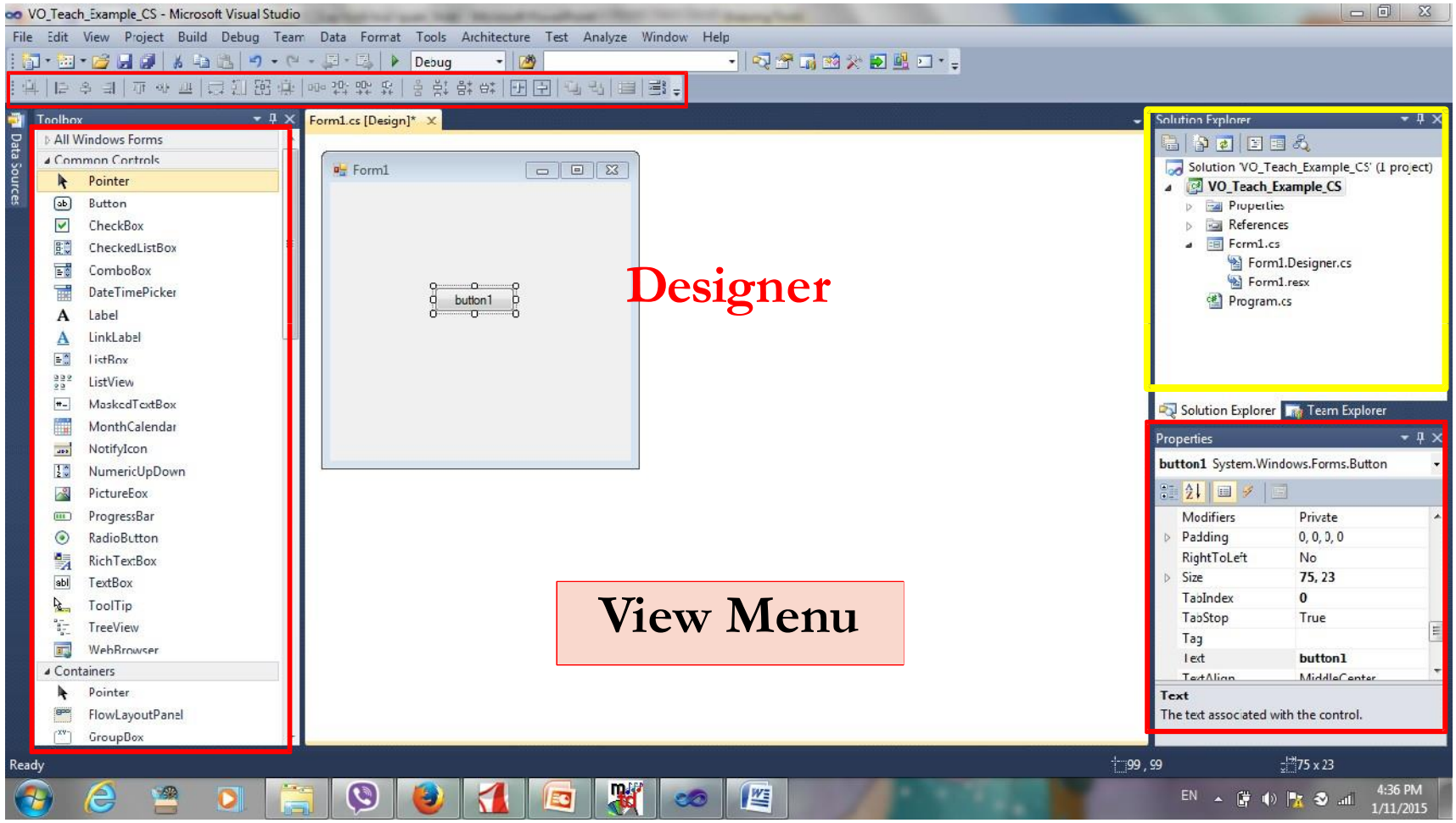
Lập trình với WINFORM

- Một số sự kiện quan trọng với các Controls

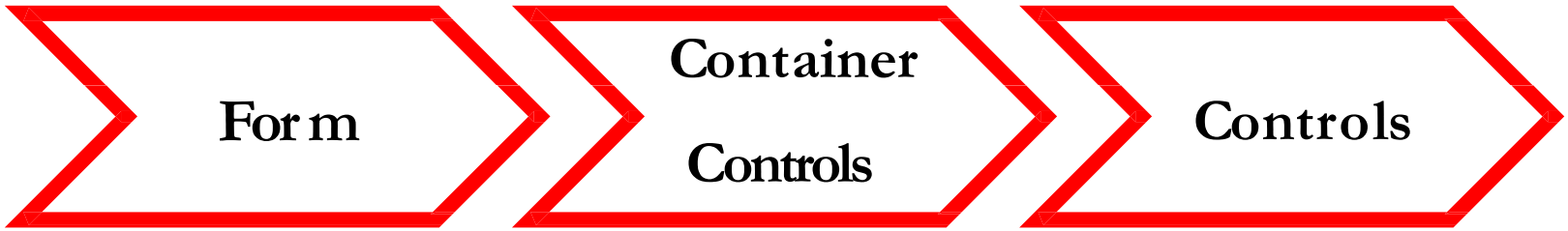
SỰ KIỆN	MÔ TẢ TÁC DỤNG
MouseLeave	Xảy ra khi chuột được di khỏi điều khiển
MouseMove	Xảy ra khi trỏ chuột di chuyển qua đối tượng
MouseUp	Xảy ra khi trỏ chuột ở trên đối tượng và nút chuột được thả ra
Validated	Xảy ra sau khi điều khiển xác nhận tính hợp lệ thành công
Validating	Xảy ra khi điều khiển đang xác nhận tính hợp lệ
…	

Lập trình với WINFORM

- Giao diện (IDE) của Visual Studio (ver 2010)



Lập trình với WINFORM



1. Thêm và cấu hình WINFORM

- NỘI DUNG CHÍNH
  - Thêm 1 windows form vào project khi design
  - Thêm 1 windows form khi run-time
  - Thay đổi kích thước window khi design và run-time
  - Các thuộc tính của form

1. Thêm và cấu hình WINFORM

- Một số thuộc tính cơ bản của winform

THUỘC TÍNH	MÔ TẢ TÁC DỤNG
(Name)	Thiết lập tên của Form, chỉ đặt được trong chế độ *design time*.
Backcolor	Chỉ định màu nền của form
BackgroundImage	Chỉ định ảnh nền của form
BackgroundImage-Layout	Xác định cách thức hình ảnh được chỉ định làm ảnh nền của form qua thuộc tính *BackgroundImage* được bố trí trên form. (Nếu không chọn ảnh nền thì thuộc tính này không tác động lên form)

1. Thêm và cấu hình WINFORM

- Một số thuộc tính cơ bản của winform

THUỘC TÍNH	MÔ TẢ TÁC DỤNG
Cursor	Xác định cách thức con trỏ xuất hiện khi di chuyển qua form
Enabled	Xác định khi nào form có thể nhận đầu vào. Nếu nhận giá trị *False*, mọi control trong form cũng bị mất khả năng (disabled)
Font	Thiết lập font mặc định cho form. Mọi control trong form cũng sẽ kế thừa font này trừ khi thuộc tính *Font* của chúng được thiết lập một cách tách biệt.
FormBorderStyle	Xác định khung viền cho form và thanh tiêu đề (title bar)

1. Thêm và cấu hình WINFORM

- Một số thuộc tính cơ bản của winform

THUỘC TÍNH	MÔ TẢ TÁC DỤNG
ForeColor	Xác định màu chữ được hiển thị. Tất cả các control trong form sẽ kế thừa mày này trừ khi chúng được thiết lập riêng.
HelpButton	Chọn hiển thị hay không hiển thị nút Help trên form
Icon	Xác định biểu tượng sử dụng để tượng trưng cho form
Location	Khi thuộc tính *StartPosition* được thiết lập là *Manual*, thuộc tính này sẽ xác định vị trí bắt đầu của form so với góc trên trái (upper left-hand) của màn hình.

1. Thêm và cấu hình WINFORM

- Một số thuộc tính cơ bản của winform

THUỘC TÍNH	MÔ TẢ TÁC DỤNG
MaximizeBox	Chỉ định form có hay không nút phóng to form (Maximize)
MaximizeSize	Xác định kích thước phóng to của form. Nếu thuộc tính này được thiết lập là *(0,0)* thì form không giới hạn kích thước trên.
MinimizeBox	Chỉ định form có hay không nút thu nhỏform (Minimize)
MinimizeSize	Xác định kích thước thu nhỏ của form, cho phép người dùng có thể đưa form trở lại kích thước trước.

1. Thêm và cấu hình WINFORM

- Một số thuộc tính cơ bản của winform

THUỘC TÍNH	MÔ TẢ TÁC DỤNG
Opacity	Xác định độ trong suốt (chắn sáng) của form từ 0% (hoàn toàn trong suốt) đến 100%. (hoàn toàn mờ đục)
Size	Lấy (get) và thiết lập (set) kích thước khởi tạo (initial) của form
StartPosition	Xác định vị trí form xuất hiện lần đầu tiên
Text	Xác định đầu đề của form

1. Thêm và cấu hình WINFORM

- Một số thuộc tính cơ bản của winform

THUỘC TÍNH	MÔ TẢ TÁC DỤNG
Visible	Chỉ định form khả dụng (thấy được) khi đang chạy hay không
WindowState	Xác định lần đầu tiên form sẽ được thu nhỏ, phóng to hay theo 1 kích thước cụ thể đã được thiết lập bởi thuộc tính *Size*.
TopMost	Nhận giá trị *True* sẽ làm cho form hiện tại luôn luôn xuất hiện bên trên tất cả các form khác .

1. Thêm và cấu hình WINFORM

- Một số chú ý khi viết code
  - Thuộc tính *Name* biểu thị tên của lớp Form, là một ngoại lệ và được sử dụng bên trong namespace để xác định duy nhất lớp mà Form là 1 thể hiện.
  - Thiết lập Title của form
    - **Form1.Text = “Please enter your title!”;**
  - Thiết lập Border Style của form
    - Thuộc tính FormBorderStyle có 7 giá trị

1. Thêm và cấu hình WINFORM

- Một số chú ý khi viết code
  - Thiết lập Border Style của form

GIÁ TRỊ	MÔ TẢ TÁC DỤNG
None	Form không có đường viền và không phóng to, thu nhỏ, không Help hay không có các nút điều khiển.
FixedSingle	Form có đường viền đơn và người dùng không thể resize. Các tính chất minimize, maximize, help, control box được xác định bởi các thuộc tính khác.
Fixed3D	Form có đường viền dạng 3D và không thể resize bởi người dùng. Có thể phóng to, thu nhỏ, help, control box.

1. Thêm và cấu hình WINFORM

- Một số chú ý khi viết code
  - Thiết lập Border Style của form

GIÁ TRỊ	MÔ TẢ TÁC DỤNG
FixedDialog	Form có đường viền đơn và không thể resize bởi người dùng. Có các thuộc tính minimize, maximize, help; nhưng không có control box.
Sizable	Là thiết lập mặc định của form, làm cho form có thể được resize bởi người dùng và có các thuộc tính minimize, maximize, help.
FixedToolWindow	Form có đường viền đơn và không thể resize bởi người dùng. Cửa sổ form không chứa control box nào trừ nút Close.
SizableToolWindow	Form có đường viền đơn và có thể resize bởi người dùng. Cửa sổ form không chứa control box nào trừ nút Close.

1. Thêm và cấu hình WINFORM

- Một số chú ý khi viết code
  - Thiết lập Border Style của form
    - Thuộc tính *FormBorderStyle* có thể thiết lập cả khi design lẫn runtime
    - **aForm.FormBorderStyle = FormBorderStyle.Fix3D;**
  - Thiết lập Window State của form
    - Thuộc tính *WindowState* xác định trạng thái form xuất hiện khi lần đầu mở.
    - Thuộc tính *WindowState* có 3 giá trị: *Normal* (mặc định – form sẽ xuất hiện theo kích thước được thiết lập tại thuộc tính *Size*), *Minimized* (form sẽ thu nhỏ tại taskbar), *Maximized* (form được phóng to).
    - Thuộc tính *WindowState* có thể được thiết lập khi run-time nhưng sẽ không ảnh hưởng đến trạng thái của form.

1. Thêm và cấu hình WINFORM

- Một số chú ý khi viết code
  - Thiết lập vị trí khi bắt đầu (startup location) của form
    - .Được thiết lập nhờ 2 thuộc tính StartPosition và Location
    - Các giá trị của *StartPosition* có thể được thiết lập các giá trị bất kỳ trong bảng liệt kê *FormStartPosition* như sau:

GIÁ TRỊ	MÔ TẢ TÁC DỤNG
Manual	Thiết lập theo thuộc tính Location
CenterScreen	Chính giữa màn hình
WindowsDefault-Location	Form sẽ được đặt tại vị trí mặc định của Windows và có kích thước theo thuộc tính Size
WindowsDefault-Bounds	Form sẽ được đặt tại vị trí mặc định của Windows và có kích thước theo kích thước mặc định của Windows
CenterParent	Chính giữa của form cha.

Nguyễn Hữu Tuân - tuannh@utt.edu.vn

1. Thêm và cấu hình WINFORM

- Một số chú ý khi viết code
  - Thiết lập vị trí khi bắt đầu (startup location) của form
    - .Khi nhận giá trị *Manual*, thì vị trí của form sẽ theo thuộc tính Location.
    - Ví dụ để bắt đầu form ở góc trên trái của màn hình thì thuộc tính Location đặt là (0,0).
    - Ví dụ form sẽ bắt đầu ở vị trí 400 pixels so với bên phải và 200 pixels bên dưới góc trên trái của màn hình thì thuộc tính Location đặt là (400, 200)
  - Giữ form ở Top của UI (user interface)
    - Thiết lập thuộc tính *TopMost* thành *True*.
    - Khi form có thuộc tính TopMost là True sẽ ở trên bất kỳ form nào có thuộc tính TopMost là False (mặc định). Khi có nhiều form có thuộc tính TopMost là True thì chúng có thể che phủ lẫn nhau.

1. Thêm và cấu hình WINFORM

- Một số chú ý khi viết code
  - Thiết lập độ trong suốt (Transparency) và độ mờ (Opacity) của form
    - .Thông qua thuộc tính Opacity, với giá trị từ 0% (hoàn toàn trong suốt) đến 100% (hoàn toàn mờ đục - solid)
    - Thiết lập nhờ Property Grid hoặc code

```
//C#
        aForm.Opacity = 0.5;
```

    - Thuộc tính Opacity hữu ích khi muốn giữ 1 form ở phía trước nhưng lại có thể quan sát được nền phía sau của form. Hầu như các control đều kế thừa opacity của form chứa nó.

1. Thêm và cấu hình WINFORM

- Một số chú ý khi viết code
  - Thiết lập form khởi động (Startup Form)
    - .Khi ứng dụng chứa nhiều form, cần thiết lập 1 form là form khởi động, form này sẽ được load khi thực thi ứng dụng
    - Form sẽ là form khởi động cần được chỉ định trong hàm *Main*. Mặc định hàm này sẽ được lưu trữ tại 1 lớp có tên *Program.cs* được tự động tạo ra bởi VS.
    - Tại Solution Explorer, click đúp chuột vào *Program.cs* và sửa tên *myForm* thành tên form muốn khởi động đầu tiên.

```
//C#
static  void Main()
{
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application..Run(new  myForm);
```

1. Thêm và cấu hình WINFORM

- Một số chú ý khi viết code
  - Ẩn form khởi động (making the startup form invisible)
    - **aForm.Visible = false;**

2. Container Controls

- NỘI DUNG CHÍNH
  - Thêm 1 control vào form hoặc container-control khi design
  - Thêm 1 control vào form hoặc container-control khi run-time
  - Nhóm và sắp các control với điều khiển *Panel*
  - Nhóm và sắp các control với điều khiển *GroupBox*
  - Nhóm và sắp các control với điều khiển *TabControl*
  - Nhóm và sắp các control với điều khiển *FlowLayoutPanel*
  - Nhóm và sắp các control với điều khiển *TableLayoutPanel*
  - Nhóm và sắp các control với điều khiển *SplitContainer*

2. Container Controls

- Thêm 1 control vào form hoặc container-control khi design
  - Cách 1: Kéo control từ Toolbox
  - Cách 2: Chọn control trong Toolbox và sau đó dùng chuột vẽ vào form
  - Cách 3: Chọn control trong Toolbox và click đúp chuột trên form
  - Cách 4: click đúp chuột vào control trên Toolbox.

2. Container Controls

- Thêm 1 control vào form hoặc container-control khi run-time
  - Thêm control vào form
  - Thiết lập các thuộc tính trước khi thêm vào
- Ví dụ thêm một Button vào một Panel hoăc Form

```
//C#
        Button  aButton = new Button();
//Set the relative location in the containing control or form
        aButton.Location = new Point(20,20);
        aButton.Text = "Test Button"
//Adds the button to a panel called Panel1
        Panel1.Controls.Add (aButton);
//Adds the button to a form called Form1
        this.Controls.Add (aButton);
```

2. Container Controls

- Thuộc tính *Anchor* và *Dock* dùng để bố trí điều khiển trong form hay điều khiển chứa nó.
- Thuộc tính ***Anchor***:
  - Cho phép xác định khoảng cách giữa 1 hay nhiều cạnh của điều khiển với 1 hay nhiều cạnh của form hay các điều khiển chứa khác
  - Nếu người dùng thay đổi kích thước form lúc chạy, các cạnh của điều khiển sẽ luôn luôn duy trì khoảng cách riêng với các cạnh
  - Thiết lập mặc định của thuộc tính Anchor là *Top*, *Left*.

2. Container Controls

- Thuộc tính ***Dock***:
  - Cho phép gắn điều khiển vào cạnh của điều khiển cha (form, container controls: tab, panel…)
  - Để gắn điều khiển vào cạnh của form, click vào thanh phải của giao diện thuộc tính Dock.
  - Để hủy chọn None.
  - Click vào chính giữa của giao diện thuộc tính Dock sẽ thiết lập Fill, nghĩa là điều khiển sẽ gắn vào cả 4 cạnh của form và phủ kín form.

2. Container Controls

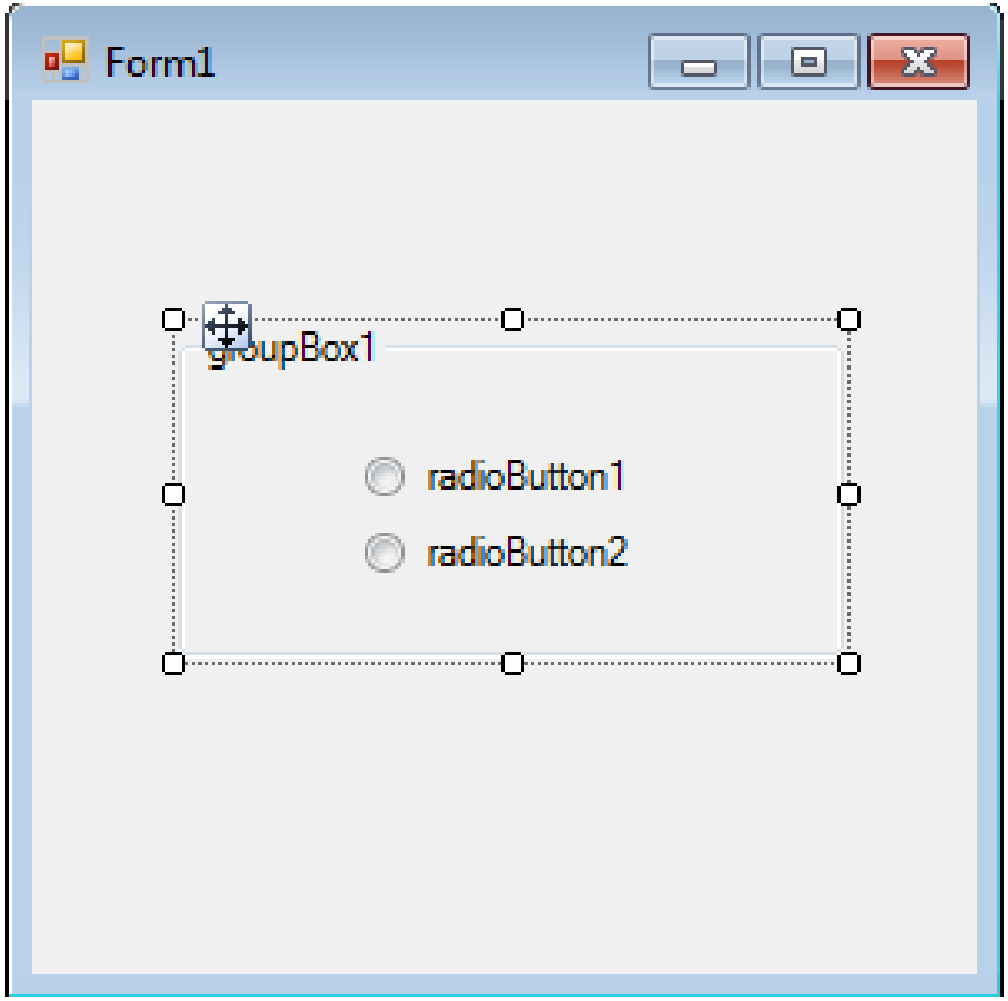
- Điều khiển **GroupBox**:

grb

  - (The GroupBox control is a container control that appears as a subdivision of the form surrounded by a border)
  - Không có thanh cuộn (scrollbars) giống như Panel
  - Không hỗ trợ bất trì một bố cục sẵn có nào (layout)
  - Thuộc tính Text của GroupBox cho phép hiện thị đầu đề (caption)
  - Hay sử dụng để gom nhóm các RadioButton. Các RadioButton được đặt trong cùng 1 GroupBox sẽ loại trừ lẫn nhau nhưng sẽ không loại trừ những RadioButton khác trong form hay trong các GroupBox khác.

2. Container Controls

- Điều khiển **GroupBox**:



2. Container Controls

- Điều khiển **Panel**:

pn

  - Điều khiển Panel sẽ tạo ra các vùng con (subsection) của form để chứa những điều khiển khác
  - Panel có thể không cần phân biệt với phần còn lại của form, hoặc có thể được bao bởi 1 khung nhờ thuộc tính *BorderStyle*.
  - Panel là điều khiển có thanh cuộn.

THUỘC TÍNH	MÔ TẢ TÁC DỤNG
AutoScroll	Nhận giá trị True: thanh cuộn sẽ xuất hiện khi nội dung điều khiển vượt ra khỏi đường viền vật lý của Panel. Ngược lại sẽ không hiển thị.
BorderStyle	None: không có khung bao quanh FixedSingle: đường viền đơn Fixed3D: đường viền 3D

2. Container Controls

- Điều khiển **FlowLayoutPanel**:

flp

  - Là lớp con của điều khiển Panel
  - Thường được dùng để tạo các vùng riêng biệt (distinct subsection) trong form nhằm chứa các điều khiển có quan hệ với nhau
  - Khác Panel, khi bị thay đổi kích thước thì điều khiển FlowLayoutPanel sẽ tự động thay đổi vị trí các điều khiển mà nó chứa cả ở 2 chế độ design và runtime.
  - Ưu điểm: FlowLayoutPanel trợ giúp nhiều trong thiết kế giao diện vì vị trí các điều khiển có thể tự động tùy chỉnh khi kích thước hay chiều của FlowLayoutPanel bị thay đổi.
  - Giống Panel, có thanh cuộn (AutoScroll)

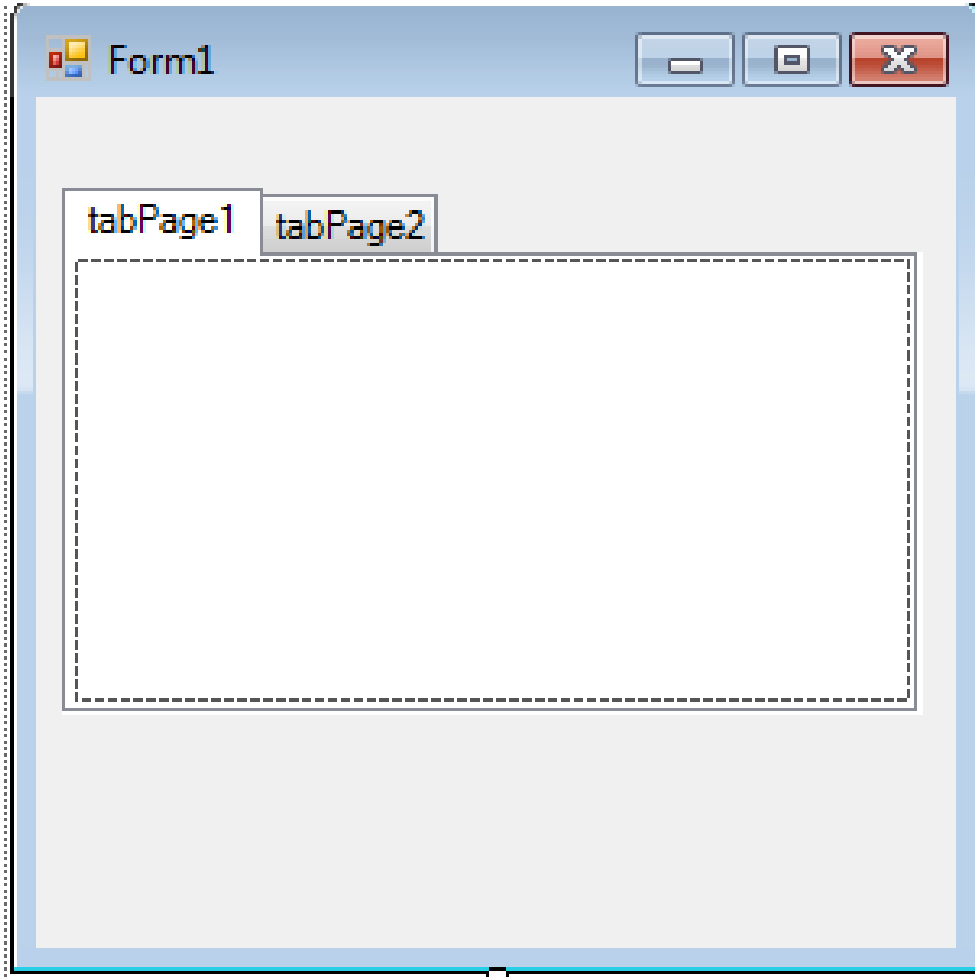
2. Container Controls

tab

- Điều khiển **TabControl**:
  - Cho phép gom nhóm các điều khiển vào các thẻ (TabPage). TabPage là một điều khiển chứa đặc biệt chỉ thuộc về TabControl
  - Thuộc tính quan trọng của TabControl là thuộc tính TabPages, mỗi TabPage lại có tập các thuộc tính riêng biệt
  - Các TabPage khá giống các điều khiển Panel.

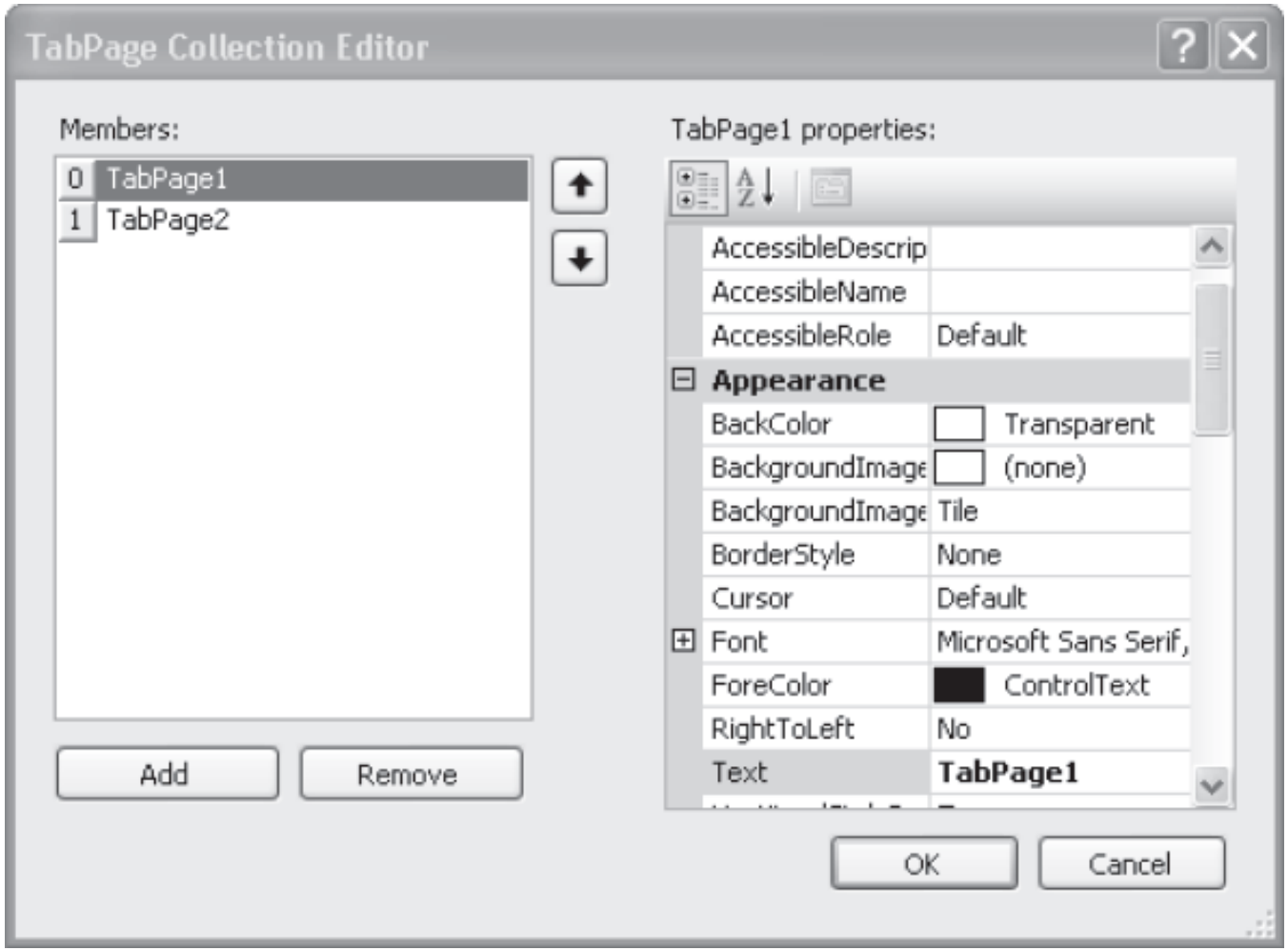
2. Container Controls

- Điều khiển **TabControl**:



2. Container Controls

- Điều khiển **TabControl**:



2. Container Controls

- Điều khiển **TabControl**:
  - Các thuộc tính cơ bản của **TabControl**.

Property	Description
Appearance	Xác định kiểu dáng của TabControl gồm các giá trị Normal, Buttons, FlatButtons
Alignment	Căn chỉnh tab về Top, Bottom, Left hoặc Right so với TabControl
Multiline	= True: hỗ trợ tab nhiều dòng (multiple rows of tabs) = False: chỉ 1 dòng tab
TabPages	Biểu diễn tập hợp các điều khiển TabPage.

2. Container Controls

- Điều khiển **TabControl**:
  - Các thuộc tính cơ bản của **TabPage**.

Property	Description
AutoScroll	Nhận giá trị True: thanh cuộn sẽ xuất hiện khi nội dung điều khiển vượt ra khỏi đường viền vật lý của Panel. Ngược lại sẽ không hiển thị.
BorderStyle	None: không có khung bao quanh FixedSingle: đường viền đơn Fixed3D: đường viền 3D
Text	Tiêu đề hiển thị của TabPage

3. Base Controls

- NỘI DUNG CHÍNH
  - Các thuộc tính chung của các điều khiển
  - Điều khiển *Button*
  - Điều khiển *Label*
  - Điều khiển *LinkLabel*
  - Điều khiển *TextBox*

3. Base Controls

- Tổng quan về Controls
  - Điều khiển (controls) là các thành phần để kết hợp giao diện đồ họa với các hàm được định nghĩa sẵn. Controls là các đơn vị code tái sử dụng đã được thiết kế để thực hiện các công việc cụ thể.
  - Tất cả các điều khiển đều thừa kế từ lớp cơ sở *Control* và có một số thuộc tính chung như sau:

3. Base Controls

- Common Properties of Controls

Property	Description
Anchor	Xác định cách thức điều khiển được gắn vào form cha hay điều khiển chứa.
BackColor	Thiết lập hoặc lấy ra màu nền của điều khiển
BackgroundImage	Chọn hình nền cho điều khiển
CauseValidation	
ContainsFocus	Chỉ định điều khiển hiện tại hoặc một điều khiển con của nó nhận con trỏ (focus)
Controls	Lấy ra tập các điều khiển được chứa bên trong điều khiển hiện tại. Chỉ sử dụng cho Container
Cursor	Biểu diễn con trỏ khi mà chuột di qua điều khiển
Dock	Cách thức điều khiển "dính" vào form cha hay

3. Base Controls

- Common Properties of Controls

Property	Description
Enabled	Thiết lập hoặc kiểm tra nếu một điều khiển khả dụng. Nếu điều khiển không khả dụng thì nó sẽ có màu xám và không thể chọn và sửa được
Font	Thiết lập hoặc lấy ra font chữ hiển thị văn bản
ForeColor	Màu chữ hoặc màu cho foreground (cận cảnh)
HasChildren	Xác định xem điều khiển có điều khiển con hay không
Height	Biểu diễn độ cao của điều khiển (pixel)
Location	Xác định vị trí góc trên trái của điều khiển so với góc trên trái của form cha hoặc điều khiển chứa
MaximumSize	Thiết lập hoặc lấy ra kích thước tối đa

3. Base Controls

- Common Properties of Controls

Property	Description
Name	Tên của điều khiển dùng trong code. Thuộc tính này chỉ có thể thay đổi khi design và không thể thay đổi lúc chạy
Parent	Thiết lập hoặc lấy ra form cha hoặc điều khiển chứa điều khiển hiện tại. Thiets lập thuộc tính này sẽ thêm điều khiển vào tập các điều khiển của điều khiển cha
Region	Thiết lập hoặc lấy ra vùng cửa sổ kết hợp với điều khiển
Size	Kích thước của điều khiển (pixel)
TabOrder	Xác định thứ tự điều khiển sẽ được chọn khi ấn nút Tab để di chuyển giữa các điều khiển

3. Base Controls

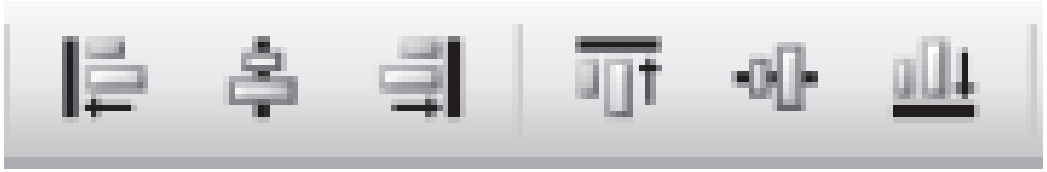
- Common Properties of Controls

Property	Description
Tag	Cho phép người lập trình lưu trữ 1 giá trị hoặc đối tượng liên quan đến điều khiển
Text	Thiết lập hoặc lấy ra xâu ký tự (text) liên quan đến điều khiển. Xâu ký tư có hoăc không hiển thi phụ thuộc vào loại điều khiển và việc thiết lập các thuộc tính
Visible	Thiết lập khả năng thấy được (hữu hình) của điều khiển
Width	Độ rộng của điều khiển (pixel)

3. Base Controls

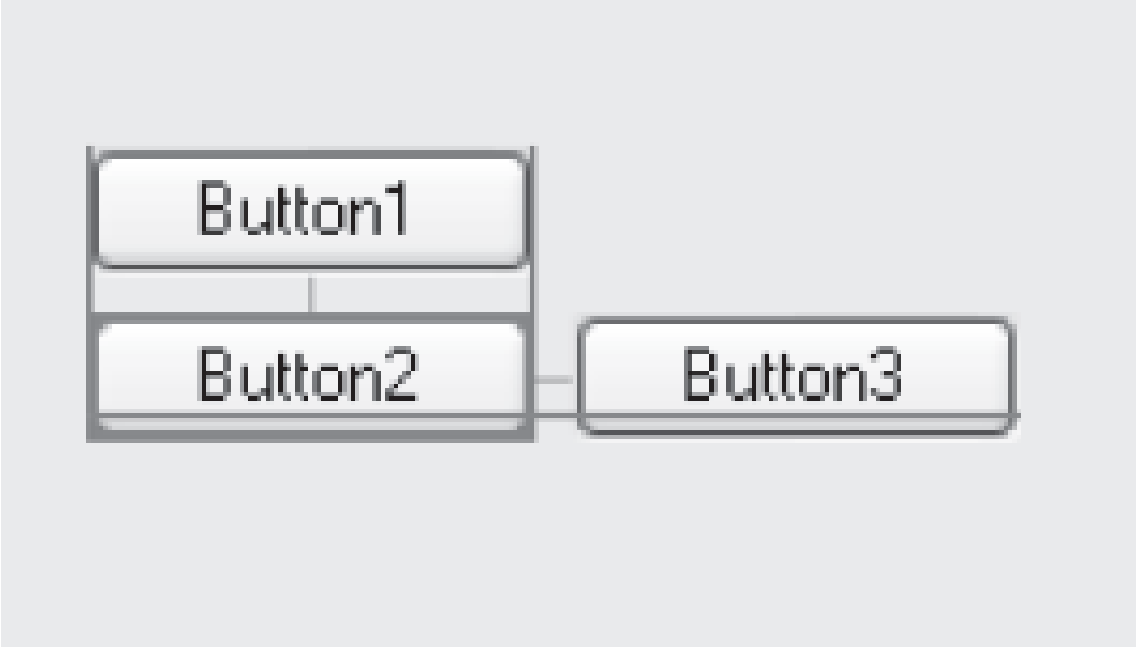
- Thanh công cụ **Layout Toolbar**:
  - Không phải là thanh công cụ mặc định
  - Để thêm vào IDE bằng cách vào thực đơn View → Toolbars → Layout
  - Chọn nhóm các điều khiển để thực hiện bố trí

Horrizontal and Vértical spacing buttons



3. Base Controls

- Tính năng **Snaplines**:
  - Xuất hiện khi 1 điều khiển được kéo vào form hoặc điều khiển chứa; khi một điều khiển được kéo gần cạnh của form, điều khiển chứa hoặc các điều khiển khác

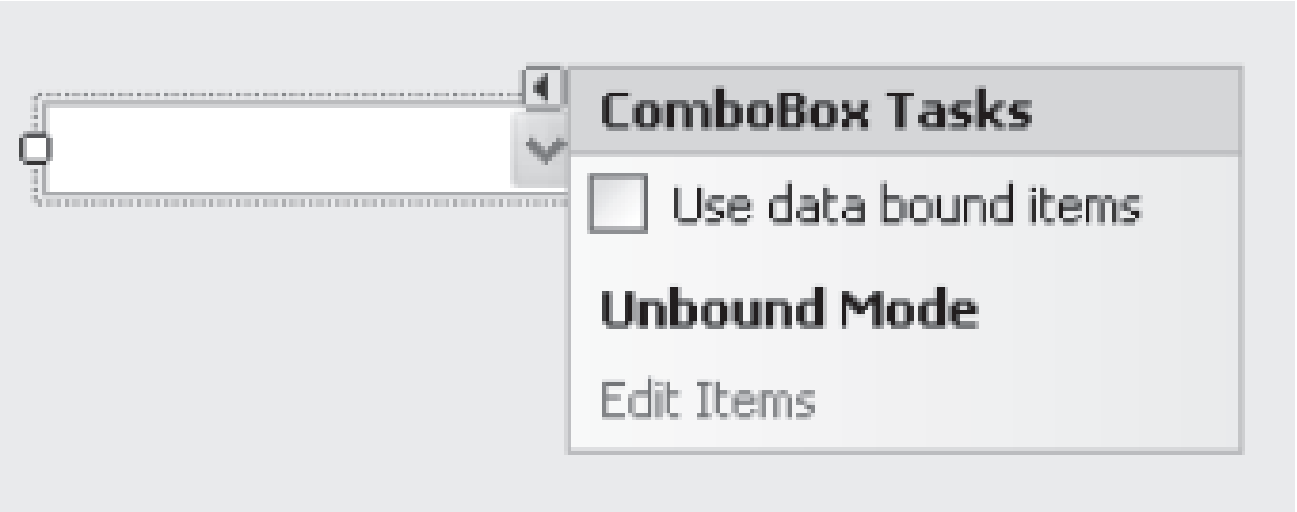


3. Base Controls

- Tính năng **Snaplines**:
  - Để kích hoạt khi snapline không xuất hiện:
    - Thực đơn Tool → Options…
    - Ở ô trái, chọn Windows Forms Designer và chọn General
    - Tại Property Grid, thiết lập LayoutMode là SnapLines
    - Ấn OK.

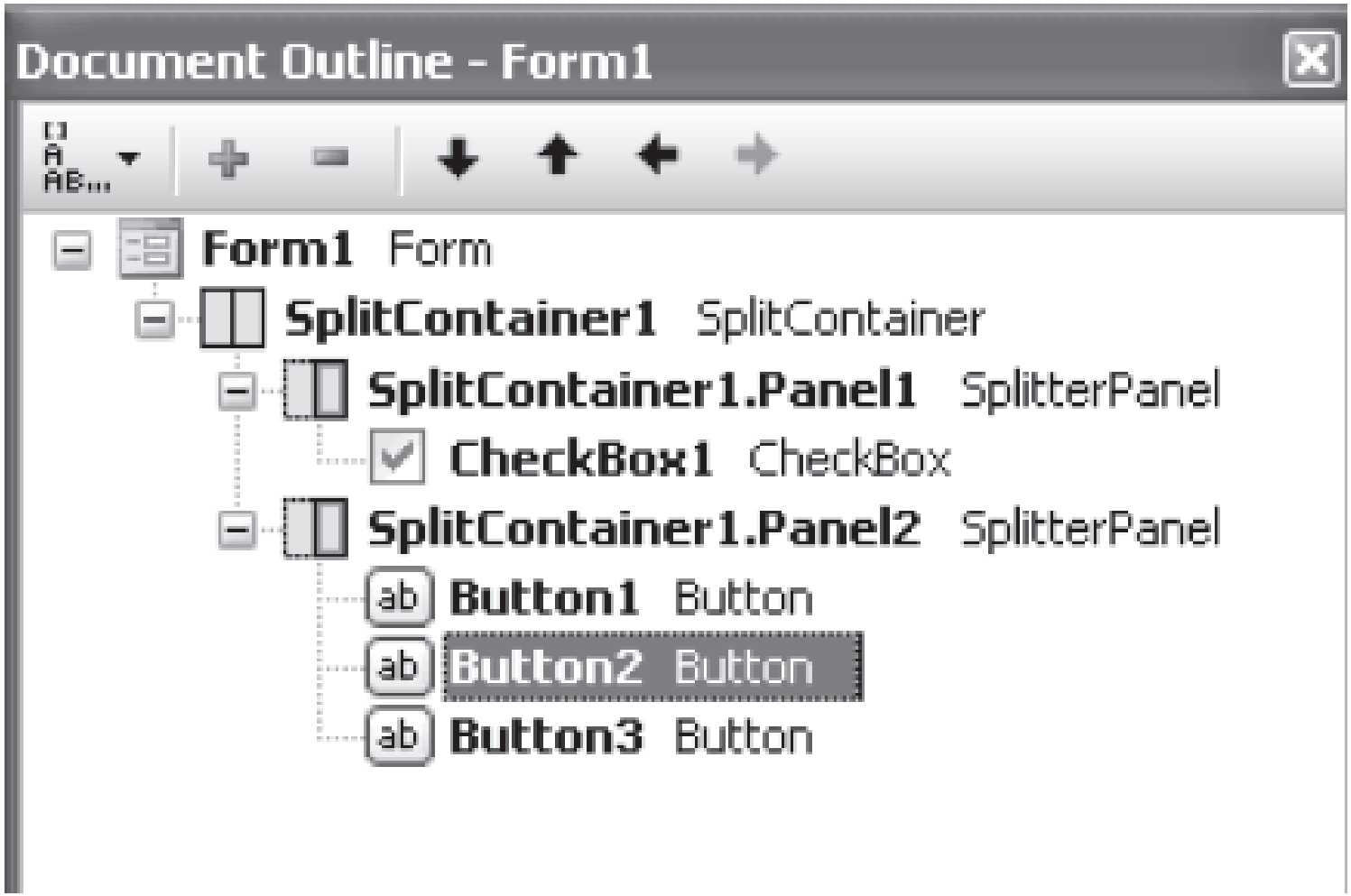
3. Base Controls

- Tính năng **Smart Tags**:
  - Một số điều khiển hiển thị những chức năng hay dùng nhất thông qua *smart tag*.



3. Base Controls

- Cửa sổ **Document Outline**:



3. Base Controls

btn

- Điều khiển **Button**:
  - Một trong những điều khiển cơ bản nhất, người dùng có thể click để thực hiện hành động
  - Các thuộc tính quan trọng của Button

Property	Description
AutoEllipsis	Cho phép tự động điều khiển văn bản khi nó vượt quá độ rộng của button
DialogResult	Thiết lập một giá trị DialogResult có thể kết hợp với button như DialogResult.OK hay DialogResult.Cancel
FlatStyle	Thiết lập hình dạng của button khi người dùng di chuột qua button và click

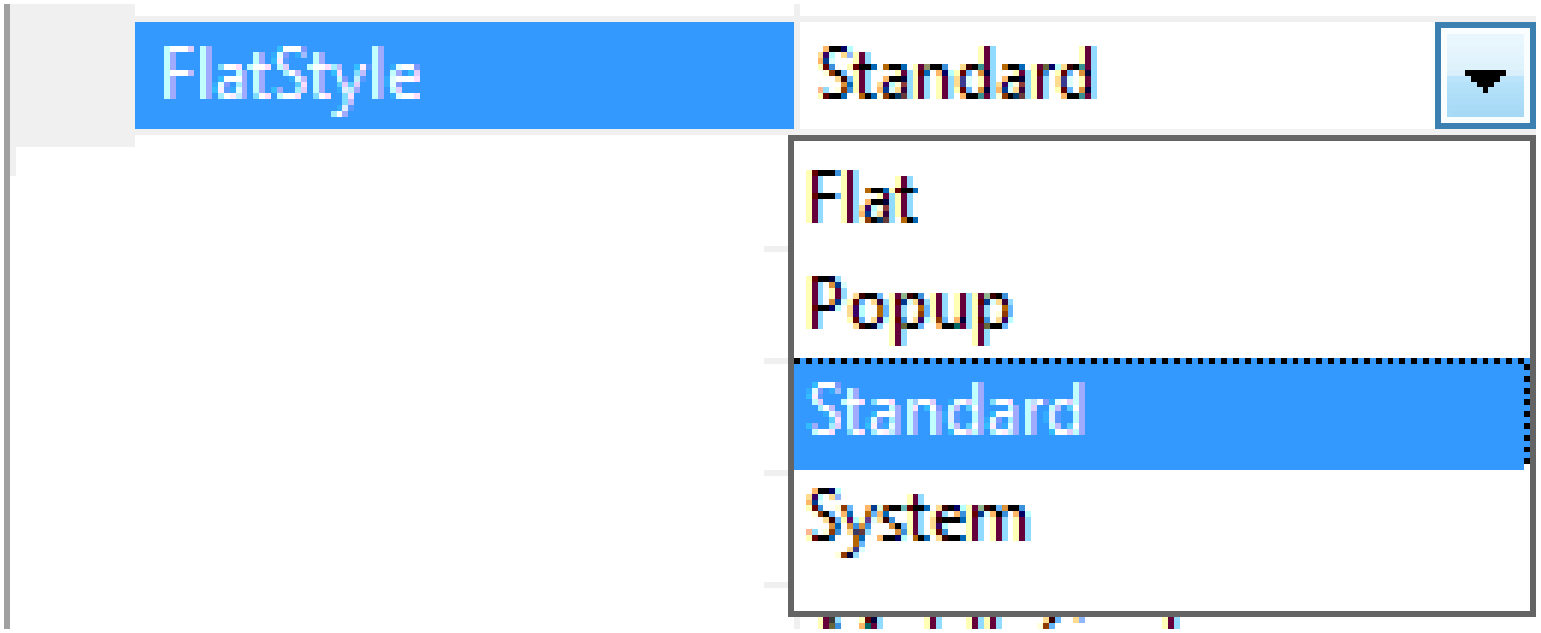
3. Base Controls

- Điều khiển **Button**:
  - Một trong những điều khiển cơ bản nhất, người dùng có thể click để thực hiện hành động
  - Các thuộc tính quan trọng của Button

Property	Description
FlatAppearance	Xác định cách thức button xuất hiện và khi thuộc tính FlatStyle thiết lập là Flat.
Text	Thiết lập xâu ký tự xuất hiện trên button
TexlAlign	Căn lề cho xâu ký tự xuất hiện trên button

3. Base Controls

- Điều khiển **Button**:
  - Thuộc tính FlatStyle và FlatAppearance
    - Thuộc tính FlatStyle



3. Base Controls

- Điều khiển **Button**:
  - Khi thuộc tính FlatStyle nhận giá trị Flat, thuộc tính FlatAppearance sẽ xác định các kiểu dáng của button.
  - FlatAppearance gồm các thuộc tính con sau:

Property	Description
BorderColor	Màu của đường viền button
BorderSize	Kích thước đường viền
MouseDownBack Color	Màu của button khi ấn chuột trái
MouseOverBack Color	Màu của button khi chuột di qua nút

3. Base Controls

- Điều khiển **Button**:
  - Tạo Accept Button và Cancel Button sử dụng thuộc tính DialogResult
    - Thêm 1 button vào form và thiết lập thuộc tính Text như mong muốn (Đồng ý – Accept button, hoặc Hủy – Cancel button)
    - Tại cửa sổ Properties, thiết lập thuộc tính DialogResult thành OK với Accept button hoặc Cancel với Cancel button
    - Tại Designer, click đúp chuột vào button để mở cửa sổ code
    - Tại sự kiện Button_Click , nếu muốn đóng form hiện tại thì thêm đoạn lệnh **this.Close();**

3. Base Controls

- Điều khiển **Button**:
  - Tạo Accept Button và Cancel Button sử dụng thuộc tính DialogResult
    - Khi form được hiển thị bằng phương thức ***ShowDialog***, nó sẽ tự động trả về kết quả dựa theo button được click. Giả sử form có tên là DialogForm, ví dụ:

```
Dialog aForm = new DialogForm();
System.Windows.Forms.DialogResult aResult;
aResult = aForm.ShowDialog();
if (aResult == DialogResult.OK)
        { //Do something      }
else
        {//Do something       }
```

3. Base Controls

lbl

- Điều khiển **Label**:
  - Dùng để hiển thị dữ liệu (read-only)
  - Chuỗi văn bản hiển thị trên Label được thiết lập tại thuộc tính *Text*.
  - Label có thể tự động điều chỉnh kích thước phù hợp với văn bản bằng cách thiết lập thuộc tính *AutoSize* là *True*. Nếu *AutoSize* là *False* thì có thể thiết lập kích thước của Label khi thiết kế bằng cách kéo các cạnh.
  - Label có thể được sử dụng để định nghĩa "phím tắt" cho các điều khiển khác (access keys – nghĩa là khi ấn cùng với phím Alt thì con trỏ sẽ chuyển đến điều khiển mòng muốn).

3. Base Controls

- Điều khiển **Label**:
  - Cách định nghĩa 1 phím tắt:
    - Từ Toolbox, kéo 1 điều khiển Label vào form gần với điều khiển muốn định nghĩa phím tắt (ví dụ, 1 Textbox)
    - Tại cửa sổ Properties, thiết lập thuộc tính Text để mô tả điều khiển. Điền ký tự & vào trước ký tự muốn dùng làm phím tắt. Ví dụ: *$Họ tên*, khi đó ký tự H sẽ là ký tự tắt và Alt+ H sẽ là phím tắt.
    - Tại cửa sổ Properties, thiết lập thuộc tính *UseMnemonic* là *True* (mặc định)
    - Tại cửa sổ Properties, thiết lập thuộc tính *TabIndex* là 1 giá trị *nhỏ hơn* giá trị TabIndex của điều khiển bạn muốn định nghĩa phím tắt cho (2 điều khiển không được cùng giá trị TabIndex)

3. Base Controls

lkb

- Điều khiển **LinkLabel**:
  - Cho phép tạo một đường link đến một Website hoặc thực hiện một số hành động
  - Các thuộc tính quan trọng của LinkLabel:

Property	Description
ActiveLinkColor	Thiết lập mày sắc của link kích hoạt (is being clicked)
LinkArea	Xác định phần text của label dùng làm hyperlink
LinkBehavior	Xác định kiểu gạch chân của link
LinkColor	Thiết lập màu sắc của link trước khi được click
LinkVisited	Xác định rằng link đã được thăm hay chưa
VisitedLinkColor	Thiết lập mày của link đã được thăm xong (has been visited). Chỉ xuất hiện màu khi LinkVisited

3. Base Controls

txt

- Điều khiển **TextBox**:
  - Dùng để nhận dữ liệu (văn bản) từ người dùng và hiển thị dữ liệu cho người dùng.
  - Các thuộc tính quan trọng của TextBox:

Property	Description
AutoCompleteCustomSource	Lưu trữ tập các xâu ký tự (mỗi xâu ký tự trên một dòng) để tự động điền (auto-complete) khi AutoCompleteMode khác *None* và AutoCompleteSource là *CustomSource*.
AutoCompleteMode	Thiết lập chế độ tự động hoàn thiện. Gồm các giá trị: None, Append (Bổ sung), Suggest (Gợi ý), SuggestAppend

3. Base Controls

- Điều khiển **TextBox**:
  - Dùng để nhận dữ liệu (văn bản) từ người dùng và hiển thị dữ liệu cho người dùng.
  - Các thuộc tính quan trọng của TextBox:

Property	Description
AutoCompleteSource	Thiết lập nguồn dữ liệu để điền tự động. Gồm các giá trị: None, FileSystem, History List, RecentlyUsedList, AllUrl, AllSystemSources, FileSystemDirectories, CustomSource
CharacterCasting	Có các giá trị None, Upper (chữ in hoa), Lower (chữ in thường).
MaxLength	Xác định độ dài của xâu ký tự sẽ được nhập vào TextBox
MultiLine	Xác định TextBox sẽ gồm 1 dòng hay nhiều dòng văn bản

Nguyễn Hữu Tuân - tuannh@utt.edu.vn

3. Base Controls

- Điều khiển **TextBox**:
  - Dùng để nhận dữ liệu (văn bản) từ người dùng và hiển thị dữ liệu cho người dùng.
  - Các thuộc tính quan trọng của TextBox:

Property	Description
Lines	Trả về 1 mảng các xâu ký tự biểu diễn từng dòng văn bản của textbox. Thường dùng khi thuộc tính MultiLine là *True*.Lưu ý rằng 1 dòng được định nghĩa như là 1 xâu ký tự được kết thúc bởi ký tự trả về và không liên quan đến các dòng hiện hữu trên UI, có thể quan sát được khi thuộc tính WordWrap là *True*.
PasswordChar	Thiết lập ký tự thay thế cho các ký tự thực sự của mật khẩu.

3. Base Controls

- Điều khiển **TextBox**:
  - Dùng để nhận dữ liệu (văn bản) từ người dùng và hiển thị dữ liệu cho người dùng.
  - Các thuộc tính quan trọng của TextBox:

Property	Description
ReadOnly	= True, textbox không thể chỉnh sửa = False (mặc định), textbox có thể hiệu chỉnh
ScrollBars	Gồm 4 giá trị: None, Horizontal, Vertical và Both
Text	Thiết lập hoặc lấy ra dữ liệu chứa trong textbox

3. Base Controls

- Điều khiển **TextBox**:
  - Dùng để nhận dữ liệu (văn bản) từ người dùng và hiển thị dữ liệu cho người dùng.
  - Các thuộc tính quan trọng của TextBox:

Property	Description
UseSystemPasswordChar	= True, sử dụng ký tự đại diện mật khẩu do hệ thống định nghĩa thay cho ký tự thực sự trong Textbox = False (mặc định)
WordWrap	= True (mặc định), tự động ngắt dòng khi thuộc tính MultiLine là True. = False, không ngắt dòng

3. Base Controls

mtb

- Điều khiển **MaskedTextBox**:
  - Giống TextBox nhưng cho phép định nghĩa trước các mẫu liệu để chấp nhận hoặc bác bỏ dữ liệu người dùng nhập vào.
  - Các thuộc tính quan trọng của MaskedTextBox:

Property	Description
AllowPromptAsInput	= True (mặc định), cho phép ký tự nhắc có giá trị như đầu vào
AsciiOnly	= True, chỉ các ký tự từ A đến Z và từ a đến z được chấp nhận = False (mặc định)
BeepOnError	= True, cho phép hệ thống phát ra tiếng beep với mỗi ký tự sai = False (mặc định)

3. Base Controls

- Điều khiển **MaskedTextBox**:
  - Các thuộc tính quan trọng của MaskedTextBox:

Property	Description
CutCopyMaskFormat	Xác định khi nào chữ và ký tự nhắc được tính đến khi văn bản bị cut và sao chép vào clipboard. Có 4 giá trị: IncludeLiterals, IncludePrompt, IncludePromptAndLiterals: Cả ký tự nhắc và chữ, ExcludePromptAndLiterals: Loại trừ cả ký tự nhắc và chữ
HidePromptOnLeave	= False (mặc định) = True, ẩn dấu nhắc khi điều khiển không còn con trỏ (lose focus)
InsertKeyMode	Xác định chế độ soạn thảo văn bản (Default, Insert, Overwrite)

3. Base Controls

- Điều khiển **MaskedTextBox**:
  - Các thuộc tính quan trọng của MaskedTextBox:

Property	Description
Mask	Định nghĩa các xâu định dạng cho đầu vào của MaskedTextbox
PromptChar	Thiết lập hoặc lấy ra ký tự được sử dụng làm ký tự nhắc (place holder)
RejectInputOnFirstFailure	= True, đầu vào sẽ bị bác bỏ khi 1 ký tự bị sai so với mask (dừng ngay khi có 1 ký tự bị sai) = False (mặc định), từng ký tự trong văn bản sẽ được lần lượt xử lý với vai trò như các đầu vào riêng rẽ.
ResetOnPrompt	= True (mặc định), thiết lập lại và nhảy qua vị trí hiện tại nếu có thể edit khi ký tự đầu vào có cùng giá trị như ký tự nhắc → Xác định đầu vào phù hợp với ký tự nhắc nên được xử lý ntn

3. Base Controls

- Điều khiển **MaskedTextBox**:
  - Các thuộc tính quan trọng của MaskedTextBox:

Property	Description
ResetOnSapce	= True (mặc định), thiết lập lại và nhảy qua ký tự hiện tại nếu có thể edit khi đầu vào là ký tự trắng → Xác định dấu trắng nhập vào sẽ được xử lý ntn
SkipLiterals	= True (mặc định), nhảy qua bị trí hiện tại nếu không thể edit và ký tự đầu vài có cùng giá trị như là văn bản tại vị trí đó → Xác định khi nào văn bản trong mask sẽ được nhập lại (reenter) hoặc bỏ qua (skip)
TextMaskFormat	Xác định khi nào chữ và ký tự nhắc được thêm vào văn bản trả về bởi thuộc tính Text. Có 4 giá trị: IncludeLiterals, IncludePrompt, IncludePromptAndLiterals, ExcludePromptAndLiterals.

3. Base Controls

- Điều khiển **MaskedTextBox**:
  - Ví dụ cho thuộc tính Mask:

Mask String	Input Text	Displayed Text
(999)-000-0000	1234567890	(123)-456-7890
00/00/0000	07141999	07/14/1999 (Chú ý, dấu / được thiết lập bởi FormatProvider)
$99,999.00	12345	$12,345.00
LL>L|LLL<LL	abcdABCD	abCdABcd

3. Base Controls

chk

- Điều khiển **CheckBox**:
  - Các thuộc tính quan trọng:

Property	Description
AutoCheck	= True, tự động đánh dấu khi tiêu đề của CheckBox được ấn
Checked	= True, CheckBox đang được chon
CheckState	Trả về trạng thái của điều khiển, các giá trị gồm: *Checked, Unchecked* (mặc định), *Indeterminate* (ô vuông có màu xanh)
Text	Tiêu đề (văn bản) hiển thị gần ô vuông
ThreeState	= False (mặc định), CheckBox cho phép 2 trạng thái chứ không phải 3 trạng thái

3. Base Controls

- Điều khiển **RadioButton**: **rdb**
  - Mọi RadioButton cùng nằm trong 1 điều khiển chứa sẽ loại trừ nhau, nghĩa là trong 1 thời điểm chỉ có 1 RadioButton được chọn.
  - Các thuộc tính quan trọng:

Property	Description
Checked	= True, điều khiển đang được chọn
Text	Văn bản hiển thị gần nút ấn

4. Advanced Controls – Part 1

lbx

- Điều khiển **ListBox**:
  - Là điều khiển hiển thị (danh sách) đơn giản nhất. Người dùng có thể chọn 1 hoặc nhiều mục (item) trong ListBox
  - Các thuộc tính quan trọng của ListBox:

Property	Description
DataSource	Thiết lập nguồn cho dữ liệu ràng buộc với ListBox
DisplayMember	Thành phần dữ liệu được hiển thị
FormatString	Định rõ chuỗi định dạng sẽ được sử dụng để định dạng mục từ (entries) của ListBox nếu FormattingEnabled thiết lập là True
FormattingEnabled	= True (mặc định), giá trị của FormatString sẽ được sử dụng để convert giá trị của DisplayMember thành 1 giá trị có thể được hiển thị

4. Advanced Controls – Part 1

- Điều khiển **ListBox**:
  - Các thuộc tính quan trọng của ListBox:

Property	Description
Items	Tập đối tượng (collection of items) được chứa trong ListBox
MultiColumn	= False (mặc định), các giá trị sẽ hiển thị trong 1 cột = True, các giá trị sẽ được hiển thị thành các cột theo chiều ngang
SelectionMode	Xác định số lượng đối tượng có thể được chọn. Các giá trị là None, One (mặc định), MultiSimple (cho phép chọn nhiều đối tượng), MultiExtend (cho phép sử dụng Shift và Ctrl khi chọn nhiều đối tượng)
ValueMember	Chỉ ra thành phần dữ liệu sẽ cung cấp các giá trị thực sự cho ListBox

4. Advanced Controls – Part 1

- Điều khiển **ListBox**:
  - Các thuộc tính quan trọng của ListBox:

Property	Description
ColumnWidth	Xác định độ rộng của mỗi cột sẽ được hiển thị trong ListBox nhiều cột (multicolumn ListBox)
ContextMenuStrip	Menu ngữ cảnh (Shortcut Menu) hiển thị khi người dùng click phải chuột vào ListBox
Cursor	Chọn hình dạng con trỏ khi chuột di qua ListBox
SelectedIndex	Lấy ra chỉ số (index) của item được chọn hoặc nếu thuộc tính SelectionMode là MultiSimple hoặc MultiSelect thì sẽ trả về bất kỳ index nào đã được chọn.
SelectedIndices	Trả về tập hợp tất cả các index được chọn

4. Advanced Controls – Part 1

- Điều khiển **ListBox**:
  - Các thuộc tính quan trọng của ListBox:

Property	Description
SelectedItem	Trả về đối tượng được chọn hoặc nếu SelectionMode là MultiSimple hoặc MultiSelect sẽ trả về bất kỳ đối tượng được chọn nào
SelectedItems	Trả về tập tất cả các đối tượng được chọn
SelectedValue	Trong 1 điều khiển ràng buộc dữ liệu, trả về giá trị kết hợp với các đối tượng được chọn. Nếu điều khiển không ràng buộc dữ liệu, hoặc nếu ValueMember không được thiết lập, thuộc tính này sẽ trả về giá trị ToString của đối tượng được chọn

4. Advanced Controls – Part 1

cbo

- Điều khiển **ComboBox**:
  - Tương tự như ListBox, nhưng ngoài việc cho phép người dùng chọn các đối tượng trong danh sách, ComboBox cung cấp khoảng trống để người dùng nhập giá trị vào giống như việc chọn từ danh sách. ComboBox có thể được cấu hình để vừa hiển thị như một danh sách lựa chọn hoặc cung cấp một danh sách thả xuống.
  - Các thuộc tính quan trọng của ComboBox:

Property	Description
DataSource	Thiết lập nguồn cho dữ liệu ràng buộc với ComboBox
DisplayMember	Thành phần dữ liệu được hiển thị
DropDoxnHeight	Thiết lập độ cao (pixel) tối đa cho ô thả xuống (drop-down box)

4. Advanced Controls – Part 1

- Điều khiển **ComboBox**:
  - Các thuộc tính quan trọng của ComboBox:

Property	Description
DropDownStyle	Xác định kiểu dáng của ComboBox. Các giá trị là Simple (tương tự như ListBox), DropDown (mặc định), DropDownList (tương tự như DropDown nhưng không cho phép người dùng nhập giá trị mới)
DropDownWidth	Thiết lập độ rộng của vùng thả xuống của ComboBox
MaxDropDownItems	Số lượng tối đa đối tượng được hiển thị trong danh sách thả xuống

4. Advanced Controls – Part 1

- Điều khiển **ComboBox**:
  - Các thuộc tính quan trọng của ComboBox:

Property	Description
SelectionMode	Xác định số lượng đối tượng có thể được chọn. Các giá trị là None, One (mặc định), MultiSimple (cho phép chọn nhiều đối tượng), MultiExtend (cho phép sử dụng Shift và Ctrl khi chọn nhiều đối tượng)
SelectedIndex	Lấy ra chỉ số (index) của item được chọn hoặc nếu thuộc tính SelectionMode là MultiSimple hoặc MultiSelect thì sẽ trả về bất kỳ index nào đã được chọn.
SelectedIndices	Trả về tập hợp tất cả các index được chọn

4. Advanced Controls – Part 1

- Điều khiển **ComboBox**:
  - Các thuộc tính quan trọng của ComboBox:

Property	Description
SelectedItem	Trả về đối tượng được chọn hoặc nếu SelectionMode là MultiSimple hoặc MultiSelect sẽ trả về bất kỳ đối tượng được chọn nào
SelectedItems	Trả về tập tất cả các đối tượng được chọn
SelectedValue	Trong 1 điều khiển ràng buộc dữ liệu, trả về giá trị kết hợp với các đối tượng được chọn. Nếu điều khiển không ràng buộc dữ liệu, hoặc nếu ValueMember không được thiết lập, thuộc tính này sẽ trả về giá trị ToString của đối tượng được chọn

4. Advanced Controls – Part 1

ckl

- Điều khiển **CheckedListBox**:
  - Dùng để hiển thị danh sách cho phép người dùng có thể chọn nhiều đối tượng bằng check vào các ô đặt cạnh gần đối tượng
  - Các thuộc tính quan trọng của CheckedListBox:

Property	Description
CheckedIndices	Trả về 1 tập tất cả các chỉ mục được check
CheckedItems	Trả về 1 tập tất cả các đối tượng được check
Items	Trả về tập các đối tượng được chứa trong điều khiển
SelectedIndex	Lấy ra chỉ số (index) của item được chọn hoặc nếu thuộc tính SelectionMode là MultiSimple hoặc MultiSelect thì sẽ trả về bất kỳ index nào đã được chọn.
SelectedItem	Trả về đối tượng được chọn hoặc nếu SelectionMode là MultiSimple hoặc MultiSelect sẽ trả về bất kỳ đối tượng

4. Advanced Controls – Part 2

lv

- Điều khiển **ListView**:
  - Cho phép tạo một danh sách các mục có thể kết hợp icon (giống Windows Explorer)
  - Các thuộc tính quan trọng của ListView:

Property	Description
Column	Chứa tập hợp các cột sẽ được hiển thị khi thuộc tính View là Details
Groups	Chứa 1 tập hợp các nhóm (tùy chọn) ;có thể được dùng để phân nhóm các mục được chứa ở Items
Items	Tập hợp các ListViewItems được hiển thị trong điều khiển

4. Advanced Controls – Part 2

- Điều khiển **ListView**:
  - Các thuộc tính quan trọng của ListView:

Property	Description
LargeImageList	Điều khiển ImageList được dùng cho ListView. Tác dụng khi thuộc tính View bằng LargeIcon
ShowGroups	= True, hiển thị các mục theo dạng nhóm
SmallImageList	Điều khiển ImageList được dùng cho ListView để lấy các ảnh trong mọi định dạng trừ khi chọn biểu tượng to (large icon).
View	Chọn 1 định dạng hiển thị các mục. Gồm 5 giá trị: LargeIcon, Details, SmallIcon, List, Title

4. Advanced Controls – Part 2

trv

- Điều khiển **TreeView**:
  - Cho phép hiển thị danh sách đối tượng theo cấu trúc phân cấp (cấu trúc cây). Mỗi đối tượng của điều khiển TreeView biểu diễn 1 thể hiện của lớp TreeNode chứa đựng các thông tin về vị trí của nút trong TreeView. Các nút có thể có các nút con (có thể thu nhỏ hay mở rộng để xem)
  - Thuộc tính cơ bản của TreeView là Nodes. Thuộc tính này chứa tập hợp của TreeNodes bao gồm cả gốc, mỗi đối tượng TreeNode lại có thể chứa tập các TreeNode (nút con) của chính nó.

4. Advanced Controls – Part 2

- Điều khiển **NumericUpDown**: **nud**
  - Cho phép thiết lập 1 miền giá trị các số để người dùng có thể duyệt qua hoặc lựa chọn bằng cách ấn vào nút lên hoặc xuống để tăng hoặc giảm giá trị số.
  - Một số thuộc tính quan trọng của NumericUpDown:

Property	Description
Hexadecimal	= True, các giá trị số ở hệ 16
Increment	Thiết lập hoặc lấy ra bước nhảy cho mỗi lần tăng hoặc giảm giá trị
Maximum	Chỉ định giá trị lớn nhất
Minimum	Chỉ định giá trị nhỏ nhất
ThousandsSeparator	=True, Hiển thị dấu phân cách phần nghìn
Value	Thiết lập hoặc lấy ra giá trị hiện tại của điều khiển

4. Advanced Controls – Part 2

- Điều khiển **DateTimePicker**: **dtp**
  - Cho phép thiết lập 1 giá trị ngày, giờ dễ dàng bằng 1 giao diện đồ họa gần giống ComboBox
  - Một số thuộc tính quan trọng:

Property	Description
CustomFormat	Định dạng tùy chọn được dùng khi thuộc tính Format là Custom
Format	Thiết lập định dạng hiển thị. Gồm: *Long* (định dạng ngày đầy đủ), *Short* (định dạng ngày thu gọn), *Time* (chỉ hiển thị ngày), hoặc *Custom* (hiển thị theo thuộc tính CustomFormat)
MaxDate	Giá trị lớn nhất nhận được
MinDate	Giá trị nhỏ nhất nhận được
Value	Giá trị hiện tại của điều khiển

4. Advanced Controls – Part 2

mc

- Điều khiển **MonthCalendar**:
  - Là 1 định dạng hữu hiệu để chọn ngày tháng
  - Một số thuộc tính quan trọng:

Property	Description
AnnuallyBolded-Dates	Chứa 1 mảng các giá trị ngày và giờ sẽ được bôi đậm mỗi năm
BoldedDates	Chứa 1 mảng các giá trị ngày và giờ sẽ được tô đậm
FirstDayOfWeek	Xác định ngày sẽ được thiết lập là ngày đầu tiên của tuần trong điều khiển MonthCalendar
MaxDate	Ngày lớn nhất có thể được chọn
MinDate	Ngày nhỏ nhất có thể được chọn
MaxSelection-Count	Thiết lập số ngày tối đa có thể được chọn

4. Advanced Controls – Part 2

- Điều khiển **MonthCalendar**:
  - Một số thuộc tính quan trọng:

Property	Description
MonthlyBolded-Dates	Chứa 1 mảng các giá trị ngày và giờ sẽ được tô đậm tại mỗi tháng
SelectionEnd	Xác định ngày giờ kết thúc của thuộc tính SelectionRange
SelectionRange	Chứa miền giá trị ngày tháng được chọn bởi người dùng
SelectionStart	Xác định ngày giờ bắt đầu của thuộc tính SelectionRange

4. Advanced Controls – Part 2

pb

- Điều khiển **PictureBox**:
  - Dùng để hiển thị hình ảnh ở nhiều định dạng
  - Một số thuộc tính quan trọng:

Property	Description
ErrorImage	Hình ảnh hiển thị nếu ảnh được chọn không load được
Image	Hình ảnh sẽ được load vào PictureBox
ImageLocation	Trang Web hoặc nơi lưu trữ ảnh
InitialImage	Hình ảnh hiển thị trong PictureBox khi hình ảnh đang được load
SizeMode	Cách thức hình ảnh được bố trí và thiết lập kích thước

4. Advanced Controls – Part 3

- Điều khiển **ToolStrip**:
  - Cho phép tạo các thanh công cụ (toolbar) chuyên nghiệp và dễ dàng với các ToolStripItems là các control
  - Một số thuộc tính quan trọng:

Property	Description
AllowItemReorder	= true, các điều khiển trong ToolStrip có thể thay đổi thứ tự khi người dùng giữ và ấn phím Alt đồng thời với ấn chuột
AllowMerge	Toolstrip có thể được gộp với toolstrip khác không
TextDirection	Xác định hướng chữ của các đối tượng thuộc ToolStrip
Stretch	= true, các control thuộc ToolStrip sẽ dãn kích thước cho phù hợp với kích thước của ToolStrip

4. Advanced Controls – Part 3

- Điều khiển **MenuStrip**:
  - Cho phép tạo các thực đơn (menu)
  - Một số thuộc tính quan trọng:

Property	Description
AllowItemReorder	= true, các điều khiển có thể thay đổi thứ tự khi người dùng giữ và ấn phím Alt đồng thời với ấn chuột
AllowMerge	Menustrip có thể được gộp với toolstrip khác không
TextDirection	Xác định hướng chữ của các đối tượng thuộc ToolStrip
Stretch	= true, các control thuộc ToolStrip sẽ dãn kích thước cho phù hợp với kích thước của ToolStrip

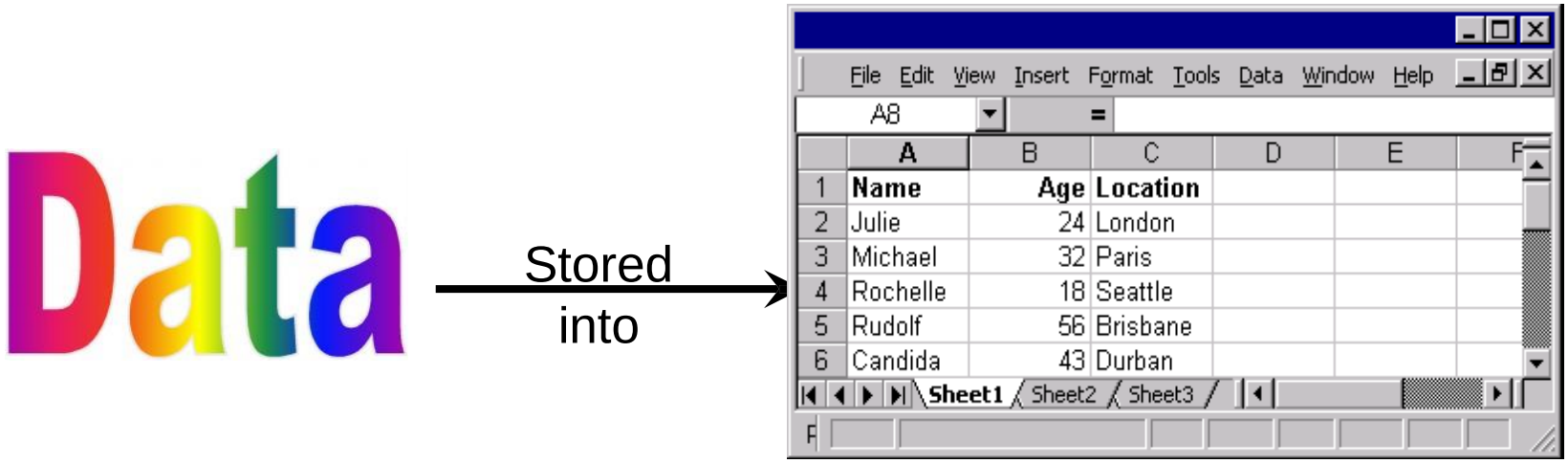
LẬP TRÌNH CSDL VỚI

ADO.NET

Tổng quan ADO.NET

- ActiveX Data Object .NET (ADO.NET)
  - Công nghệ của Microsoft
  - Là 1 phần của .NET Framework
  - Phát triển từ nền tảng ADO
  - Cung cấp các lớp đối tượng và các hàm thư viện phục vụ cho việc kết nối và xử lý dữ liệu

Tổng quan ADO.NET



Tổng quan ADO.NET

- Đặc điểm chính của ADO.NET là làm việc với dữ liệu **không kết nối**, dữ liệu được lưu trữ trong bộ nhớ như một CSDL thu nhỏ gọi là DataSet, nhằm tăng tốc độ tính toán, xử lý tối đa và hạn chế việc sử dụng tài nguyên trên Database Server.
- Đặc điểm quan trọng thứ 2 là khả năng xử lý dữ liệu dạng chuẩn XML (eXtensible Markup Language – chuẩn giao tiếp dữ liệu tốt nhất hiện nay trên môi trường Internet), dữ liệu ở dạng XML có thể trao đổi giữa bất kỳ hệ thống nào giúp ứng dụng có nhiều khả năng làm việc với nhiều ứng dụng khác

Kiến trúc ADO.NET

- Gồm 2 phần chính:
  - Managed Provider Component: bao gồm các đối tượng như DataAdapter, DataReader,… giữ nhiệm vụ làm việc trực tiếp với dữ liệu như database, file…
  - Content Component: bao gồm các đối tượng như DataSet, DataTable,… đại diện cho dữ liệu thực sự cần làm việc.

Kiến trúc ADO.NET

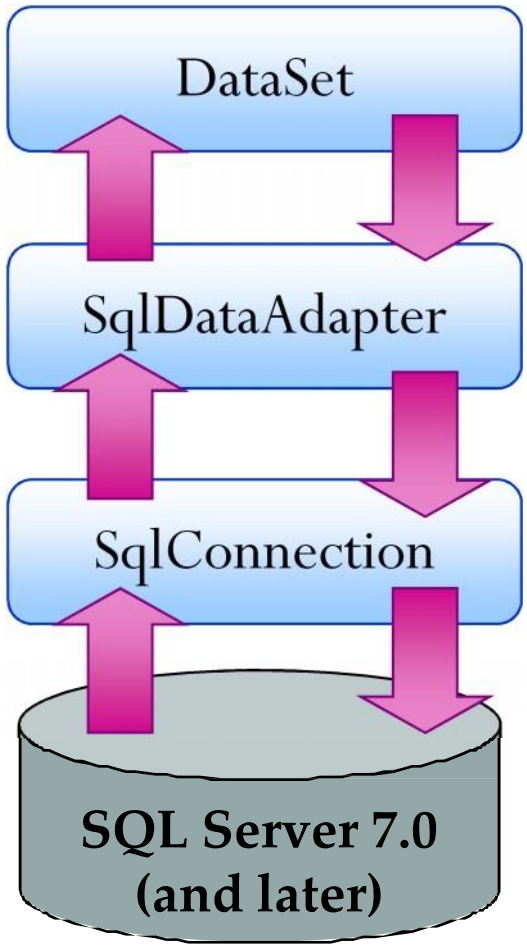
- Gồm 2 phần chính:
  - DataReader là một đối tượng mới giúp truy cập dữ liệu nhanh chóng nhưng forward-only và read-only giống như ADO RecordSet sử dụng Server cursor, OpenForwardOnly và LockReadOnly.
  - DataSet cũng là 1 đối tượng mới, không chỉ là dữ liệu, DataSet có thể coi là một bản sao gọn nhẹ của CSDL trong bộ nhớ với nhiều bảng và các mối quan hệ.
  - DataAdapter là đối tượng kết nối giữa DataSet với CSDL, nó bao gồm 2 đối tượng Connection và Command để cung cấp dữ liệu cho DataSet cũng như cập nhật dữ liệu từ DataSet xuống CSDL.

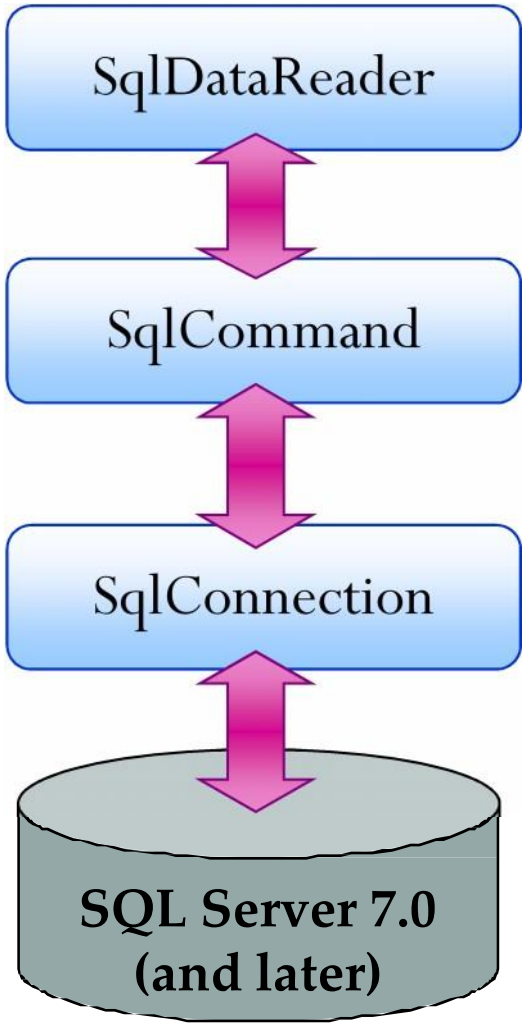
Connected và Disconected

- Connected Model

- Disconnected Model

Connected và Disconected





Connected và Disconected

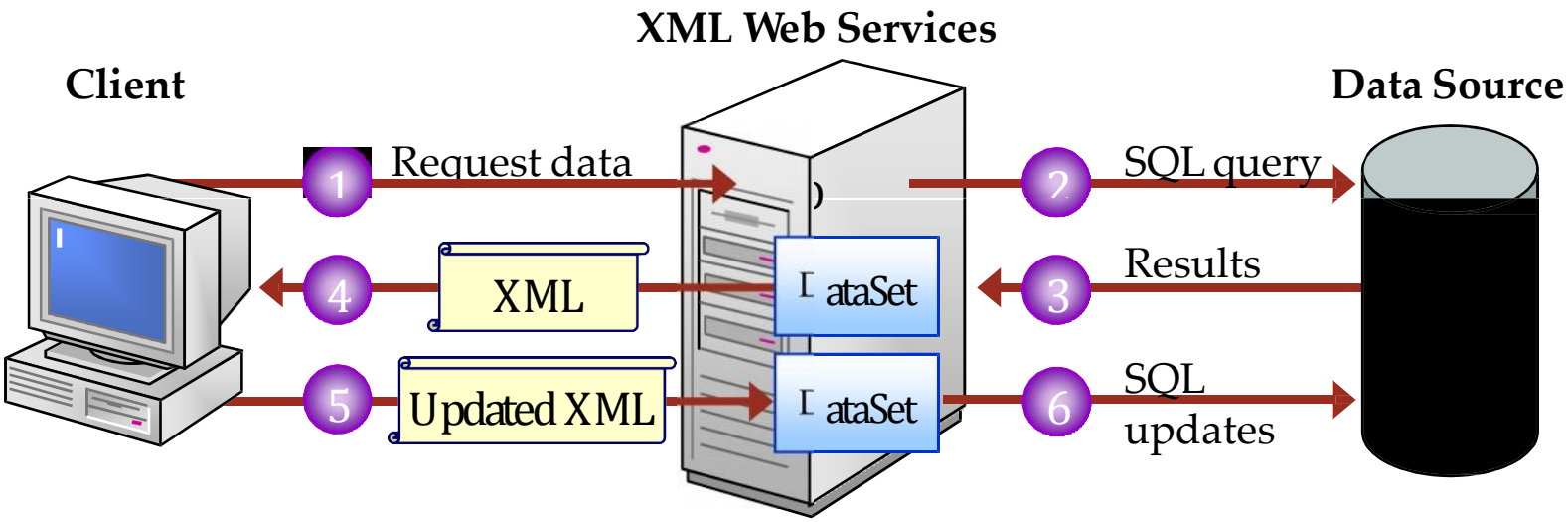
- Môi trường Connected
  - Mỗi user có một kết nối cố định tới Data Source
  - Ưu điểm:
    - Môi trường được bảo vệ tốt
    - Kiểm soát được sự đồng bộ
    - Dữ liệu luôn được mới
  - Nhược điểm:
    - Phải có một kết nối mạng cố định
    - Scalability

Connected và Disconected

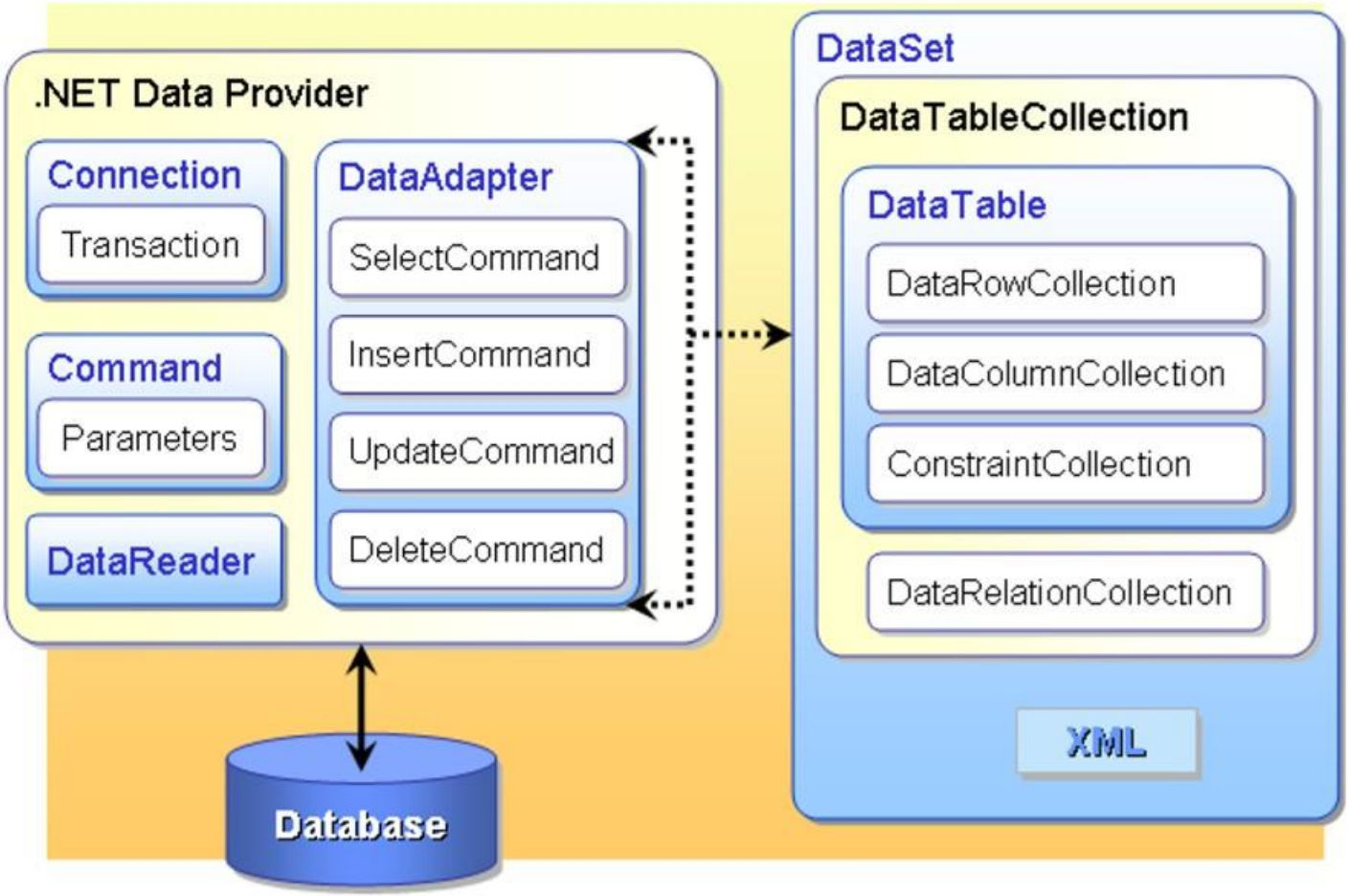
- Môi trường Disconnected
  - Cho phép lấy cả 1 cấu trúc phức tạp của dữ liệu từ CSDL, sau đó ngắt kết nồi rồi mới thực hiện thao tác xử lý
  - Nói cách khác: một tập con của dữ liệu trung tâm được sao chép và bổ sung độc lập, sau đó sẽ được trộn (merge) lại vào dữ liệu trung tâm
  - Ưu điểm:
    - Có thể làm việc bất cứ lúc nào, cũng như có thể kết nối bất kỳ vào Data Source
    - Cho phép user khác có thể kết nối
    - Nâng cao hiệu suất thực hiện của ứng dụng
  - Nhược điểm:
    - Dữ liệu không được cập nhật 1 cách nhanh nhất
    - Sự tranh chấp có thể xuất hiện và phải giải quyết

ADO.NET và XML

- Sử dụng XML ở ứng dụng ADO.NET Disconnected



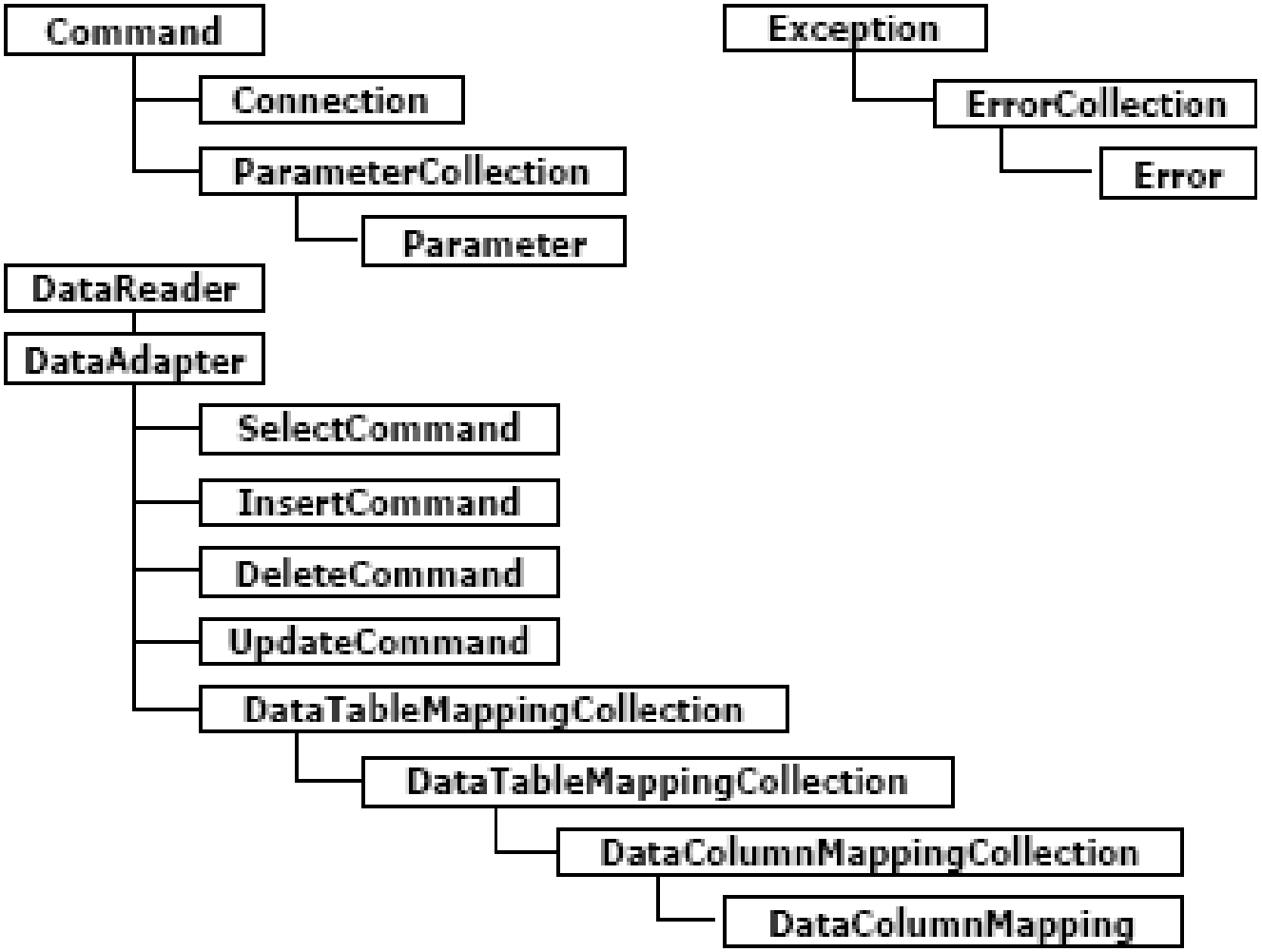
Mô hình đối tượng của ADO.NET



Managed Provider Component

- Mục đích chính là kết nối CSDL (còn gọi là Phần kết nối): sử dụng khi kết nối CSDL và thao tác dữ liệu. Phải thực hiện kết nối khi thao tác:
  - Connection: Quản lý việc đóng mở DB. Ví dụ: **???Connection**, **SqlConnection**, **OleDbConnection**…
  - Command: Lệnh truy vấn, tương tác dữ liệu khi đang lập kết nối. Ví dụ: **???Command**, **SqlCommand**, **OleDbCommand**…
  - DataReader: Đọc dữ liệu, chỉ xử lý 1 dòng dữ liệu tại một thời điểm. Ví dụ: **??? DataReader**, **SqlDataReader**, **OleDbDataReader**…
  - DataAdapter: Cầu nối giữa DB và DataSet

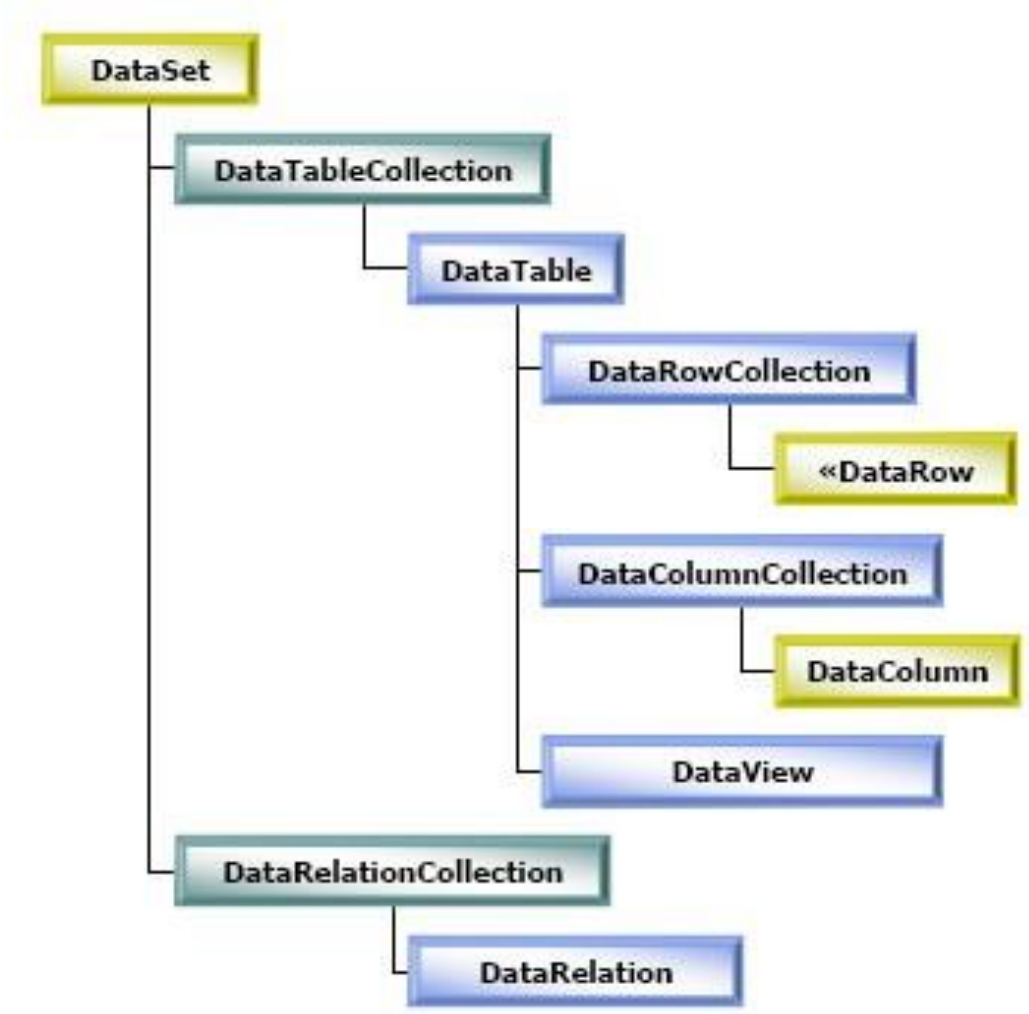
Managed Provider Component



Content Component

- Content Component là các đối tượng đại diện cho dữ liệu cần xử lý.
- Trong ADO.NET dữ liệu được đại diện bởi DataSet dưới hình ảnh của một CSDL thu gọn.
- Các class chính: DataSet, DataTable, DataView, DataRow, DataColumn, DataRelation và các đối tượng nhóm như: DataTableCollection, DataRowCollection, DataColumnCollection
- Phần ngắt kết nối: là DataSet. DataSet được xem như 1 DB trong bộ nhớ. DataSet không quan tâm đến DB thuộc kiểu gì, và lấy dữ liệu từ DataAdapter để xử lý

Content Component



Content Component

Namespace

- System.Data: Tất cả các lớp truy cập dữ liệu nói chung
- System.Data.Common: Các lớp được shared (hoặc overridden) bởi các data provider riêng lẻ.
- System.Data.Odbc: ODBC provider classes
- System.Data.OleDb: OLE DB provider classes
- System.Data.ProviderBase: New base classes and connection factory classes
- System.Data.Oracle: Oracle provider classes
- System.Data.Sql: New generic interfaces and classes for SQL Server data access
- System.Data.SqlClient: SQL Server provider classes
- System.Data.SqlTypes: SQL Server data types

Các lớp thư viện ADO.NET

- System.Data.OleDb: Làm việc được với Access, SQL Server, Oracle
- System.Data.SqlClient: Làm việc với SQL Server
- System.Data.OracleClient: Làm việc với Oracle
- Cả 3 thư viện về giao tiếp lập trình là giống nhau
- Dùng thư vieennj SqlClient truy xuất SQL Server sẽ nhanh hơn OleDb, tương tự với OracleClient

KẾT NỐI CSDL – Connected Model

- Các lớp phụ trách kết nối:
  - ODBCConnection: (Open DataBase Connectivity) như Data Source Name (DNS) được định nghĩa trong hộp thoại ODBC Data Source Administrator
  - OleDBConnection: Các nguồn dữ liệu OLE DB như Office Access (Jet 4.0)…
  - SqlConnection: SQL Server 7.0 trở đi
  - OracleConnection: Oracle 7.3, 8i, 9i
- Có 2 cách tạo và cấu hình đối tượng kết nối:
  - Thông qua giao diện người dùng (UI), sử dụng hộp thoại Add Connection
  - Thông qua câu lệnh một cách thủ công (code)

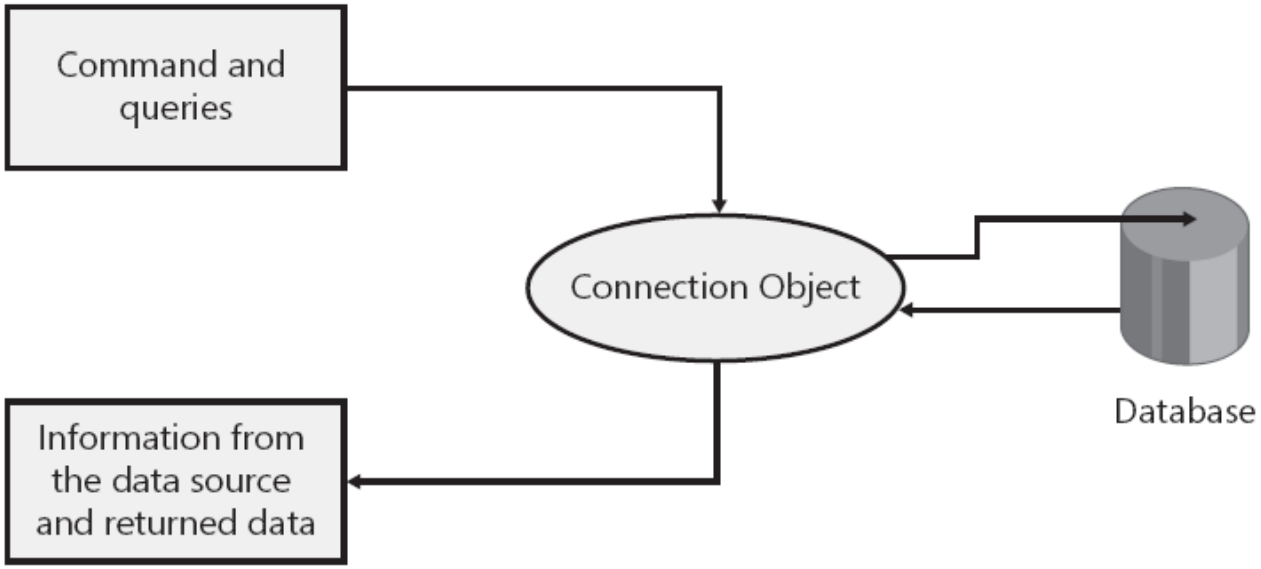
Kết nối CSDL

1. Creating Connection Objects

- Nội dung mục 1:
  - Cấu hình một kết nối tới CSDL sử dụng Server Explorer
  - Cấu hình một kết nối tới CSDL sử dụng Data Source Configuration Wizard
  - Cấu hình một kết nối tới CSDL sử dụng lớp *Connection*.
  - Kết nối tới CSDL sử dụng các đối tượng kết nối CSDL cụ thể.

1. Creating Connection Objects

- Connection Objects:
  - Là kênh truyền thông giữa CSDL với ứng dụng
  - Biểu diễn một kết nối mở đến nguồn dữ liệu
  - Không lấy hoặc cập nhật dữ liệu, không thực hiện truy vấn, không bao gồm kết quả các câu truy vấn



1. Creating Connection Objects

- Tạo kết nối trong Server Explorer
  - Server Explorer: là trung tâm quản lý các kết nối dữ liệu độc lập với bất kỳ 1 project nào (nói cách khác các kết nối trong SE có thể được truy cập bởi bất kỳ 1 project nào)
  - Các kết nối được tạo ra trong SE là các thiết lập đặc biệt để hiển thị các kết nối mỗi khi mở VS thay vì tạo các kết nối như 1 phần của 1 ứng dụng cụ thể.

1. Creating Connection Objects

- Tạo kết nối trong Server Explorer
  - B1: View / Server Explorer…
  - B2: R-click vào node Data Connections và chọn Add Connection. Trong lần đầu tiên thêm 1 connection trong Visual Studio, hộp thoại Choose Data Source xuất hiện. Hộp thoại Choose Data Source dùng để chọn nguồn dữ liệu muốn kết nối, cũng như provider dùng để kết nối.
  - Chú ý: nếu hộp thoại Add Connection xuất hiện hay vì hộp thoại Choose Data Source, chọn nút Change ở top của hộp thoại Add Connection.
  - B3: Chọn nguồn dữ liệu (Data Source) là Microsoft SQL Server và click OK (hoặc Continue). Hộp thoại Add Connection xuất hiện với SQL Server là nguồn dữ liệu.

1. Creating Connection Objects

- Tạo kết nối trong Server Explorer
  - Chú ý: .NET Framework Data Providers cho SQL Server được thiết kế cho SQL Server 7.0 trở đi, nếu muốn kết nối với SQL Server 6.0 trở về trước thì chọn nguồn dữ liệu là <other> và chọn .NET Framework Data Provider cho OLE DB. Sau đó, trong hộp thoại Add Connection chọn Microsoft OLE DB Provider for SQL Server
  - B4: Nhập tên của SQL Server trong vùng tên server
  - B5: Chọn phương thức thích hợp để xác thực truy cập vào SQL Server
  - B6: Chọn tùy chọn Select Or Enter A Database Name và chọn CSDL muốn kết nối từ danh sách đổ xuống

1. Creating Connection Objects

- Tạo kết nối trong Server Explorer
  - Nút Test Connection để kiểm tra kết nối và sau đó ấn OK để đóng hộp thoại và tạo kết nối trong Server Explorer. Sau khi tạo kết nối, cửa sổ Propertise cung cấp các thông tin liên quan đến kết nối cũng như CSDL thực sự được kết nối đến
  - Chọn kết nối vừa tạo trong Server Explorer để xem các thông tin hiện tại trong cửa sổ thuộc tính. Các thuộc tính này tùy thuộc vào kiểu nguồn dữ liệu cũng như là trạng thái của kết nối. Ví dụ nếu kết nối đã bị close thì chỉ có thể xem xâu kết nối, .NET Framework data provider được dùng và trạng thái của kết nối… Để xem thêm các thuộc tính khác, cần mởi kết nối bằng cách mở rộng nút Connection trong SE.

1. Creating Connection Objects

- Tạo kết nối sử dụng Data Wizards
  - Wizard cơ bản là Data Source Configuration Wizard
  - Phải taọ sẵn 1 ứng dụng Windows Forms
  - Vào menu Data / Add New Data Source…
  - Nguồn dữ liệu mặc định là Database, sau đó chọn Next. Tiếp đến chọn DataSet và Next
  - Ấn New Connection để mở hộp thoại Add Connection và làm tương tự trong SE
  - Tại Choose Your Database Objects, chọn Tables, Views, Stored Procedures, v…v… sẽ được sử dụng trong ứng dụng. (Mở rộng đối tượng Tables để chọn các bảng bên trong).
  - Ấn Finish. Một dataset với đối tượng kết nối vừa được tạo sẽ được thêm vào ứng dụng.

1. Creating Connection Objects

- Tạo kết nối trong chế độ run-time (Code)
  - Các thuộc tính chung cho mọi đối tượng kết nối:

PROPERTIES	MÔ TẢ TÁC DỤNG
ConnectionString	Thiết lập hoặc lấy ra xâu kết nối
ConnectionTimeOut	*(Chỉ đọc)* Lấy ra thời gian chờ khi đang thiết lập 1 kết nối trước khi ngừng việc thử lại và phát sinh lỗi
Database	*(Chỉ đọc)* Lấy ra tên của Database Server được kết nối
Server Version	*(Chỉ đọc)* Lấy ra xâu ký tự biểu diễn phiên bản của server của đối tượng được kết nối
State	*(Chỉ đọc)* Lấy ra giá trị biểu diễn trạng thái của kết nối (System.Data.ConnectionState)

1. Creating Connection Objects

- Tạo kết nối trong chế độ run-time (Code)
  - Các phương thức chung cho mọi đối tượng kết nối:

METHOD	MÔ TẢ TÁC DỤNG
BeginDbTransaction	Bắt đầu một phiên làm việc CSDL
BeginTransaction	Bắt đầu một phiên làm việc CSDL
ChangeDatabase	Thay đổi CSDL hiện tại của 1 kết nối đang mở
Close	Đóng kết nối tới CSDL. *(Đây là phương thức ưa thích để đóng bất kỳ một kết nối đang mở)*
CreateCommand	Tạo và trả về 1 đối tượng *System.Data.Common.DbCommand* của kết nối hiện tại

1. Creating Connection Objects

- Tạo kết nối trong chế độ run-time (Code)
  - Các phương thức chung cho mọi đối tượng kết nối:

METHOD	MÔ TẢ TÁC DỤNG
CreateDbCommand	Tạo và trả về 1 đối tượng *System.Data.Common.DbCommand* của kết nối hiện tại
EnlistTransaction	Gia nhập thêm 1 phiên làm việc cụ thể cho 1 phiên làm việc phân tán.
GetSchema	Trả về giản đồ thông tin cho nguồn dữ liệu của lớp *System.Data.Common.DbConnection*
New	Khởi tạo một thể hiện của lớp *System.Data.Common.DbConnection*

1. Creating Connection Objects

- Tạo kết nối trong chế độ run-time (Code)
  - Các phương thức chung cho mọi đối tượng kết nối:

METHOD	MÔ TẢ TÁC DỤNG
OnStateChange	Phát sinh sự kiện *System.Data.Common.DbConnection.StateChange*
Open	Mở 1 kết nối CSDL với các thiết lập cụ thể trong *System.Data.Common.DbConnection..ConnectionString*

1. Creating Connection Objects

- Tạo kết nối trong chế độ run-time (Code)
  - Các sự kiện chung cho mọi đối tượng kết nối:

METHOD	MÔ TẢ TÁC DỤNG
StateChange	Xảy ra khi trạng thái của kết nối thay đổi (từ Open thành Closed)
InfoMessage	Xảy ra khi server trả về 1 cảnh báo hoặc 1 thông báo. Thông báo thông thường phát sinh khi có các lỗi mức độ thấp (<10 với SQL) ví dụ như kết nối vào đối tượng đã kết nối…

1. Creating Connection Objects

- Tạo 1 đối tượng kết nối SQL Server trong Code (Creating SQL Server Connection Objects in Code)
  - Tạo một ứng dụng Windows và 1 form để minh họa kết nối (ví dụ form có tên DataConnections để kết nối với SQL Server)
  - Mở form vừa tạo trong code view
  - Viết code cho đối tượng kết nối

1. Creating Connection Objects

- Tạo 1 đối tượng kết nối SQL Server trong Code (Creating SQL Server Connection Objects in Code)

```
//Tham chiếu namespace
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
```

1. Creating Connection Objects

- Tạo 1 đối tượng kết nối SQL Server trong Code (Creating SQL Server Connection Objects in Code)

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

1. Creating Connection Objects

- Tạo 1 đối tượng kết nối SQL Server trong Code (Creating SQL Server Connection Objects in Code)

```
// Declare the connection objects for the for SQL Server provider in Windows Authentication

private SqlConnection ConnectionToSql = new
SqlConnection( "Data Source=.\\sqlexpress;Initial
Catalog=Northwind;Integrated Security=True");
}
```

1. Creating Connection Objects

- Tạo 1 đối tượng kết nối SQL Server trong Code (Creating SQL Server Connection Objects in Code)

```
// Declare the connection objects for the for SQL Server provider in SQL Server Logins

private SqlConnection ConnectionToSql = new SqlConnection( "Persist Security Info = False; User ID = sa; Password = 123456; Initial Catalog=Northwind; Server=sqlexpress");
}
```

1. Creating Connection Objects

- Tạo 1 đối tượng kết nối SQL Server trong Code (Creating SQL Server Connection Objects in Code)

```
// Open connection to a database, Open method
        ConnectionToSql.Open();
```

1. Creating Connection Objects

- Tạo 1 đối tượng kết nối SQL Server trong Code (Creating SQL Server Connection Objects in Code)

```
// Close connection to a database, Close method
        ConnectionToSql.Close();
```

2. Securing Sensitive CS Data

- Bảo mật dữ liệu nhạy cảm của chuỗi kết nối (Securing Sensitive Connection String Data):
  - Phương thức đề xuất là sử dụng Windows Authentication (còn gọi là Integrated Security)
  - Thêm nữa, nên thiết lập từ khóa *Persist Security Information* trong chuỗi kết nối thành *False*. Điều này đảm bảo rằng những ủy nhiệm được dùng để mởi kết nối đều bị loại bỏ và không được lưu trữ tại nơi có thể tìm thấy.

Làm việc với dữ liệu trong môi trường kết nối

Làm việc trong môi trường kết nối

- Working with Data in a Connected Environment:
  - Tạo và thực thi đối tượng Command
  - Làm việc với tham số trong câu lệnh SQL
  - Lưu và truy cập các giá trị BLOB (Birnary Large Object) trong CSDL
  - Quản lý các phiên làm việc sử dụng đối tượng Transaction
  - Truy vấn dữ liệu sử dựng LINQ (LINQ to SQL, LINQ to Objects, LINQ to MS ADO.NET, LINQ to XML)

1. Command Objects

- Đối tượng Command
  - Dùng để thực thi câu lệnh SQL hoặc các Stored Procedures
  - Command Objects bao gồm các thông tin cần thiết để thực thi các câu lệnh SQL, Stored Procedure, Functions…; lấy dữ liệu từ nguồn dữ liệu; trả dữ liệu về ứng dụng; thực thi các thao tác CSDL; thay đổi và xóa các đối tượng của CSDL
  - Cũng giống như đối tượng Connection, đối tượng Command sử dụng cũng tùy thuộc với .NET Framework Data Providers được dùng để kết nối với nguồn dữ liệu. Ví dụ nếu dùng .NET Framework Data Providers cho SQL thì phải sử dụng đối tượng SqlCommand

1. Command Objects

- Đối tượng Command
  - Một số thuộc tính chung của đối tượng Command

NAME	DESCRIPTION
CommandText	Thiết lập với mọi câu lệnh SQL hoặc Stored Procedure
CommandTimeout	Khoảng thời gian (giây) trước khi kết thúc quá trình cố gắng thực thi một câu lệnh
CommandType	= Text, thực hiện CommandText là 1 câu lệnh SQL = StoredProcedure, thực hiện stored procedure cụ thể xác định tại thuộc tính CommandText
Connection	Chỉ rõ đối tượng connection mà Command sử dụng

1. Command Objects

- Đối tượng Command
  - Một số thuộc tính chung của đối tượng Command

NAME	DESCRIPTION
Parameters	Tập hợp tham số của Command. Khi chạy các câu lệnh có tham số hoặc các stored procedure cần truyền tham số cho tập hợp này
Transaction	SqlTransaction bên trong SqlCommand thực thi

1. Command Objects

- Đối tượng Command
  - Một số phương thức chung của đối tượng Command

NAME	DESCRIPTION
Cancel	Cố gắng hủy bỏ việc thực thi 1 câu lệnh
ExecuteNonQuery	Thực thi các câu lệnh SQL hoặc các stored procedure không trả về dữ liệu (not return data)
ExecuteReader	Thực thi các câu lệnh mà kết quả trả về dưới dạng bảng (hoặc các dòng) dữ liệu
ExecuteXMLReader	Trả về dữ liệu định dạng XML. Trả về đối tượng System.Xml.XmlReader
ExecuteScalar	Thực thi các câu lệnh SQL hoặc các Stored Procedure trả về 1 giá trị đơn.

1. Command Objects

- Đối tượng Command
  - Một số sự kiện chung của đối tượng Command

NAME	DESCRIPTION
Disposed	Khi câu lệnh được hủy bỏ (disposed)
StatementCompleted	***(SqlCommand)*** Xảy ra khi 1 câu lệnh SQL hoàn thành

1. Command Objects

- Tạo đối tượng Command thực thi 1 câu lệnh SQL

```
// Create a Command object that executes a SQL statement
SqlCommand CustomersCommand = new SqlCommand();
CustomersCommand.Connection = NorthwindConnection;
CustomersCommand.CommandType = CommandType.Text;
CustomersCommand.CommandText = "SELECT CustomerID,
CompanyName FROM Customers";
```

1. Command Objects

- Tạo đối tượng Command thực thi 1 Stored Procedure

```
// Create a Command object that executes a SP
SqlCommand TopTenCommand = new SqlCommand();
TopTenCommand.Connection = NorthwindConnection;
TopTenCommand.CommandType =
CommandType.StoredProcedure;
TopTenCommand.CommandText = "Ten Most Expensive
Products";
```

1. Command Objects

- Tạo đối tượng Command thực thi DDL (Performs Catalog Operations)

```
// Create a Command object that executes Data Definition Language – DDL. Create a new table
SqlCommand CreateTableCommand = new SqlCommand();
CreateTableCommand.Connection = NorthwindConnection;
CreateTableCommand.CommandType = CommandType.Text;
CreateTableCommand.CommandText = "CREATE TABLE
SalesPersons (" + "[SalesPersonID] [int] IDENTITY(1,1) NOT
NULL, " + "[FirstName] [nvarchar](50) NULL, " +
"[LastName] [nvarchar](50) NULL)";
```

1. Command Objects

- Tạo đối tượng Command trả về 1 giá trị đơn

```
// Create a Command object that returns a Single Value
SqlCommand ExecuteScalarCommand = new SqlCommand();
ExecuteScalarCommand.Connection = NorthwindConnection;
ExecuteScalarCommand.CommandType = CommandType.Text;
ExecuteScalarCommand.CommandText = "SELECT Count(*) FROM
Customers";
// Open the connection and execute the command
ExecuteScalarCommand.Connection.Open();
int CustomerCount = (int)ExecuteScalarCommand.ExecuteScalar();
MessageBox.Show("There are " + CustomerCount.ToString() + "
customers");
ExecuteScalarCommand.Connection.Close();
```

1. Command Objects

- Tạo đối tượng Command trả về dữ liệu XML

```
// Create a Command object that returns XML Data
SqlCommand ExecuteXMLCommand = new SqlCommand();
ExecuteXMLCommand.Connection = NorthwindConnection;
ExecuteXMLCommand.CommandType = CommandType.Text;
// Add the For XML Auto clause to return the data as well-formed
XML
ExecuteXMLCommand.CommandText = "SELECT CustomerID
FROM Customers For XML Auto";
ExecuteXMLCommand.Connection.Open();
System.Xml.XmlReader reader =
ExecuteXMLCommand.ExecuteXmlReader();
// Add code here to iterate through the XMLReader;
reader.Close();
ExecuteXMLCommand.Connection.Close();
```

2. Parameters in SQL Commands

- Các tham số (Parameters)
  - Giống như là 1 loại biến dùng để truy cập và trả về dữ liệu giữa ứng dụng và CSDL
  - Kiểu dữ liệu của các tham số được định nghĩa trong tập hợp System.Data.SqlDbType
  - Tập hợp SqlDbType chứa 1 danh sách các kiểu hiện có trong SQL Server
  - Thường sử dụng ở mệnh đề WHERE của truy vấn SQL

2. Parameters in SQL Commands

- Các loại tham số (Types of Parameters)
  - Input Parameters: thường dùng để gửi dữ liệu vào CSDL
  - Output Parameters: nhận thông tin đi ra từ CSDL
  - InputOutput Parameters: vừa gửi và nhận dữ liệu khi thực thi 1 câu lệnh
  - Kiểu tham số được thiết đặt trong thuộc tính Direction của tham số và được gán giá trị từ tập hợp ParameterDirection.

2. Parameters in SQL Commands

- Tạo các tham số (Creating Parameters)
  - Bằng cách khai báo 1 thể hiện của lớp Parameter
  - Sau đó thiết lập tên và kiểu…

```
//Ví dụ tạo Input Parameter
SqlParameter TotalCostParameter = new SqlParameter();
TotalCostParameter.ParameterName = "@TotalCost";
TotalCostParameter.SqlDbType = SqlDbType.Money;
```

```
//Ví dụ tạo Output Parameter
SqlParameter TotalCostParameter = new
SqlParameter("@TotalCost", SqlDbType.Money);
TotalCostParameter.Direction = ParameterDirection.Output;
```

2. Parameters in SQL Commands

- Thêm các tham số vào đối tượng Command
  - Đối tượng Command có thuộc tính Parameters biểu diễn tập các tham số của đối tượng (ví dụ thuộc tính SQLCommand.Parameters)
  - Sau khi tạo 1 tham số, phải thêm nó vào tập hợp các tham số của đối tượng Command sẽ thực thi câu lệnh SQL hoặc stored procedure (và sử dụng các tham số)

```
// Adding Parameters to Command Objects
GetCostCommand.Parameters.Add(TotalCostParameter);
```

3. Transaction Object

- Transaction
  - Là các câu lệnh được thực thi như là 1 group với giả thiết là nếu bất kỳ 1 câu lệnh nào trong group bị hỏng thì toàn bộ Transaction sẽ bị hủy bỏ (aborted) và mọi thay đổi đã được tạo ra bởi bất kỳ câu lệnh nào trong group đều có thể khôi phục lại (roll-back) như thể chưa có câu lệnh nào được thực hiện.
  - Transaction chủ yếu để duy trì (bảo dưỡng - maintain) tính toàn vẹn của dữ liệu trong CSDL

3. Transaction Object

- Tạo các Transaction
  - Sử dụng đối tượng Transaction
  - Gán cho nó kết quả trả về từ phương thức BeginTransaction của đối tượng kết nối

```
// Create Transaction
SqlTransaction transaction;
transaction = NorthwindConnection.BeginTransaction();
```

3. Transaction Object

- Thiết lập mức độ cách ly của Transaction
  - Để kiểm soát các tiến trình khác có thể truy cập dữ liệu khi transaction đang truy cập nó.
  - Sử dụng phương thức BeginTransaction như sau:

```
// Create Transaction
SqlTransaction transaction;
transaction =
NorthwindConnection.BeginTransaction(IsolationLevel.Snapshot);
```

4. LINQ

- LINQ Queries

Làm việc với dữ liệu trong môi trường không kết nối

1. Đối tượng DataSet

- DataSet Object
  - Nằm trong namespace System.Data
  - Được sử dụng như là 1 bộ nhớ đệm (lưu trữ tạm thời) của dữ liệu đang được dùng trong ứng dụng
- Có 2 loại đối tượng DataSet riêng biệt là type và untyped.
  - Untyped DataSet là thể hiện di truyền chuẩn của lớp DataSet nơi mà ta xây dựng định nghĩa DataSet (schema) bằng cách tạo các đối tượng DataTable (untyped DataTable object) và thêm chúng vào tập hợp Tables của DataSet.
  - Typed DataSet lấy nguồn giản đồ (schema) của nó từ một file .xsd và bao gồm tập các kiểu rõ ràng (ví dụ như đối t]ngj Customers Table cụ thể)

1. Đối tượng DataSet

- Có 3 cách phân biệt để tạo đối tượng DataSet
  - Khai báo mới 1 đối tượng DataSet trong phần soạn thảo code. Kết quả là 1 đối tượng rỗng (empty) đòi hỏi tạo DataTable và đối DataRelation (tùy chọn) được thêm vào DataSet
  - Sử dụng DataSet Designer và Data Source Configuration Wizard để tạo các đối tượng DataSet từng bước một hoặc tạo một kết nối dữ liệu và sau đó cho phép chọn đối tượng CSDL từ kết nối đó để xây dựng một typed DataSet và hầu hết (không phải tất cả) code cần thiết được sinh ra
  - Kéo đối tượng DataSet từ Toolbox vào trong form và sử dụng trình soạn thảo Table and Column Collection đẻ xây dựng giản đồ cho DataSet

1. Đối tượng DataSet

- Tạo đối tượng DataSet nhờ code
  - Khai báo thể hiện của DataSet
  - Có thể cung cấp tên của DataSet nếu muốn.
  - Ví dụ tạo mới 1 DataSet đặt tên là NorthwindDataSet

```
// Creating DataSet Objects Programmatically
DataSet NorthwindDataSet = new DataSet(NorthwindDataSet);
```

1. Đối tượng DataSet

- Tạo đối tượng DataSet nhờ code
  - Sau khi khai báo 1 DataSet mới, cần phải thêm đối tượng DataTable (đối tượng thực sự giữ dữ liệu trong ứng dụng)
  - Ví dụ minh họa cách thức thêm đối tượng DataTable (tên CustomersTable và OrderDataTable) vào 1 DataSet (NorthwindDataSet)

```
// Adding DataTable Objects to a DataSet
// Create some DataTables
DataTable CustomersTable = new DataTable ();
DataTable OrdersDataTable = new DataTable ();
// Add DataTables to the Dataset's Tables collection.
NorthwindDataset.Tables.Add(CustomersTable);
NorthwindDataset.Tables.Add(OrdersDataTable);
```

1. Đối tượng DataSet

- Tạo đối tượng DataSet nhờ code
  - Sau khi thêm các bảng vào DataSet, cần sử dụng đối tượng DataRelation để biểu diễn quan hệ giữa các DataTable (giống các table trong CSDL)
  - Khai báo các đối tượng DataRelation và cung cấp các cột từ bảng cha và bảng con. Sau đó phải thêm nó vào tập hợp Relations của DataSet

1. Đối tượng DataSet

- Tạo đối tượng DataSet nhờ code
  - Ví dụ minh họa cách thức tạo đối tượng DataRelation trong DataSet có tên NorthwindDataSet. Giả sử rằng các đối tượng DataTable là Customers và Orders đều có cột CustomerID được dùng để liên kết dữ liệu

```
// Adding a Relationship Between Tables in a DataSet
// Create the new relationship.
DataRelation CustomersOrders = new DataRelation
("CustomersOrders",
CustomersTable.Columns["CustomerID"],
OrdersTable.Columns["CustomerID"]);
// Add the relationship to the DataSet.
NorthwindDataset.Relations.Add(CustomersOrders);
```

1. Đối tượng DataSet

- Tạo đối tượng DataSet nhờ code
  - Để truy cập và các bản ghi có liên kết trong các đối tượng DataTable, đầu tiên phải chọn một DataRow từ một trong 2 bảng là bảng cha và bảng con và sau đó gọi một trong phương thức là GetParentRow hoặc GetChildRows của DataRow.
  - Gọi phương thức GetParentRow sẽ trả về một DataRow đơn lẻ biểu diễn bản ghi ở bảng cha
  - Gọi phương thức GetChildRows sẽ trả về một mảng các đối tượng DataRow biểu điên mọi dòng có liên quan đến đối tượng cha được chọn

1. Đối tượng DataSet

- Tạo đối tượng DataSet nhờ code
  - Trả về bản ghi cha (parent row) của một bản ghi con được chọn
  - Ví dụ trả về Customers của Orders được chọn:

```
// Returning the parent row of a selected child record
DataRow Customer =
SelectedOrdersRow.GetParentRow("FK_Orders_Customers");
```

1. Đối tượng DataSet

- Tạo đối tượng DataSet nhờ code
  - Trả về các bản ghi con liên đới của 1 bản ghi cha được chọn
  - Ví dụ trả về các bản ghi của Orders có liên quan đến Customers được chọn:

```
// Returning the related child rows of a selected parent row
DataRow Order() =
SelectedCustomersRow.GetChildRows("FK_Orders_Customers");
```

1. Đối tượng DataSet

- Tạo đối tượng DataSet nhờ code
  - Gộp các nội dung DataSet: ví dụ minh họa gộp nội dung của OldSalesDataSet với nội dung của SalesHistoryDataSet. Thuộc tính PreserveChanges thiết lập là True và mọi giản đồ khác đều bị bỏ qua

```
// Merging DataSet Contents
SalesHistoryDataSet.Merge(OldSalesDataSet, true,
MissingSchemaAction.Ignore);
```

1. Đối tượng DataSet

- Tạo đối tượng DataSet nhờ code
  - Sao chép nội dung DataSet: thường dùng khi muốn tạo 1 bản copy của dữ liệu và thực hiện 1 số xử lý mà không muốn thay đổi dữ liệu gốc
  - Để sao chép chỉ cần tạo 1 đối tượng DataSet và gán cho nó giá trị trả về từ phương thức DataSet.Copy

```
// Copying DataSet Contents
DataSet  CopyOfDataSet = new  DataSet();
CopyOfDataSet = OriginalDataSet.Copy();
```

1. Đối tượng DataSet

- Các phương thức chính của DataSet

NAME	DESCRIPTION
Tables.Add	Thêm 1 bảng vào tập hợp Tables của DataSet
Tables.AddRange	Thêm nhiều bảng vào tập hợp Tables của DataSet
Tables.Remove	Xóa DataTable khỏi tập hợp Tables của DataSet
Tables.RemoveAt	Xóa DataTable xác định theo chỉ số khỏi tập hợp Tables của DataSet
Tables.Clear	Xóa tất cả các bảng khỏi DataSet
Tables.Contains	Trả về True nêu trong Tables có DataTable cần kiểm tra, ngược lại là không có

1. Đối tượng DataSet

- Các phương thức chính của DataSet

NAME	DESCRIPTION
Tables.IndexOf	Trả về chỉ số của DataTable
Tables.Count	Lấy số lượng bảng chứa trong DataSet
HasChanges	Kiểm tra sự thay đổi của tất cả các dòng dữ liệu trên bảng (thêm, sửa, xóa) và trả về True nếu có
HasChanges (<trạng thái dòng>)	Kiểm tra sự thay đổi của tất cả các dòng dữ liệu trên bảng có trạng thái như <trạng thái dòng> và trả về True nếu có
GetChanges	Trả về bản sao của DataSet gồm những dòng dữ liệu đã bị thay đổi trên các bảng (do thêm, sửa, xóa)

1. Đối tượng DataSet

- Các phương thức chính của DataSet

NAME	DESCRIPTION
GetChanges (<trạng thái dòng>)	Trả về bản sao của DataSet gồm những dòng dữ liệu đã bị thay đổi có trạng thái như <trạng thái dòng>
AcceptChanges	Cập nhật các thay đổi kể từ lúc lấy dữ liệu về hoặc từ lần gọi AcceptChanges trước. Trên DataTable, DataRow cũng có phương thức tương ứng. Khi gọi AcceptChanges của DataSet sẽ kéo theo gọi AcceptChanges của DataTable rồi DataRow
RejectChanges	Phục hồi tất cả các thay đổi kể từ lúc lấy dữ liệu về hoặc từ lần gọi AcceptChanges trước. Khi gọi RejectChanges của DataSet sẽ kéo theo gọi RejectChanges của DataTable rồi DataRow

2. Đối tượng DataTable

- DataTable
  - Là đối tượng của ADO.NET nằm trong namespace System.Data
  - Cung cấp vùng lưu trữ trong bộ nhớ (in-memory storage) cho dữ liệu của ứng dụng tương tự như các bảng trong CSDL
  - Có thể tạo DataTable và thêm vào DataSet và liên kết bởi DataRelation hoặc có thể sử dụng DataTable một cách độc lập với đối tượng DataSet

2. Đối tượng DataTable

- Tạo đối tượng DataTable và thêm vào DataSet (xem lại phần trước)
- Tạo một cột biểu thức (Expression Columns) trong đối tượng DataTable
  - Để chứa kết quả của các phép tính giữa các cột đã tồn tại như là cột thêm vào
  - Ví dụ minh họa cách tạo 1 cột biểu thức và thêm vào bảng NorthwindDataSet.Order_Details.

2. Đối tượng DataTable

- Tạo một cột biểu thức (Expression Columns) trong đối tượng DataTable
  - Ví dụ minh họa cách tạo 1 cột biểu thức và thêm vào bảng NorthwindDataSet.Order_Details.

```
// Create a new DataColumn and set its name and data type.
DataColumn TotalPriceColumn = new DataColumn
("TotalPrice", System.Type.GetType("System.Double"));
// Set the column's Expression property to the desired expression,
// in this case the UnitPrice x Quantity, which is the total price of this
record.
TotalPriceColumn.Expression = ("UnitPrice * Quantity");
// Now add the column to the DataTable
northwindDataSet.Order_Details.Columns.Add(TotalPriceColumn);
```

2. Đối tượng DataTable

- Tạo một cột tự tăng (Autoincrementing Columns) trong đối tượng DataTable
  - Thường tạo các cột không trùng giá trị (cột khóa – unique column) nhưng giá trị thực sự của cột lại không quá quan trọng so với tính chất duy nhất của nó
  - Ví dụ tạo cột SalesOrderID bắt đầu từ giá trị 100, bước nhảy là 5.

2. Đối tượng DataTable

- Tạo một cột tự tăng (Autoincrementing Columns) trong đối tượng DataTable

```
// Create the SalesOrderID column and set its data type to an integer.
SalesTable.Columns.Add("SalesOrderID",
Type.GetType("System.Int32"));
// Setting the AutoIncrement property to true makes this an
autoincrement column!
SalesTable.Columns["SalesOrderID"].AutoIncrement = true;
// Provide the starting value in the AutoIncrementSeed property.
SalesTable.Columns["SalesOrderID"].AutoIncrementSeed = 100;
// The amount added to the previous row's value is determined by the
AutoIncrementStep value.
SalesTable.Columns["SalesOrderID"].AutoIncrementStep = 5;
```

2. Đối tượng DataTable

- Thêm ràng buộc (Constraints) vào DataTable
  - Tạo một ràng buộc khóa ngoại (Foreign Key Constraint)

```
// Create a Foreign Key Constraint
ForeignKeyConstraint ForeignKey = new
ForeignKeyConstraint("FK_Orders_OrderDetails",
NorthwindDataset.Orders.Columns["OrderID"],
NorthwindDataset.Order_Details.Columns["OrderID"]);

NorthwindDataset.Orders.Constraints.Add(ForeignKey);
```

2. Đối tượng DataTable

- Thêm ràng buộc (Constraints) vào DataTable
  - Tạo một ràng buộc khóa "duy nhất" (Unique Constraint)

```
// Create a Unique Constraint
UniqueConstraint unique = new
UniqueConstraint(NorthwindDataSet.Orders.OrderIDColumn);

northwindDataSet.Orders.Constraints.Add(unique);
```

2. Đối tượng DataTable

- Thêm dữ liệu vào một DataTable

```
// Adding Data to a DataTable
NorthwindDataset.CustomersRow NewRow =
(NorthwindDataset.CustomersRow)NorthwindDataset1.Cus
tomers.NewRow;
NewRow.CustomerID = "WINGT";
NewRow.CompanyName = "Wingtip Toys";
NewRow.ContactName = "Steve Lasker";
…
NorthwindDataset1.Customers.Rows.Add(NewRow);
```

2. Đối tượng DataTable

- Sửa dữ liệu trong một DataTable
  - Ví dụ thay đổi cột CompanyName của SelectedRow bởi giá trị mới là “Contoso”

```
// Editing Data to a DataTable
SelectedRow ["CompanyName"] = "Contoso";
```

3. Đối tượng DataColumn

- Các thuộc tính chính của DataColumn
  - System.Data.DataColumn
  - Cấu trúc của bảng là tập hợp các DataColumn. Có thể tạo cấu trúc bảng từ cấu trúc bảng trong nguồn dữ liệu hoặc tạo lập từ các DataColumn

NAME	DESCRIPTION
AllowDBNull	Thuộc tính cho biết cột có chấp nhận giá trị Null không (Đọc/Ghi)
AutoIncrement	Thuộc tính cho biết giá trị cột có tự động tăng khi thêm dòng mới không (Đọc/Ghi)
AutoIncrementSeed	Giá trị bắt đầu cho cột khi thêm dòng đầu tiên nếu AutoIncrement = True
AutoIncrementStep	Bước tăng cho dòng thêm mới kế tiếp nếu AutoIncrement = True

3. Đối tượng DataColumn

- Các thuộc tính chính của DataColumn

NAME	DESCRIPTION
Caption	Tiêu đề của cột
ColumnName	Tên cột
DataType	Kiểu dữ liệu của cột
DefaultValue	Giá trị mặc định cho cột khi thêm 1 dòng mới
Expression	Biểu thức dùng để tính giá trị cho cột

3. Đối tượng DataColumn

- Các thuộc tính chính của DataColumn

NAME	DESCRIPTION
MaxLength	Độ rộng tối đa cho cột kiểu chuỗi
Ordinal	Trả về số thứ tự của cột trong tập hợp DataColumn
ReadOnly	Giá trị cho biết cột có được phép sửa đổi sau khi dòng mới được thêm vào
Table	Trả về DataTable chứa cột
Unique	Giá trị cho biết giá trị cột của mỗi dòng có phải là duy nhất

4. Đối tượng DataRow

- Các thuộc tính chính của DataRow
  - System.Data.DataRow
  - Dữ liệu lưu trữ trong DataTable qua các DataRow

NAME	DESCRIPTION
HasErrors	Thuộc tính cho biết dòng có đang bị lỗi hay không
RowError	Mô tả nội dung lỗi của dòng nếu đang bị lỗi
ItemArray	Tất cả nội dung của DataRow dưới dạng 1 mảng
Table	Trả về tên bảng chứa DataRow
RowState	Trả về tình trạng của DataRow.

4. Đối tượng DataRow

- Các thuộc tính chính của DataRow

NAME	DESCRIPTION
Item	Nội dung dữ liệu lưu giữ của cột trên dòng có *tên*, *cột*, *số thứ tự cột* truyền vào như sau: <tên DataRow>.Item (<tên>) <tên DataRow>.Item (<cột>) <tên DataRow>.Item (<chỉ số>) Nội dung dữ liệu lưu trữ của cột trên dòng có *tên*, *cột*, *số thứ tự cột* truyền vào, giá trị trả về theo phiên bản truyền vào như sau: <tên DataRow>.Item (<tên>, <phiên bản>) <tên DataRow>.Item (<cột>, <phiên bản>) <tên DataRow>.Item (<chỉ số>, <phiên bản>) Phiên bản có các giá trị sau: Current, Original, Proposed

4. Đối tượng DataRow

- Các thuộc tính chính của DataRow
  - Các giá trị của thuộc tính RowState:

NAME	DESCRIPTION
Unchanged	Không có thay đổi nào được tạo ra kể từ lần cuối AcceptChanges được gọi hoặc kể từ khi khởi tạo DataTable (mặc định)
Added	Dòng này đã được thêm vào từ lần cuối AcceptChanges được gọi nhưng chưa được cập nhật
Modified	Dòng này đã được sửa đổi từ lần cuối AcceptChanges được gọi nhưng chưa được cập nhật
Deleted	Dòng này đã được đánh dấu xóa từ lần cuối AcceptChanges được gọi bằng phương thức Delete
Detached	Dòng này chưa bao giờ được thêm vào bất kỳ tập hợp DataTable.Rows nào

4. Đối tượng DataRow

- Chấp nhận hay hủy bỏ những thay đổi trên DataRow trong DataTable
  - Nếu muốn tất cả thay đổi của 1 dòng là có giá trị, gọi phương thức DataRow.AcceptChanges
  - AcceptChanges sẽ thiết lập trạng thái RowState về Unchanges và chuyển mọi giá trị hiện tại thành Original
  - Có thể gọi AcceptChanges trong DataRow, DataTable hoặc toàn bộ DataSet
  - Nếu bỏ qua mọi thay đổi thay vì chấp nhận chúng, gọi phương thức RejectChanges.

4. Đối tượng DataRow

- Các phương thức chính của DataRow

NAME	DESCRIPTION
BeginEdit	Bắt đầu chỉnh sửa dòng. Trong chế độ này, mọi sự kiện tạm thời bị ngưng lại, các kiểm tra tạm thời bỏ qua. Khi người dùng bắt đầu thay đổi nội dung trên các điều khiển liên kết, phương thức này được **ngầm** gọi. Ở chế độ này, và khi EndEdit chưa được gọi, dòng sẽ mạng trị các phiên bản Original (giá trị gốc) và Proposed (giá trị mới đề nghị). Có thể truy xuất các trị này thông qua thuộc tính Item (<cột>, <phiên bản>)
CancelEdit	Hủy bỏ việc chỉnh sửa trên dòng
RejectChanges	Hủy bỏ các thay đổi từ lần cập nhật trước

4. Đối tượng DataRow

- Các phương thức chính của DataRow

NAME	DESCRIPTION
Delete	Đánh dấu hủy dòng, khi phương thức được gọi, dòng có RowState là Deleted
EndEdit	Chấm dứt việc chỉnh sửa
GetChildRows	Trả về các dòng có quan hệ nhánh con với dòng đang tham chiếu theo các cú pháp: GetChildRows(<Đối tượng DataRelation>) GetChildRows(<tên DataRelation>) GetChildRows(<Đối tượng DataRelation>, <phiên bản>) GetChildRows(<tên DataRelation>, <phiên bản>)

4. Đối tượng DataRow

- Các phương thức chính của DataRow

NAME	DESCRIPTION
GetParentRow	Trả về dòng có quan hệ nhánh cha với dòng đang tham chiếu theo các cú pháp: GetParentRow(<Đối tượng DataRelation>) GetParentRow(<tên DataRelation>) GetParentRow(<Đối tượng DataRelation>, <phiên bản>) GetParentRow(<tên DataRelation>, <phiên bản>)
IsNull	Cho biết trị của cột truyền vào có mang trị Null IsNull (<DataColumn>) IsNull (<chỉ số của DataColumn>) IsNull (<tên DataColumn>) IsNull (<DataColumn>, <phiên bản>)

5. Đối tượng Constraint

- Constraint (ràng buộc) là một quy tắc áp dụng cho một hoặc nhiều cột để bảo đảm tính toàn vẹn dữ liệu trên DataTable
- System.Data.Constraint
- Có 2 loại constraint là
  - System.Data.ForeignKeyConstraint: ràng buộc khóa ngoại
  - System.Data.UniqueConstraint: ràng buộc duy nhất
- Những ràng buộc có thể do chúng ta tạo ra hoặc do thiết lập quan hệ giữa các bảng trong DataSet mà có

5. Đối tượng Constraint

- Các phương thức chính của ForeignKeyConstraint

NAME	DESCRIPTION
AcceptRejectRule	Quy định cách xử lý khi phương thức AcceptChanges trên DataTable cha xảy ra: - Cascade: cập nhật dây chuyền trên bảng con - None: không làm gì cả
Columns	Trả về các cột tham gia khóa ngoại trên DataTable con
RelatedColumns	Trả về các cột tham giá khóa ngoại trên DataTable cha
ConstraintName	Tên của ràng buộc
RelatedTable	Trả về DataTable cha
Table	Trả về DataTable con

5. Đối tượng Constraint

- Các phương thức chính của ForeignKeyConstraint

NAME	DESCRIPTION
DeleteRule	Quy định cách xử lý khi một DataRow trên DataTable cha bị xóa, có các giá trị: - Cascade: xóa các DataRow tương ứng trên DataTable con - None: không làm gì cả - SetDefault: gán giá trị mặc định của DataColumn cho các dòng trên DataTable con liên quan đến dòng vừa bị xóa trên bảng cha - SetNull: gán giá trị Null cho các dòng trên DataTable con ứng với dòng vừa bị xóa trên DataTable cha

5. Đối tượng Constraint

- Các phương thức chính của ForeignKeyConstraint

NAME	DESCRIPTION
UpdateRule	Quy định cách xử lý khi một DataRow trên DataTable cha bị thay đổi, có các giá trị: - Cascade: cập nhật các giá trị thay đổi trên DataTable cha vào các DataRow tương ứng trên DataTable con - None: không làm gì cả - SetDefault: gán giá trị mặc định của DataColumn cho các dòng trên DataTable con ứng với dòng vừa thay đổi trên bảng cha - SetNull: gán giá trị Null cho các dòng trên DataTable con ứng với dòng vừa thay đổi trên DataTable cha

5. Đối tượng Constraint

- Các phương thức chính của UniqueConstraint

NAME	DESCRIPTION
Columns	Trả về mảng các cột tham gia khóa duy nhất
ConstraintName	Tên ràng buộc khóa duy nhất
IsPrimaryKey	Khóa duy nhất có đồng thời là khóa chính không (True/False)
Table	Trả về tên bảng chứa ràng buộc

8. DataRelation

- DataSet quản lý các quan hệ giữa các DataTable thông qua tập hợp Relations. Đây là những đối tượng DataRelation chứa thông tin về mối quan hệ đã tạo ra trong DataSet
- Phân biệt: quan hệ (relation) và ràng buộc (constraint)
- System.Data.DataRelation

8. DataRelation

- Một số thuộc tính của DataRelation

NAME	DESCRIPTION
ChildColumns	Trả về các DataColumn trên DataTable con tham gia trong DataRelation
ChildKeyConstraint	Trả về ràng buộc khóa ngoại của DataRelation
ChildTable	Trả về DataTable con trong DataRelation
DataSet	Trả về DataSet chứa DataRelation
ParentColumns	Trả về các cột trên DataTable cha tham gia trong DataRelation

8. DataRelation

- Một số thuộc tính của DataRelation

NAME	DESCRIPTION
ParentKeyConstraint	Trả về ràng buộc duy nhất đảm bảo giá trị cột trên DataTable cha trong DataRelation có giá trị duy nhất
ParentTable	Trả về DataTable cha trong DataRelation
RelationName	Tên DataRelation (Đọc/Ghi)

9. Đối tượng DataAdapter

- DataAdapter
  - Là đối tượng cung cấp riêng biệt, mỗi 1 provider có 1 adapter riêng biệt
  - DataAdapter chứa thông tin (như là chuỗi kết nối và câu lệnh) cho phép ứng dụng có thể kết nối với CSDL và nạp dữ liệu vào đối tượng DataTable của DataSet; cho phép cập nhật dữ liệu trở lại CSDL sau khi thay đổi dữ liệu trong đối tượng DataTable.

9. Đối tượng DataAdapter

- Tạo đối tượng DataAdapter
  - Tạo DataAdapter thông qua UI, hoặc Data Adapter Configuration Wizard, hoặc viết code
  - Ví dụ minh họa cách tạo một DataAdapter mới bằng một câu lệnh SELECT khởi đầu và chỉ rõ đối tượng Connection

```
// Create the SqlDataAdapter
private SqlDataAdapter sqlDataAdapter1 = new SqlDataAdapter
("SELECT * FROM Shippers", NorthwindConnection);
```

9. Đối tượng DataAdapter

- DataAdapter Commands
  - Thông thường cấu hình đối tượng DataAdapter bằng một câu lệnh SELECT đơn giản để lấy dữ liệu và chuyển vào DataTable trong DataSet
  - Điều quan trọng là phải hiểu cách thức DataAdapter sử dụng câu lệnh SELECT đó như là nền tảng cho việc tự động sinh các câu lệnh INSERT, UPDATE, DELETE, những câu lệnh cần thiết để lưu các thay đổi được tạo ra trong DataSet trở lại CSDL khi gọi phương thức Update của DataAdapter
  - Khi tạo các DataAdapter trong code, nếu muốn có thể tạo thủ công các command, hoặc khi sử dụng các truy vấn phức tạp, có thể cần cấu hình thủ công các command dùng để update

9. Đối tượng DataAdapter

- DataAdapter Commands
  - Có 2 cách để cấu hình command cho 1 DataAdapter:
    - Tạo đối tượng Command và gán chúng cho thuộc tính DataAdapter tương ứng
    - Hoặc sử dụng đối tượng CommandBuilder
  - Tạo đối tượng Command cho 1 DataAdapter: có thể tạo các câu lệnh SELECT, INSERT, UPDATE và DELETE cho DataAdapter bằng việc code riêng lẻ các câu lệnh hợp lệ cho CSDL cá nhân và gán chúng cho thuộc tính DataAdapter command tương ứng.

9. Đối tượng DataAdapter

- DataAdapter Command
  - Ví dụ minh họa cách tạo các câu lệnh riêng lẻ cho DataAdapter có tên SqlDataAdapter1, giả sử rằng đã có đối tượng NorthwindConnection

```
// Create the SqlDataAdapter
private SqlDataAdapter  sqlDataAdapter1 = new
SqlDataAdapter ("SELECT * FROM Shippers",
NorthwindConnection);
SqlCommand InsertCommand = new SqlCommand("Valid
SQL INSERT statement", NorthWindConnection);
// configure any necessary parameters for your command
SqlCommand UpdateCommand = new SqlCommand("Valid
SQL UPDATEstatement", NorthWindConnection);
```

9. Đối tượng DataAdapter

- DataAdapter Command
  - Ví dụ minh họa cách tạo các câu lệnh riêng lẻ cho DataAdapter có tên SqlDataAdapter1, giả sử rằng đã có đối tượng NorthwindConnection

```
// configure any necessary parameters for your command
SqlCommand DeleteCommand = new SqlCommand("Valid
SQL DELETE statement", NorthWindConnection);
// configure any necessary parameters for your command
// Add the commands to the DataAdapter
SqlDataAdapter1.InsertCommand = InsertCommand;
SqlDataAdapter1.UpdateCommand = UpdateCommand;
SqlDataAdapter1.DeleteCommand = DeleteCommand;
```

9. Đối tượng DataAdapter

- DataAdapter Command
  - Sử dụng CommandBuilder: nếu DataAdapter sử dụng 1 câu lệnh SELECT lấy dữ liệu từ 1 bảng đơn, thì có thể sử dụng CommandBuiler để tự động sinh ra các câu lệnh INSERT, UPDATE, DELETE cho adapter. Một yêu cầu khác khi sử dụng CommandBuilder là câu lệnh SELECT phải trả về ít nhất 1 cột khóa chính hoặc một cột unique từ bảng.
  - Ví dụ minh họa sinh command cho 1 DataAdapter có tên SqlDataAdapter1 sử dụng CommandBuilder

9. Đối tượng DataAdapter

- DataAdapter Command

```
// Instantiate a DataAdapter with a valid SELECT statement
SqlDataAdapter SqlDataAdapter1 = new SqlDataAdapter(
"Valid Single Table SELECT Statement");
// Instantiate a CommandBuilder for SqlDataAdapter1
SqlCommandBuilder commands = new
SqlCommandBuilder(SqlDataAdapter1);
```

11. Đối tượng DataView

- Đối tượng System.Data.DataView cung cấp 1 cách làm việc với đối tượng DataTable, có thể được hiển thị trong các điều khiển ràng buộc dữ liệu như DataGridView
- DataView cho phép sắp xếp hay đặt lọc, cho phép sửa đổi dữ liệu gắn liền với DataTable

11. Đối tượng DataView

- Tạo đối tượng DataView
  - Có thể tạo mới 1 DataView hoặc tham chiếu đến 1 DataView đã tồn tại
  - Đối tượng DataTable thực sự có 1 thuộc tính DefaultView chứa DataView mặc định của bảng
  - Tham chiếu DataView đã tồn tại bằng cách gán 1 thể hiện của 1 DataView bởi thuộc tinh DataTable.DefaultView
  - DataView có nhiều ưu điểm cho phép ràng buộc nhiều điều bởi cùng nguồn dữ liệu và hiển thị các bản ghi khác nhau hoặc các thứ tự sắp xếp khác nhau

11. Đối tượng DataView

- Tạo đối tượng DataView
  - Ví dụ minh họa cách tạo đối tượng DataView:

```
// Create a new DataView
DataView CustomersDataView = new DataView
(NorthwindDataSet.Customers);
// Create a reference to the DataTable's default DataView
DataView CustomersDataView =
Northwind.Customers.DefaultView
;
```

11. Đối tượng DataView

- Tạo đối tượng DataView
  - Ví dụ minh họa việc tạo 1 DataView có ràng buộc giữa 2 view Orders và Customers

```
// Navigating Related Data in a DataView
//Create a DataView made up of orders for a selected customer
OrdersDataView =CustomersDataRowView.
        CreateChildView( "FK_Orders_Customers";
OrdersDataGridView.DataSource = OrdersDataView;
```

11. Đối tượng DataView

- Sắp xếp và đặt lọc dữ liệu dùng DataView
  - Sắp xếp bằng cách thiết lập thuộc tính DataView.Sort là tên cột muốn sắp xếp. Nếu sắp xếp trên nhiều cột, phân cách tên các cột bởi dấu phẩy.
  - Kết thúc bởi ASC để sắp xếp tăng dần (mặc định), DESC để sắp giảm dần
  - Ví dụ minh họa cách sắp xếp cột ContactName tăng dần trong DataView:

```
// Sorting Data Using a DataView
CustomersDataView.Sort = "ContactName DESC";
```

11. Đối tượng DataView

- Xem dữ liệu sử dụng DataView
  - Thông thường các đối tượng DataView sẽ ràng buộc với các điều khiển (ví dụ như DataGridView). Hoặc ràng buộc mỗi cột trong DataRowView với các điều khiển riêng lẻ (ví dụ như TextBox)
  - Chú ý rằng DataView chứa 1 tập các đối tượng DataRowView biểu diễn các dòng trong DataTable có liên quan
  - Mỗi DataRowView chứa 1 mảng biểu diễn các cột trong dòng
  - Để truy cập vào các giá trị cụ thể trong mỗi cột, duyệt qua các đối tượng DataRowView và truy cập các cột qua chỉ số hoặc tên cột

11. Đối tượng DataView

- Xem dữ liệu sử dụng DataView
  - Ví dụ minh họa cách gán các giá trị của cột thứ nhất (First Name) và thứ 2 (Last Name) vào biến RowValues

```
// Access column values by passing the column name to access the DataRowView column
string FullName = DataRowView("First Name").ToString() + " " + DataRowView("Last Name").ToString();
// Access column values by passing the column index to access the DataRowView column
string FullName = DataRowView(0).ToString() + " " + DataRowView(1).ToString();
```

11. Đối tượng DataView

- Sửa đổi dữ liệu trong DataView
  - Ví dụ minh họa cách gán giá trị Steve cho cột FirstName của DataRowView đã chọn

```
// Edit column values by passing the column name to access the DataRowView column and assigning the new value
DataRowView("First Name") = "Steve";
// Edit column values by passing the column index to access the DataRowView column and assigning the new value
DataRowView(0) = "Steve";
```

11. Đối tượng DataView

- Tìm kiếm dữ liệu trong DataView
  - Tìm kiếm các bản ghi trong DataView sử dụng phương thức Find và FindRows. Hai phương thức đều tìm kiếm
  - thức Find và FindRows
  - Ví dụ minh họa việc thiết lập khóa sắp xếp là CustomerID và sau đó gọi phương thức Find để tìm mã khách hàng "ALFKI"

```
CustomersDataView.Sort = "CustomerID";
Int FoundRow;
FoundRow = CustomersDataView.Find("ALFKI");
string s =
CustomersDataView.Item[FoundRow]["CompanyName"].
ToString();
```

11. Đối tượng DataView

- DataViewManager
  - Là tập hợp cơ bản của các đối tượng DataViewSetting được dùng để thiết lập cách thức sắp xếp và đặt lọc mặc định cho mỗi DataTable trong một DataSet.
  - Có thể dùng DataViewManager như 1 nguồn dữ liệu cho các điều khiển ràng buộc dữ liệu. Ví dụ nếu tạo 1 DataViewManager

```
// Setting the DataTable Objects Default TableViews Using a
DataViewManager
DataViewManager dvm = new DataViewManager
(NorthwindDataSet1);
OrdersDataGridView.DataSource =
dvm.DataViewSettings("Customers").Table;
```

11. Đối tượng DataView

- DataRowView
  - Tương tự DataRow trên DataTable
  - Là một cách hiển thị của DataRow theo phiên bản* trên DataView
  - Khi liên kết với 1 điều khiển dữ liệu, chỉ có 1 phiên bản của DataRowView được hiển thị. Các phiên bản đó là Default, Original, Current, Proposed như của DataRow
  - Chỉ có thể tạo DataRowView bằng phương thức AddNew của DataView

Điều khiển ràng buộc dữ liệu *Data-Bound Controls*

Data-Bound Forms

- Tạo một Form ràng buộc dữ liệu với Data Sources Wizard (Bài 10)

Data-Bound Controls

- Điều khiển ràng buộc dữ liệu (Data-Bound Controls hoặc Data Binding Controls): là các điều khiển của Windows Form hiển thị dữ liệu từ CSDL…
- Phân loại:
  - Điều khiển ràng buộc dữ liệu đơn giản (Simple Data Binding Controls) là các điều khiển hiển thị một phần tử đơn lẻ của dữ liệu. Ví dụ TextBox hiển thị giá trị của 1 cột trong 1 bảng
  - Điều khiển ràng buộc dữ liệu phức tạp (Complex Data Binding Controls) là các điều khiển có nhiều hơn 1 nguồn dữ liệu. Ví dụ ComboBox, DataGrid…

Data-Bound Controls

- Thuộc tính liên kết dữ liệu của điều khiển

Tên điều khiển	Thuộc tính liên kết DL
Label	Text
TextBox	Text
CheckBox	Checked
RadioButton	Checked
ComboBox	SelectedValue
ListBox	SelectedValue
CheckListBox	SelectedValue
DateTimePicker	Value
NumericUpDown	Value

Data-Bound Controls

- Ví dụ cách liên kết cột ProductName từ 1 DataTable vào 1 TextBox có tên TextBox1:

```
// Binding TextBox
TextBox1.DataBindings.Add("Text", productsBindingSource,
"ProductName");
```

- Ví dụ cách liên kết 1 DataGridView vào bảng Northwind Customers sử dụng 1 dataset

```
// Binding DataGridView
DataGridView1.DataSource = northwindDataSet1;
DataGridView1.DataMember = "Customers";
```

Data-Bound Controls

- Ví dụ liên kết 1 DataGridView vào bảng Northwind Customers sử dụng thành phần BindingSource:

```
// Binding DataGridView
BindingSource customersBindingSource = new
BindingSource (northwindDataSet1.“Customers”);
```

- Thành phần BindingSource chứa thông tin mà điều khiển cần để liên kết bằng cách tham chiếu đến 1 DataTable trong 1 DataSet. Bằng việc liên kết vào BindingSource thay vì DataSet, có thể dễ dàng gửi lại ứng dụng tới nguồn dữ liệu khác mà không cần phải gửi lại toàn bộ mã liên kết dữ liệu chỉ rõ nguồn dữ liệu mới!!!

1. ComboBox, ListBox, CheckListBox

- Khi các điều khiển này được dùng để hiển thị dữ liệu, cần quan tâm đến các thuộc tính sau:

NAME	DESCRIPTION
DataSource	Chỉ ra nguồn dữ liệu để lấy giá trị liệt kê chọn lựa
DisplayMember	Cho biết tên cột trên nguồn dữ liệu liệt kê sẽ được hiển thị trên điều khiển
ValueMember	Cho biết tên cột trên nguồn dữ liệu liệt kê sẽ được lấy giá trị để cập nhật vào nguồn dữ liệu khi chọn 1 dòng trên điều khiển
SelectedValue	Giá trị của dòng được chọn ứng với cột có tên là giá trị của ValueMember

1. ComboBox, ListBox, CheckListBox

- Một số sự kiện cần lưu ý:

NAME	DESCRIPTION
SelectedIndexChanged	Xảy ra khi thuộc tính SelectedIndex thay đổi (do lệnh và tương tác)
SelectedValueChanged	Xảy ra khi thuộc tính SelectedValue thay đổi (do lệnh và tương tác)
SelectedChangeCommitted	Xảy ra khi dòng chọn thay đổi và đã được cập nhật (do tương tác)

2. DataGridView

- Hiển thị DataSet trong DataGridView
  - Để hiển thị một DataSet (hay là DataTable) trong một DataGridView:
    - Thiết lập thuộc tính DataSource của DataGridView là DataSet
    - Thiết lập thuộc tính DataMember của DataGridView là tên của DataTable
- Cấu hình các cột của DataGridView
  - Có 6 kiểu cột có sẵn để sử dụng trong DataGridView
  - Khi thêm cột vào DataGridView chọn kiểu cột dựa trên dữ liệu định hiển thị

2. DataGridView

- Cấu hình các cột của DataGridView

NAME	DESCRIPTION
DataGridView-TextBoxColumn	Sử dụng để hiển thị văn bản và giá trị số.
DataGridView-CheckBoxColumn	Sử dụng để hiện thị giá trị Boolean
DataGridView-ImageColumn	Sử dụng để hiển thị hình ảnh (các đối tượng Image và Icon, hoặc byte array)
DataGridView-ButtonColumn	Sử dụng để cung cấp cho người dùng các điều khiển Button

2. DataGridView

- Cấu hình các cột của DataGridView

NAME	DESCRIPTION
DataGridView-ComboBoxColumn	Sử dụng kiểu này để hiển thị danh sách chọn (thường dùng để tham chiếu tới bảng khác)
DataGridView-LinkColumn	Sửu dụng để hiển thị link đến dữ liệu khác
Custom Column	Nếu các kiểu trên chưa phải là kiểu mong muốn thì có thể tạo một kiểu tùy chọn. Để tạo một kiểu tùy chọn (custom column), định nghĩa lớp kế thừ từ DataGridViewColumn hoặc bất kỳ lớp nào với lớp cơ sở của DataGridViewColumn.

2. DataGridView

- Thêm bảng và cột vào DataGridView
  - Trong Designer với hộp thoại Add Column
    - Chọn DataGridView trong form
    - Mở smart tag của DataGridView
    - Chọn Add Column
    - Trong hộp thoại Add Column, định nghĩa cột bằng cách thiết lập các giá trị thích hợp
  - Trong Code

```
DataGridViewTextBoxColumn ProductNameColumn = new
DataGridViewTextBoxColumn();
ProductNameColumn.Name = "ProductName";
ProductNameColumn.HeaderText = "Product Name";
ProductNameColumn.ValueType = System.Type.GetType("System.String");
DataGridView1.Columns.Add(ProductNameColumn);
```

2. DataGridView

- Xóa cột trong DataGridView
  - Trong Designer
    - Chọn DataGridView trong form
    - Mở smart tag của DataGridView
    - Chọn Edit Columns
    - Trong hộp thoại Edit Columns, chọn cột muốn xóa
    - Ấn nút Remove
  - Trong Code

```
DataGridView1.Columns.Remove("ProductName");
```

2. DataGridView

- Xác định Clicked Cell trong DataGridView
  - Sử dụng thuộc tính CurrentCell của DataGridView
  - Thuộc tính CurrentCell cho phép truy cập đến giá trị trong ô cũng như chỉ số dòng, cột.
  - Ví dụ:

```
// C#
String CurrentCellValue;
CurrentCellValue =
        CustomersDataGridView.CurrentCell.Value.ToString();
```

2. DataGridView

- Kiểm duyệt giá trị vào trong DataGridView
  - Sử dụng sự kiện CellValidating của DataGrid (được phát sinh khi một ô mất con trỏ)
  - Ví dụ kiểm duyệt cột ProductName có chứa xâu rỗng không?

```
// Sử dụng ví dụ cho DataGridView KHÔNG ràng buộc dữ liệu
if
(DataGridView1.Columns[e.ColumnIndex].Name=="ProductName")
{
        if (e.FormattedValue.ToString() == "")
        {DataGridView1.Rows[e.RowIndex].ErrorText="Product Name
                Is a required field";
        e.Cancel = true;}
        else
        {DataGridView1.Rows[e.RowIndex].ErrorText = "";}
}
```

2. DataGridView

- Kiểm duyệt giá trị vào trong DataGridView
  - Ví dụ kiểm duyệt cột ProductName có chứa xâu rỗng không?

```
// Sử dụng ví dụ cho DataGridView CÓ ràng buộc dữ liệu
if (DataGridView1.Columns[e.ColumnIndex].DataPropertyName
== "ProductName")
{
if (e.FormattedValue.ToString() == "")
        { DataGridView1.Rows[e.RowIndex].ErrorText = "Product
                Name is a required field";
        e.Cancel = true;}
else
{DataGridView1.Rows[e.RowIndex].ErrorText = "";}
}
```

2. DataGridView

- Định dạng DataGridView sử dụng Styles có sẵn
- Định dạng DataGridView sử dụng Custom Painting
  - Sử sụng sự kiện CellPainting
  - Chú ý khi xử lý sự kiện CellPainting, thì e.Handled = true.


```
// Paint the cell background color LightSkyBlue
e.Graphics.FillRectangle(Brushes.LightSkyBlue, e.CellBounds);
// Draw the contents of the cell
if (e.Value != null)
{
e.Graphics.DrawString(e.Value.ToString(), e.CellStyle.Font,
Brushes.Black, e.CellBounds.X, e.CellBounds.Y);
}
e.Handled = true;
```

2. DataGridView

- DataGridTableStyle
  - DataGrid có tập hợp TableStyles gồm những DataGridTableStyle giúp định dạng lưới theo một yêu cầu hiển thị cụ thể tùy theo bảng dữ liệu. DataGridTableStyle thuộc không gian tên System.Windows.Forms.DataGridTableStyle.

2. DataGridView

- TableStyles
  - Lưới có thể dùng để hiển thị dữ liệu của các nguồn khác nhau do đó có thể tạo nhiều DataGridTableStyle cho phù hợp với mỗi nguồn dữ liệu.
  - Các DataGridTableStyle của lưới được tập hợp lại thành TableStyles.
  - TableStyles là một tập hợp (collection) có các thuộc tính và phương thức đặc trưng của tập hợp như Count, Item, Add, AddRange, Clear, Contains, Remove, RemoveAt.

2. DataGridView

- DataGridColumnStyle
  - DataGridTableStyle quy định cách hiển thị của nguồn dữ liệu của lưới thì DataGridColumnStyle thuộc không gian tên System.Windows.Forms.DataGridColumnStyle. Đây là lớp phải kế thừa (MustInherit)
  - Có 2 lớp phát sinh từ lớp này là:
    - System.Windows.Forms.DataGridBoolColumn
    - System.Windows.Forms.DataGridTextBoxColumn

2. DataGridView

- GridColumnStyles
  - Là tập hợp các DataGridColumnStyle (cả DataGridBoolColumn lẫn DataGridTextBoxColumn) của một DataGridTableStyle.

Báo biểu & In ấn

Printing in Windows Forms

1. Crystal Report

- Crystal Report là phần mềm thiết kế báo biểu chuyên nghiệp được tích hợp trong các phiên bản của Visual Studio. Phiên bản Studio.NET được tích hợp Crystal Report 8.5
- Báo biểu: là một phần của project. Để tạo báo biểu vào thực đơn Project / Add New Item…
- Trong phần Templates, chọn mục Crystal Report để tạo mới một báo biểu
- Trong hộp thoại Add New Item nhập vào tên file báo biểu tại phần Name
- Nhấn nút Open, Visual Studio.NET sẽ tự động chuyển đổi sang giao diện của Crystal Report.

1. Crystal Report

- Có 3 hình thức tạo mới một báo biểu là:
  - Wizard (Using Report Expert)
  - Blank Report (Tự thiết kế từ đầu)
  - Mở một báo biểu đã tạo
- Chú ý khi chọn Blank Report khác với sử dụng phần mềm Crystal Report độc lập, VS.NET sẽ chuyển ngay sang màn hình thiết kế mà không chọn CSDL cho báo biểu
- Để chọn CSDL cho báo biểu, R-click vào mục Database Fields và chọn Add/Remove Database

1. Crystal Report

- Crystal Report chạy độc lập có một số thay đổi trong giao diện so với bản tích hợp sẵn. Ví dụ bản tích hợp cho phép tìm thấy TextObject trong phần ToolBox hay thuộc tính của các đối tượng thiết kế có thay đổi trực tiếp cửa sổ Properties như đang thiết kế form
- Không thể chuyển sang chế độ Preview để xem kết quả thiết kế báo biểu, thay vào đó cần phải sử dụng đối tượng Crystal Report Viewer của VS.NET

1. Crystal Report

- Nguồn dữ liệu
  - Crystal Report kết nối với nguồn dữ liệu thông qua các trình dữ liệu của nó. Mỗi trình được viết để xử lý cho một loại dữ liệu hoặc cho một kỹ thuật truy xuất dữ liệu cụ thể
  - Có 2 mô hình truy xuất là PULL và PUSH

1. Crystal Report

- Nguồn dữ liệu
  - Mô hình kéo (PULL Model)
    - Trình sẽ kết nối với nguồn dữ liệu và lấy về dữ liệu yêu cầu.
    - Kết nối với nguồn và lệnh SQL truy xuất dữ liệu đều do Crystal Report xử lý (ta không cần viết code)
    - Nếu không có dòng code nào viết cho lúc thực hiện, báo biểu sử dụng mô hình PULL

Mô hình Pull

1. Crystal Report

- Nguồn dữ liệu
  - Mô hình đẩy (PUSH Model)
    - Đòi hỏi phải viết lệnh kết nối dữ liệu, thực hiện truy xuất dữ liệu để tạo các DataSet chứa các field cần thiết cho báo biểu.
    - Cho phép chia sẻ kết nối trong ứng dụng và lọc dữ liệu trước khi đưa vào báo biểu

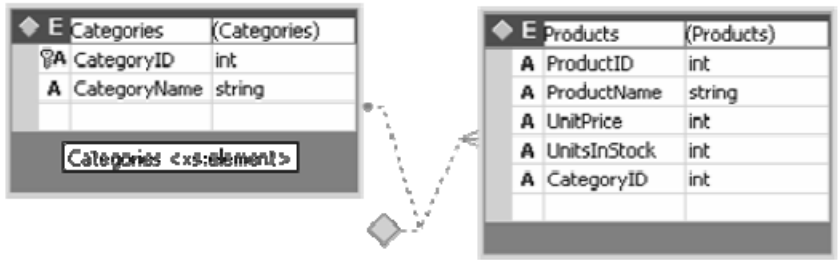
Mô hình Push

1. Crystal Report

- Sử dụng Crystal Report Viewer để hiển thị báo biểu
  - Để hiển thị báo biểu cần có điều khiển CrystalReportViewer.
  - Tạo một biến đối tượng có tên cv sử dụng CrystalDecisions.Windows.Forms.CrystalReportViewer.
  - Khi đưa (this.Add (cv)) vào form nhớ đặt thuộc tính Dock là Fill
  - Gán report sẽ hiển thị trên CrystalReportViewer vừa tạo là 1 thể hiện của CrystalReport1.rpt (đã tạo): cv.ReportSource = new CrystalReport1();
  - Hiển thị form bằng hàm ShowDialog()

1. Crystal Report

- Nguồn dữ liệu cho báo biểu từ DataSet (PUSH Model)
  - Cho phép tạo ra những báo biểu không cần kết nối với CSDL, thậm chí không cần có CSDL (từ file XML)
  - Sử dụng phương thức SetDataSource để gán một DataSet làm nguồn dữ liệu cho báo biểu
  - Ví dụ cách tạo:
    - Tạo mới DataSet: Vào Project chọn Add New Item, chọn DataSet, trong phần Name đặt tên là Products_Schema.xsd. Thiết kế DataSet như sau:



1. Crystal Report

- Nguồn dữ liệu cho báo biểu từ DataSet (PUSH Model)
  - Ví dụ cách tạo:
    - Lưu lại DataSet sau khi tạo xong cấu trúc bảng
    - Tạo mới một báo biểu, khi chọn nguồn dữ liệu cho báo biểu ta chọn mục ADO.NET DataSet, chọn DataSet vừa tạo, chọn 2 bảng Catagories và Products làm nguồn dữ liệu. Ấn OK để lưu lại nguồn dữ liệu của báo biểu
    - Thiết kế báo biểu, gợi ý đoạn lệnh hiển thị dữ liệu như sau…

1. Crystal Report

- Định lại dữ liệu cho báo biểu từ nguồn CSDL
  - Một số đối tượng trong CrystalDecisions.CrystalReports.Engine
    - Database: Nguồn dữ liệu sử dụng cho báo biểu
    - Table: Thông qua chỉ số, tên trả về bảng tương ứng được sử dụng trong báo biểu
    - Tables: tập hợp các bảng sử dụng trong báo biểu
  - Khi vị trí nguồn thay đổi cần định lại vị trí nguồn mới cho báo biểu thông qua thông tin đăng nhập của Table, thuộc tập hợp Tables của Database.

1. Crystal Report

- Định lại dữ liệu cho báo biểu từ nguồn CSDL
  - Thông tin đăng nhập thuộc lớp TableLogOnInfo trong không gian tên CrystalDecisions.Shared.TableLogOnInfo với các thông tin:
    - ConnectionInfo: Đối tượng chứa thông tin kết nối của bảng
    - DatabaseName: Tên tập tin CSDL
    - Password: Mật khẩu truy cập nguồn CSDL
    - ServerName: Tên server hoặc nguồn dữ liệu ODBC
    - UserID: Tên người dùng để truy cập CSDL
  - Thông tin TableLogOnInfo có tính chỉ đọc, khi thay đổi và muốn cập nhật lại phải sử dụng phương thức ApplyLogOnInfo của đối tượng Table với tham số là TableLogOnInfo đã thay đổi

1. Crystal Report

- Lọc dữ liệu báo biểu
  - Sử dụng thuộc tính: <CrystalReport>. RecordSelectionFormula = "{<Tên bảng> . <Tên field> } <Toán tử so sánh> <Giá trị>"
    - Toán tử so sánh: =, >=, <=, >, <, <>, In, Not In…
    - Giá trị: cần phải có ký hiệu thể hiện hằng dữ liệu
  - Hoặc sử dụng thuộc tính: <CrystalReportViewer>. <SelectionFormula> = "<Tên bảng> . <Tên field> } <Toán tử so sánh> <Giá trị>"

1. Crystal Report

- Truyền tham số cho báo biểu
  - Sử dụng phương thức: <CrystalReport>. SetParameterValue (<Tên tham số>, <Giá trị>)
    - <Tên tham số > là tên của tham số (ParameterField) đã tạo trong báo biểu
    - <Giá trị> không cần có ký hiệu của ký hiệu

1. Crystal Report

- Các loại kết xuất báo biểu
  - Kết xuất ra máy in: <CrystalReport>.PrintToPrinter (<số bản>, <Collate>, <từ trang>, <đến trang>)
  - Kết xuất ta file: <CrystalReport>.ExportToDisk (<Loại file>, <Tên file>)
    - <Loại file> có thể là: Excel, HTML40, NoFormat, PortableDocFormat, RichText, WordForWindows.

2. Printing

- Quản lý quá trình in ấn