



TỔNG QUAN .NET NGÔN NGỮ C#



Nội dung

1. Giới thiệu .NET Framework

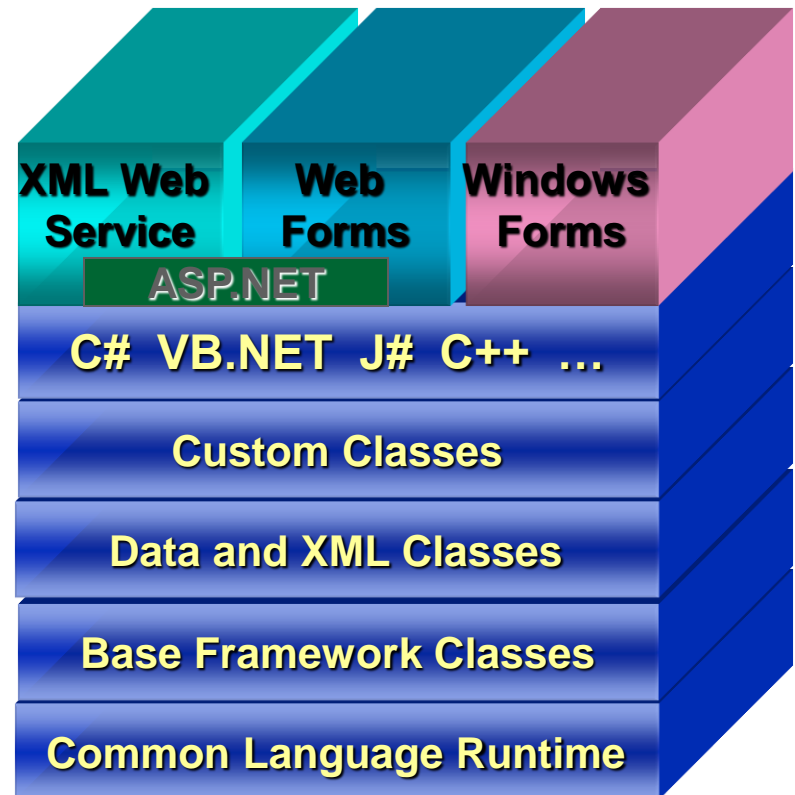
1. CLR, CTS, CLS, MSIL...
2. Garbage collection
3. Namespace

2. Tổng quan ngôn ngữ C#

1. Đặc điểm ngôn ngữ
2. Quá trình biên dịch CT C#
3. Các loại CT C#
4. Cấu trúc chương trình C#
5. Chương trình C# đơn giản

.NET Framework

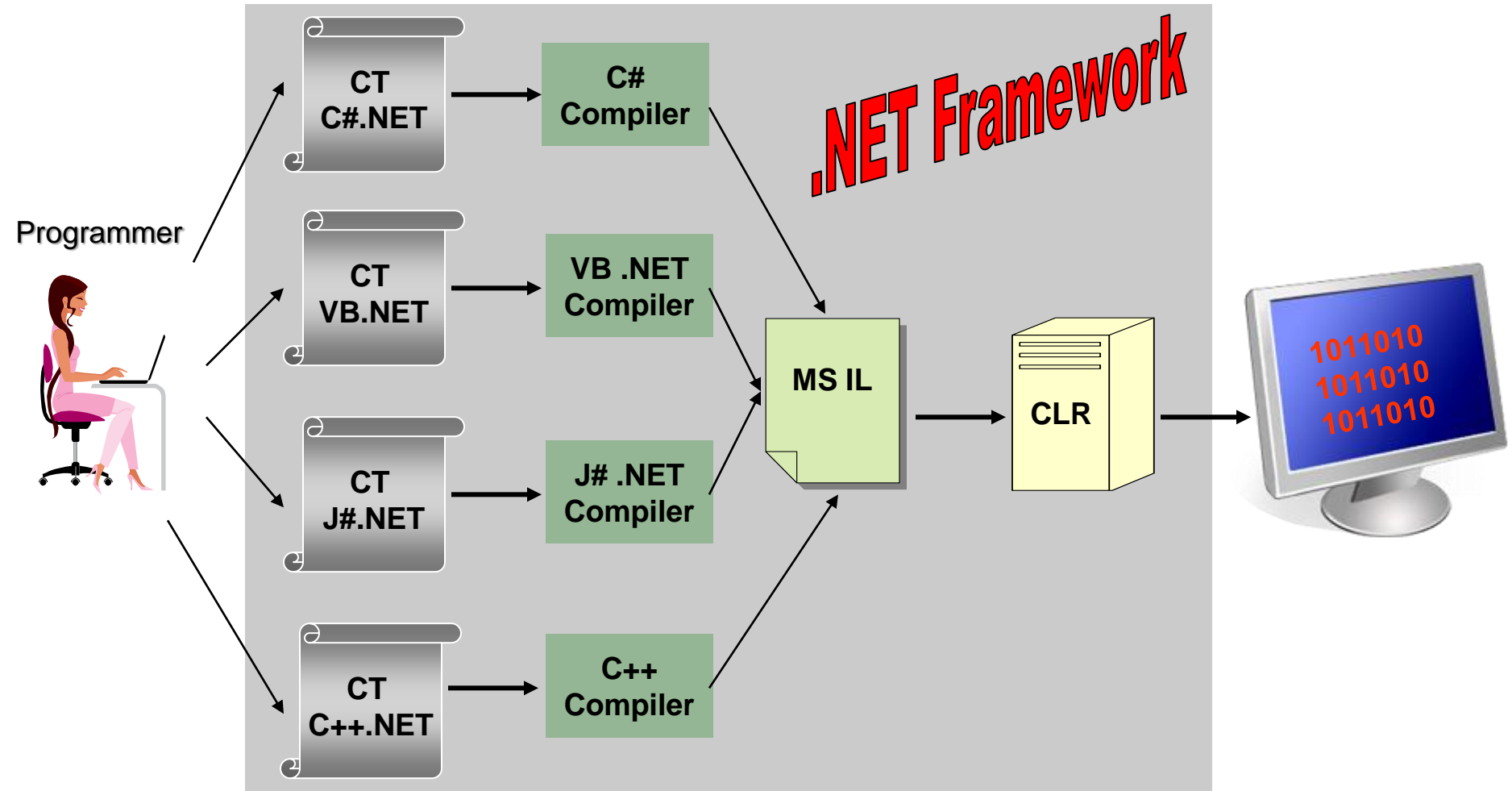
- Microsoft .NET gồm 2 phần chính : Framework và Integrated Development Environment (IDE). Framework cung cấp những gì cần thiết và căn bản. IDE thì cung cấp một môi trường giúp chúng ta triển khai dễ dàng, và nhanh chóng các ứng dụng dựa trên nền tảng .NET.
- Chương trình nền tảng cho công nghệ .NET
- Cung cấp tập hợp class library thường dùng
- Quản lý sự thực thi của các chương trình .NET



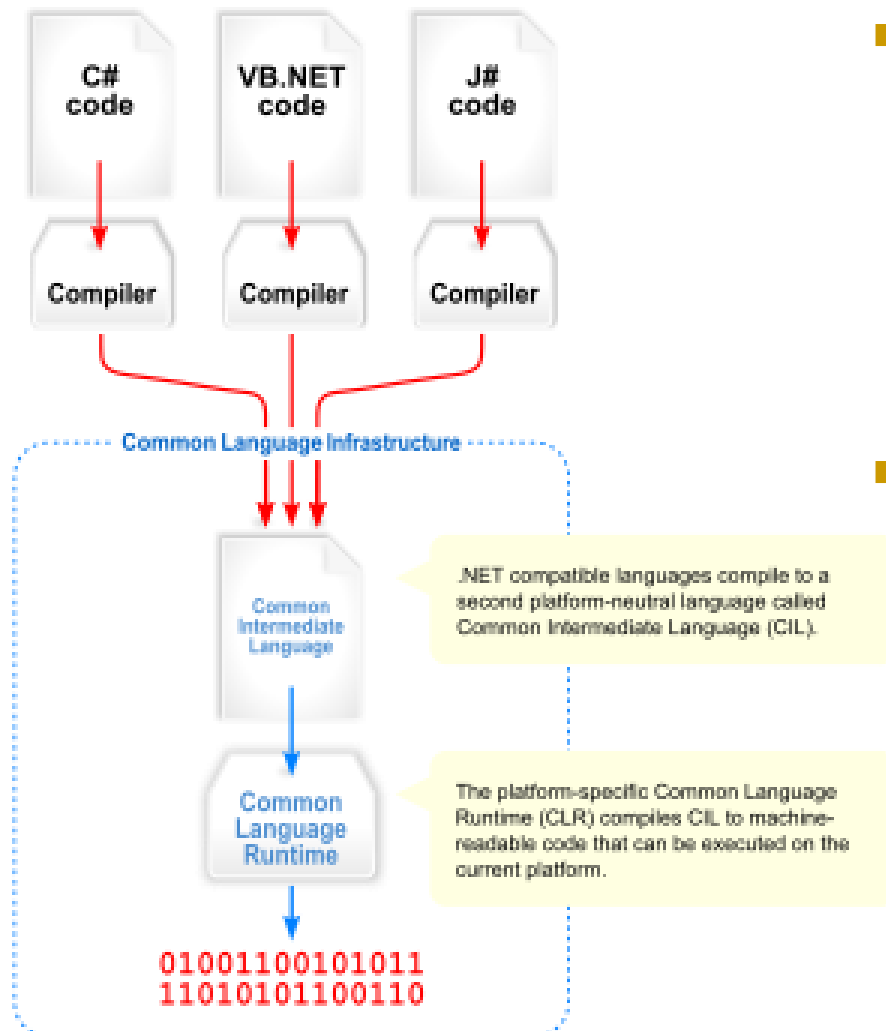
Đặc điểm của ứng dụng .NET

- Chạy trên nền (.NET framework)
- Mã nguồn được biên dịch qua MSIL(**MS Intermediate Language**)
- MSIL được thông dịch qua mã máy lúc thực thi nhờ vào CLR (Common Language RunTime)
- Độc lập nền tảng
 - Về lý thuyết có thể chạy trên mọi nền!
- Install .NET Framework redistribute packadge (dotnetfx.exe) để chạy ứng dụng .NET trên máy client.

Đặc điểm của ứng dụng .NET



.NET Framework - Architecture



- .NET Framework có hai thành phần chính: Common Language Runtime (CLR) và thư viện lớp .NET Framework. CLR là nền tảng của .NET Framework.
- **Common Infrastructure Language (CIL)**
 - provide a language-neutral platform for application development and execution

.NET Framework - CLR

- Theo quan điểm của người lập trình, .NET có thể hiểu như môi trường thực thi mới và thư viện lớp cơ sở cải tiến.
- Môi trường thực thi là: Common Language Runtime - CLR
- Vai trò chính CLR: locate, load, manage .NET types
- CLR còn quản lý những phần ở mức thấp như: memory management, security check
- Thư viện lớp .NET Framework là một tập hợp hướng đối tượng của các kiểu dữ liệu được dùng lại, nó cho phép chúng ta có thể phát triển những ứng dụng từ những ứng dụng truyền thống command-line hay những ứng dụng có giao diện đồ họa (GUI) đến những ứng dụng mới nhất được cung cấp bởi ASP.NET, như là Web Form và dịch vụ XML Web.

.NET Framework - CTS

- Common Type System (CTS):
 - CTS đảm bảo rằng những mã nguồn được quản lý thì được tự mô tả (selfdescribing).
 - Mã nguồn được quản lý có thể sử dụng những kiểu được quản lý khác và những thể hiện, trong khi thúc đẩy nghiêm ngặt việc sử dụng kiểu dữ liệu chính xác và an toàn.
 - Mục đích hỗ trợ thực thi chéo ngôn ngữ
 - Định nghĩa kiểu dữ liệu tiên định và có sẵn trong IL:
 - Tất cả ngôn ngữ .NET sẽ được sinh ra mã cuối trên cơ sở kiểu dữ liệu này



.NET Framework - CLS

- Common Language Specification:
 - Đảm bảo sự thực thi chéo
 - Tất cả compiler hướng .NET đều phải tuân thủ theo CLS
 - Có thể viết mã non-CLS nhưng sẽ ko đảm bảo thực thi chéo
 - IL phân biệt loại ký tự, VB.NET ko phân biệt, CLS báo rằng ko cho phép 2 định danh chỉ khác nhau về kiểu ký tự, do đó VB.NET có thể hoạt động trong CLS

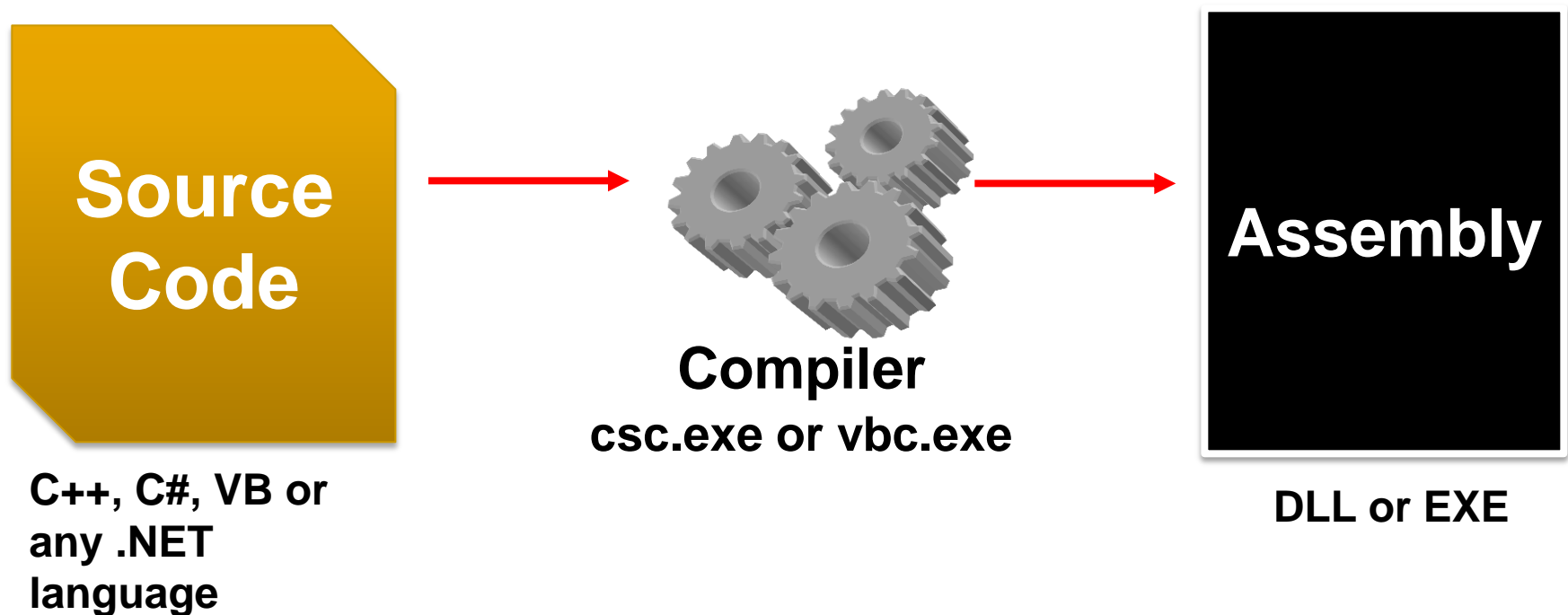


MS Intermediate Language

- Trong .NET Framework, chương trình không được biên dịch vào các tập tin thực thi mà thay vào đó chúng được biên dịch vào những tập tin trung gian gọi là Microsoft Intermediate Language (MSIL).
- IL
 - Abstracted assembly language
 - Ý tưởng tương tự mã Java bytecode
 - Mã cấp thấp cú pháp đơn giản \Rightarrow quá trình dịch sang mã máy nhanh hơn
- CLR chuyển IL thành mã máy lúc runtime
 - Sự chuyển này gọi là Just – In – Time Compilation hay JIT compiling

Common Language Runtime - compilation

- Mã nguồn C# được biên dịch vào MSIL khi chúng ta build project. Mã MSIL này được lưu vào trong một tập tin trên đĩa.
- Khi chúng ta chạy chương trình, thì MSIL được biên dịch một lần nữa, sử dụng trình biên dịch Just-In-Time (JIT). Kết quả là mã máy được thực thi bởi bộ xử lý của máy.



Assembly

- Assembly là tập hợp mã đã được biên dịch sang .NET.
- Assembly chứa nội dung thực thi chương trình hoặc các thư viện động.
- Assembly có thể được chứa trong nhiều file.
- Assembly cũng có thể chứa metadata dùng để mô tả các kiểu và phương thức được định nghĩa trong mã tương ứng.
- Assembly metadata được hiểu như là manifest, cho phép kiểm tra phiên bản và tính trạng của assembly.
- Portable Executable (PE)
 - Process assembly (EXE)
 - Library assembly (DLL)



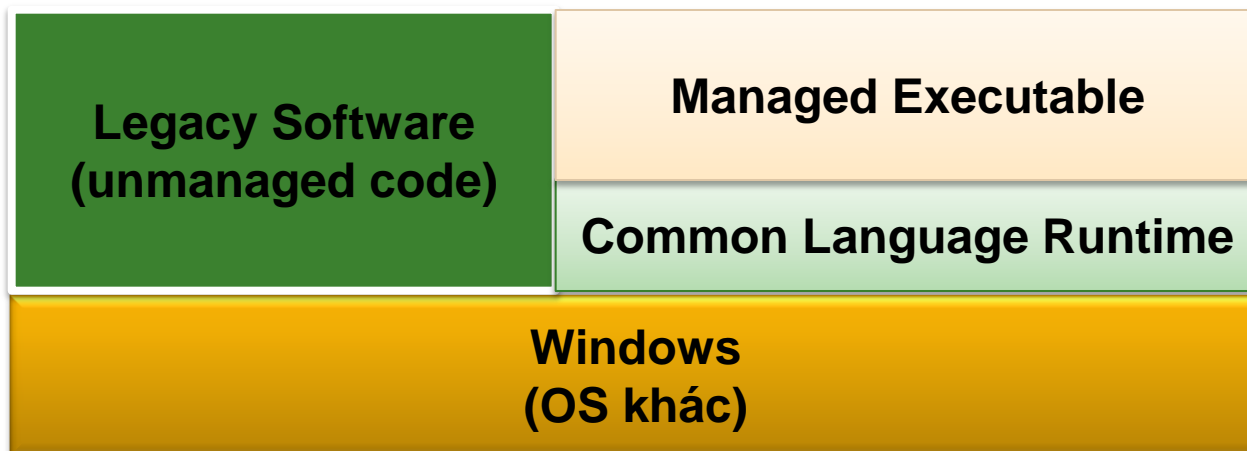
Assembly

Tiến trình thực thi bởi một chương trình C#:

- Khi chương trình được thực thi, CLR sẽ xác nhận đến các Assembly manifest và quyền hạn chạy của chương trình trên hệ thống.
- Nếu hệ bảo vệ hệ thống không cho phép chương trình chạy, chương trình sẽ không chạy.
- Nếu được phép, CLR sẽ thực thi mã lệnh.
- Bit đầu tiên của code được nạp vào bộ nhớ và được biên dịch thành mã nhị phân từ IL bởi JIT.
- Sau khi được biên dịch, mã được thực thi và chứa trong bộ nhớ.

Managed Code

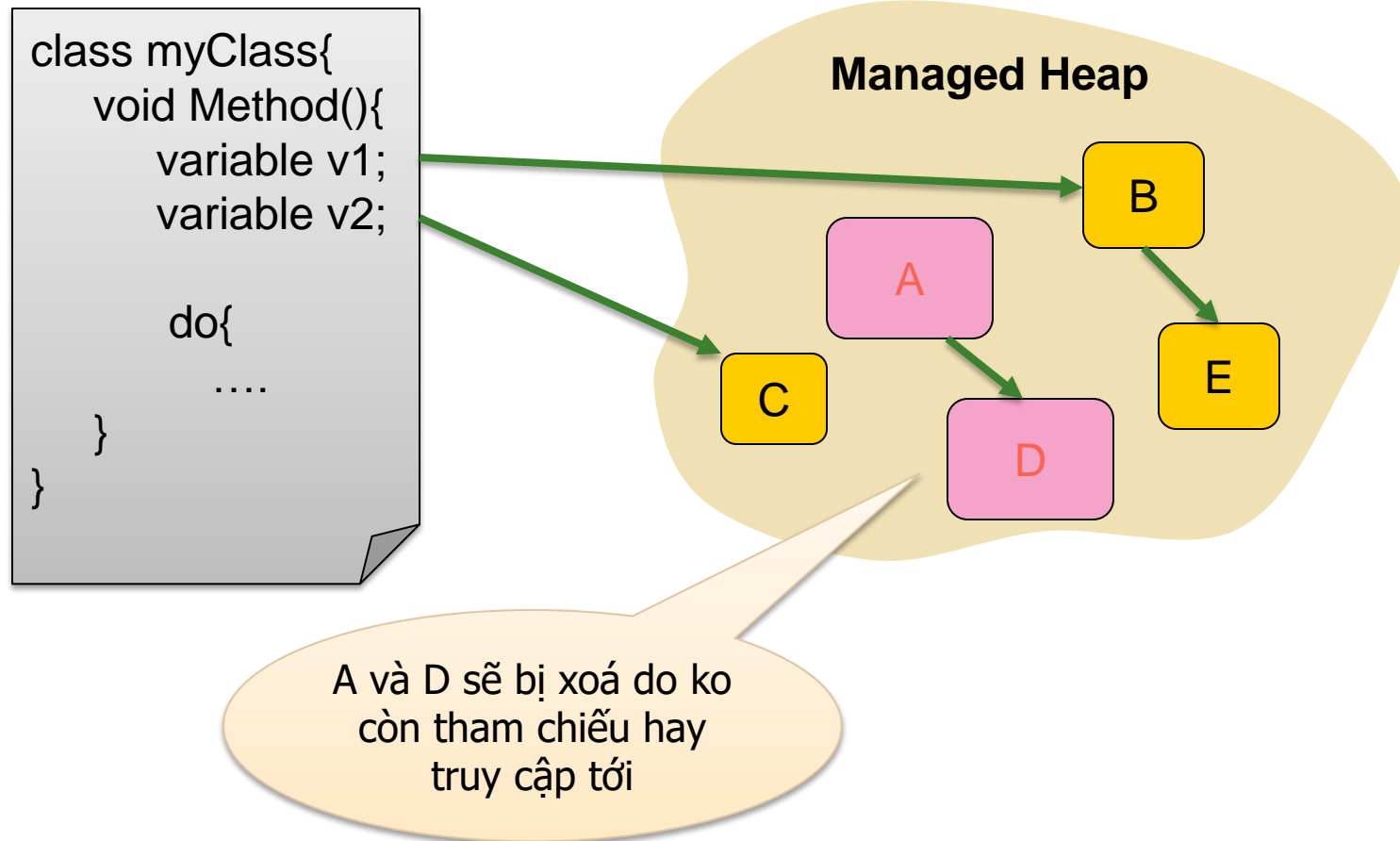
- Managed code là phần mềm được viết để sử dụng trong .NET Framework
- Phần mềm khác thì gọi là Unmanaged code
- “Managed”: chạy dưới sự giám sát của cơ chế thực thi (CLR)



Garbage collection

- Thời gian chạy .NET hoàn toàn phụ thuộc vào garbage collector instead.
- GC là một chương trình hỗ trợ việc thu dọn bộ nhớ.
- Thỉnh thoảng .NET sẽ kiểm tra xem vùng heap đầy chưa để nó tiến hành thu dọn, và nó gọi đây là tiến trình thu dọn rác.
- Trình thu dọn rác sẽ kiểm tra các tham chiếu từ mã của bạn, ví dụ các tham chiếu từ mã của bạn đến các đối tượng được lưu trên heap được nhận dạng, nó có nghĩa là đối tượng đó vẫn còn tham chiếu, các đối tượng không còn tham chiếu nữa sẽ bị huỷ.
- Một điều đặc biệt quan trọng là tính không định trước của trình thu gom rác. Bạn không thể bảo đảm được khi nào trình thu dọn rác sẽ được gọi; nó sẽ được gọi khi CLR cảm thấy cần (nếu bạn không thực hiện lời gọi tường minh).

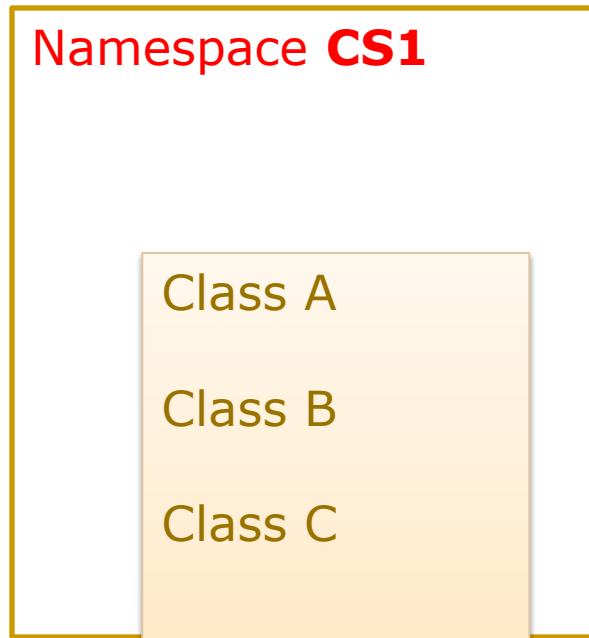
Garbage collection



Garbage collection

- GC xuất hiện (không định trước) khi ko đủ bộ nhớ để cung cấp cho ứng dụng.
- GC thực hiện việc tìm kiếm những đối tượng trong managed heap, xoá nếu ko còn tham chiếu tới.
- Có thể gọi GC một cách tường minh

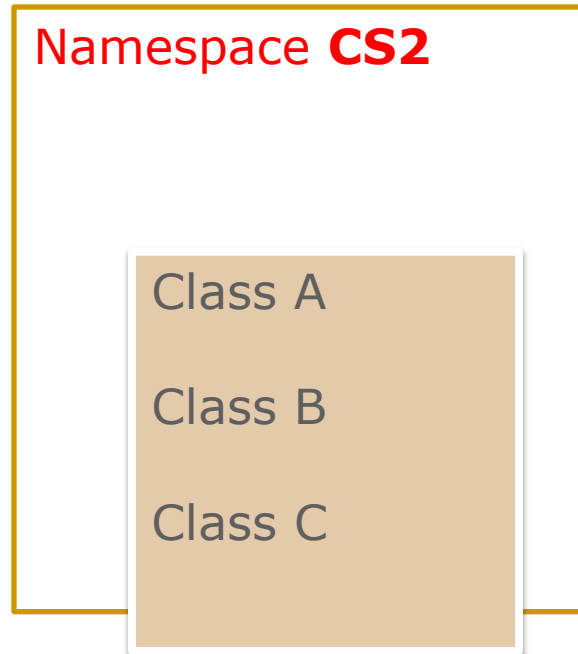
Namespace



CS1 . A...

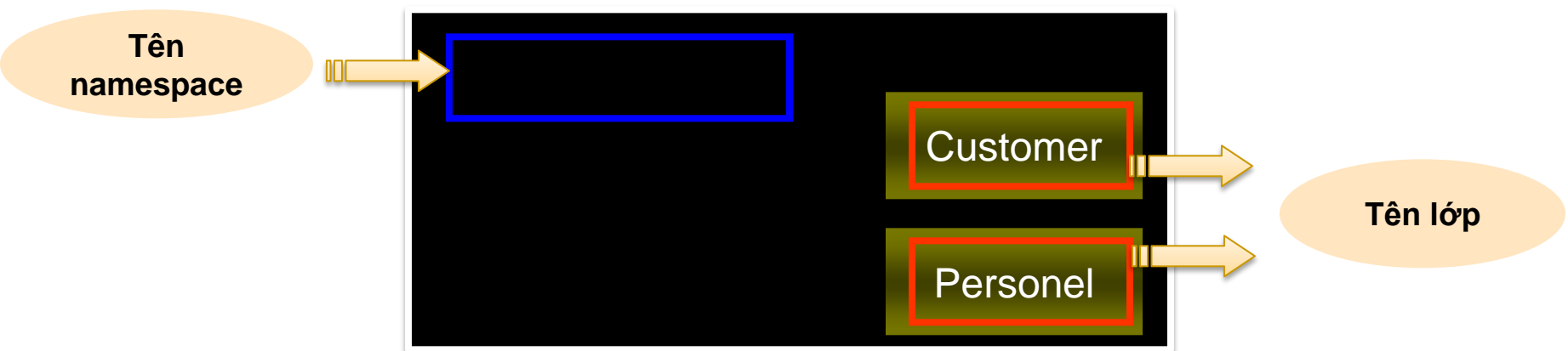
CS2 . A...

- Tránh xung đột tên
- Cho phép nest
- Truy cập đầy đủ qua tên
- Tất cả data type có tiếp đầu ngữ là tên namespace



Namespace

- Hầu hết các lớp cơ sở chung của .NET đều thuộc namespace System
 - Lớp CS Array thuộc System → System.Array
- .NET đề nghị tất cả kiểu do user định nghĩa phải nằm trong 1 namespace



Các lớp .NET Framework

- Thư viện lớp cơ sở .NET là managed code
- Khá đa dạng & đầy đủ:
 - Cho phép kế thừa để phát triển UD
- Bao bọc tất cả các hàm API
 - Dễ sử dụng (khác với VC++ trước đây)

Các lớp .NET Framework

- Các lớp .NET bao gồm:
 - ❑ Đặc tính lõi: IL, kiểu dữ liệu trong CTS
 - ❑ Hỗ trợ Win GUI và control
 - ❑ WebForm (ASP.NET)
 - ❑ Data Access (ADO.NET)
 - ❑ Directory Access
 - ❑ File System, registry access
 - ❑ Networking and web browsing
 - ❑ .NET attributes and reflection
 - ❑ WinOS access
 - ❑ COM interoperability

Framework (Base) Class Library

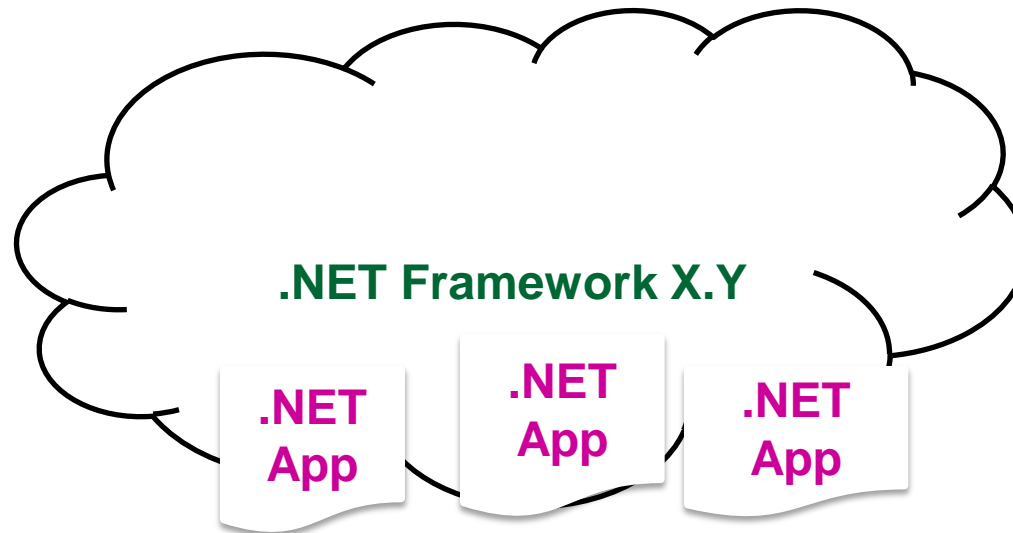
- Thư viện nền tảng cho .NET Framework
- Các namespace cơ bản của FCL/BCL

Namespace	Description
System	Chứa lớp toán học, chuyển đổi dữ liệu
System.IO	Các lớp cho thao tác Input và Output
System.Net	Các lớp liên quan đến network protocol
System.Collections	Chức các lớp liên quan đến xử lý tập hợp
System.Data	Các lớp của ADO.NET
System.Drawing	Các lớp thực thi chức năng GUI
System.Threading	Các lớp lập trình MultiThread
System.Web	Các lớp liên quan đến HTTP protocol
System.Xml	Các lớp liên quan XML

FCL
BCL

Run .NET App from Client

- Required MS .NET Framework compatible
 - MS .NET Framework 2.0, 3.5
- Install .NET 3.5 Full Redistributable package
 - (dotnetfx35.exe, 197MB)
- Windows Vista comes with .NET Framework 3.0



Tóm tắt

- .NET Framework: nền tảng cho ứng dụng mới của MS
- Tất cả các chương trình viết bằng ngôn ngữ khác nhau (trên .NET) sẽ được chuyển về mã thống nhất MSIL
- Cho phép thực thi chéo giữa các ngôn ngữ
- Khái niệm Managed Code
- Cơ chế thu gom vùng nhớ tự động
- Các lớp thư viện .NET phong phú & mạnh mẽ.
- Namespace giúp tổ chức tốt mã nguồn

C# Language

Ngôn ngữ C#

- Ngôn ngữ ra đời cùng với .NET
 - Kết hợp C++ và Java
 - Hướng đối tượng
 - Hướng thành phần
 - Mạnh mẽ (robust) và bền vững (durable)
 - Anders Hejlsberg và MS team xây dựng C#

Ngôn ngữ C#

- Mọi thứ trong C# đều Object oriented
 - Kể cả kiểu dữ liệu cơ bản
- Chỉ cho phép đơn kế thừa
 - Dùng interface để khắc phục
- Lớp Object là cha của tất cả các lớp
 - Mọi lớp đều dẫn xuất từ Object

Ngôn ngữ C#

- Cho phép chia chương trình thành các thành phần nhỏ độc lập nhau
- Mỗi lớp gói gọn trong một file, không cần file header như C/C++
- Bổ sung khái niệm namespace để gom nhóm các lớp
- Bổ sung khái niệm "*property*" cho các lớp
- Khái niệm delegate & event

C# - mạnh mẽ & bền vững

- Garbage Collector
 - Tự động thu hồi vùng nhớ không dùng
- Kiểm soát và xử lý ngoại lệ exception
 - Đoạn mã bị lỗi sẽ không được thực thi
- Type – safe
 - Không cho gán các kiểu dữ liệu khác nhau
- Versioning
 - Đảm bảo sự tương thích giữa lớp con và lớp cha

Vai trò C# trong .NET Framework

- .NET runtime sẽ phổ biến và được cài trong máy client
 - Việc cài đặt App C# như là tái phân phối các thành phần .NET
 - Nhiều App thương mại sẽ được cài đặt bằng C#
- C# tạo cơ hội cho tổ chức xây dựng các App Client/Server n-tier.
- Kết nối ADO.NET cho phép truy cập nhanh chóng & dễ dàng với SQL Server, Oracle...

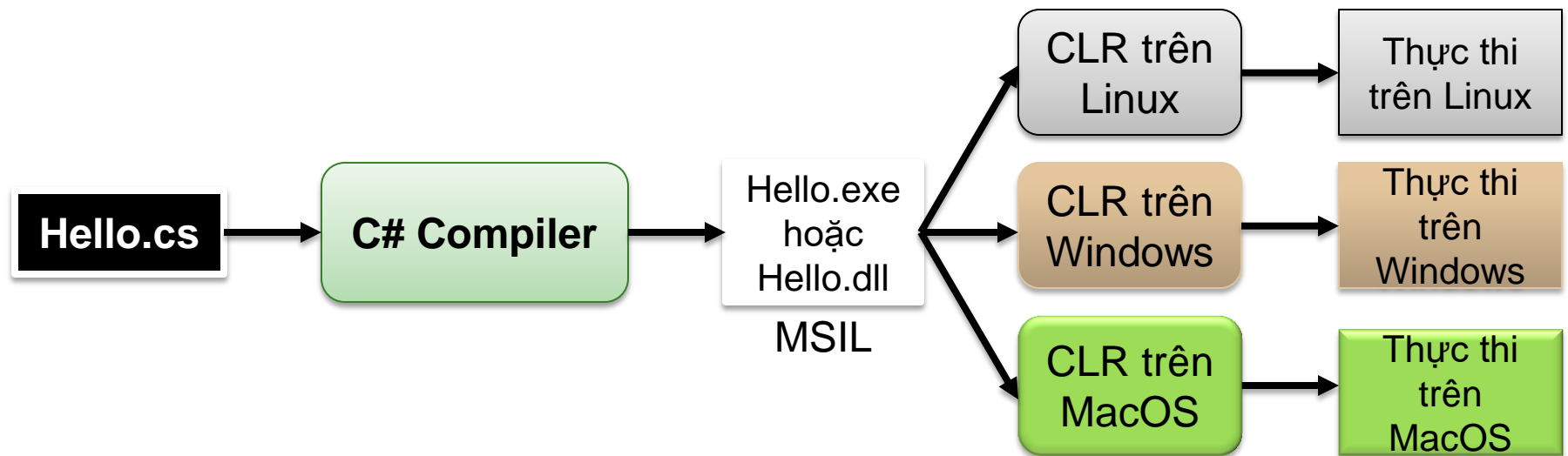
Vai trò C# trong .NET Framework

- Cách tổ chức .NET cho phép hạn chế những vấn đề phiên bản
 - Loại bỏ “*DLL Hell*”...
- ASP.NET viết bằng C#
 - Chạy nhanh hơn (đặc tính của .NET)
 - Mã ASP.NET ko còn là mớ hỗn độn
 - Khả năng bắt lỗi tốt, hỗ trợ mạnh trong quá trình xây dựng App Web.

Quá trình dịch CT C#

- Mã nguồn C# (tập tin *.cs) được biên dịch qua MSIL
 - MSIL: tập tin .exe hoặc .dll
- MSIL được CLR thông dịch qua mã máy
 - Dùng kỹ thuật JIT (just-in-time) để tăng tốc độ

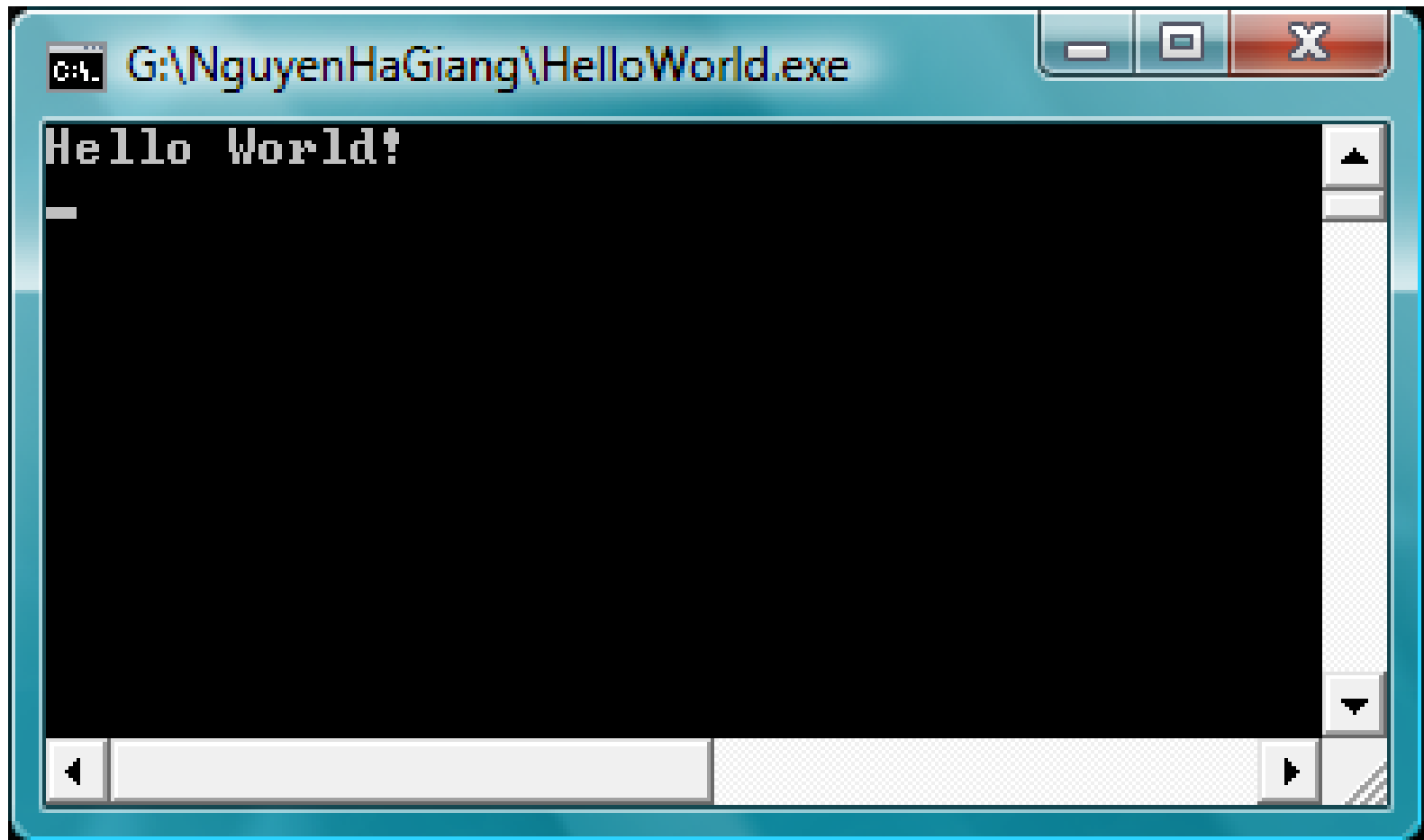
Quá trình dịch CT C#



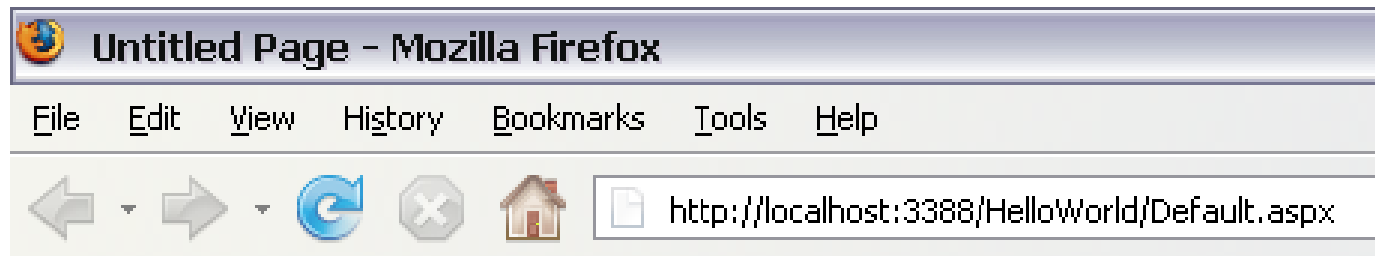
Các loại ứng dụng C#

- Chương trình Console (TUI)
 - Giao tiếp với người dùng bằng bàn phím
 - Không có giao diện đồ họa (GUI)
- Chương trình Windows Form
 - Giao tiếp với người dùng bằng bàn phím và mouse
 - Có giao diện đồ họa và xử lý sự kiện
- Chương trình Web Form
 - Kết hợp với ASP .NET, C# đóng vai trò xử lý bên dưới (underlying code)
 - Có giao diện đồ họa và xử lý sự kiện

Ứng dụng Console

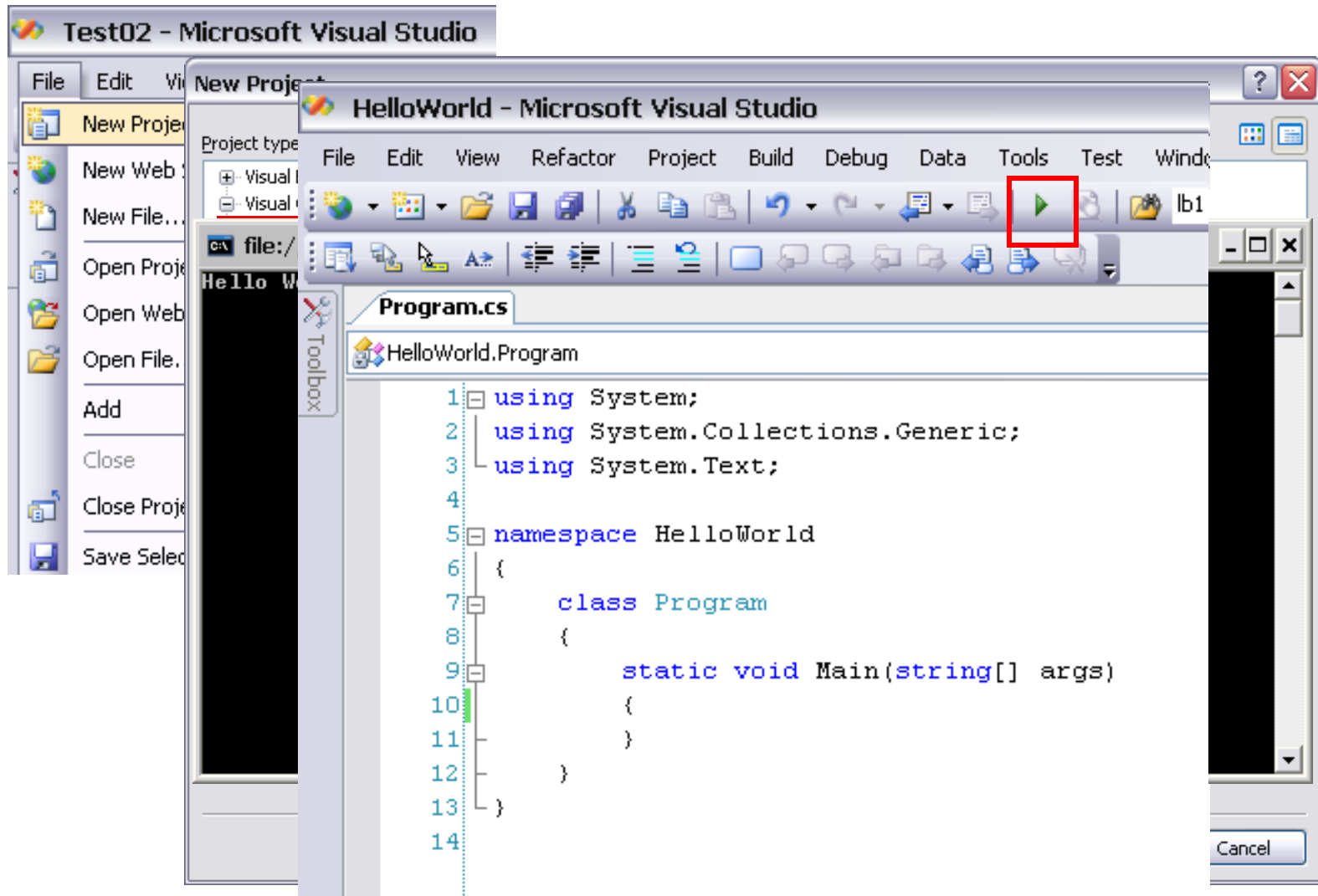


UD WinForm và Web Form



Hello World!

Tạo Ứng Dụng Console



UD C# đầu tiên

```
// Chương trình C# đầu tiên

using System;
using System.Collections.Generic;
using System.Text;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Hello World!");
            Console.ReadLine();
        }
    }
}
```

Cấu trúc chương trình C#

- Phần chú thích (option)

```
// Chương trình C# đầu tiên
```

- Phần khai báo dùng namespace (option)

```
using System;  
using System.Collections.Generic;  
using System.Text;
```

- Phần định nghĩa namespace và lớp

```
namespace HelloWorld {  
    class Program {  
        static void Main(string[] args) {  
            Console.Write("Hello World!");  
            Console.ReadLine();  
        }  
    }  
}
```

Câu lệnh

- Các câu lệnh được viết trong thân của phương thức (ở đây là phương thức Main)
- Thực hiện một công việc nào đó
- Kết thúc bởi dấu chấm phẩy (;)

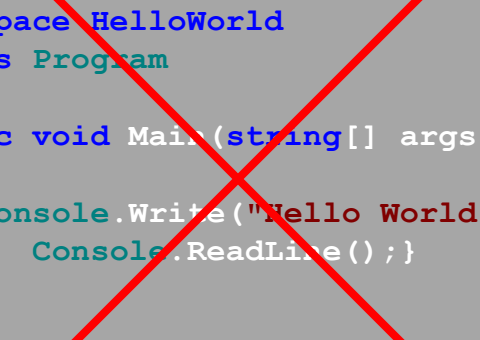
Phương thức Main

Các câu lệnh

```
namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
            Console.ReadLine();
        }
    }
}
```


Khoảng trắng

- Bao gồm
 - Ký tự trắng, ký tự xuống dòng, ký tự tab
 - Dòng trống
- Sử dụng hợp lý \Rightarrow chương trình dễ đọc



```
namespace HelloWorld
{class Program
{
static void Main(string[] args)
{
    Console.WriteLine("Hello World!");
    Console.ReadLine();
}
}
```

```
namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
            Console.ReadLine();
        }
    }
}
```

Chú thích

- Chú thích (comment) được dùng để giải thích về chương trình và các câu lệnh
- Giúp cho chương trình dễ hiểu hơn
- Được bỏ qua khi biên dịch
- Không ảnh hưởng tới kết quả thực thi của chương trình
- Có thể phát sinh ra documentation của chương trình qua chú thích XML

Hai cách tạo chú thích cơ bản

- Gõ phần chú thích sau cặp ký tự `//`
- Gõ phần chú thích giữa cặp ký tự `/*` và `*/`

```
/* Chương trình C# đầu tiên
   In ra câu chào "Hello World" */

using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Hello World!"); // Xuất ra câu chào
            Console.ReadLine(); // Chờ nhấn Enter
        }
    }
}
```

Tạo Ứng Dụng WinForm

Cơ chế xử lý sự kiện code behind

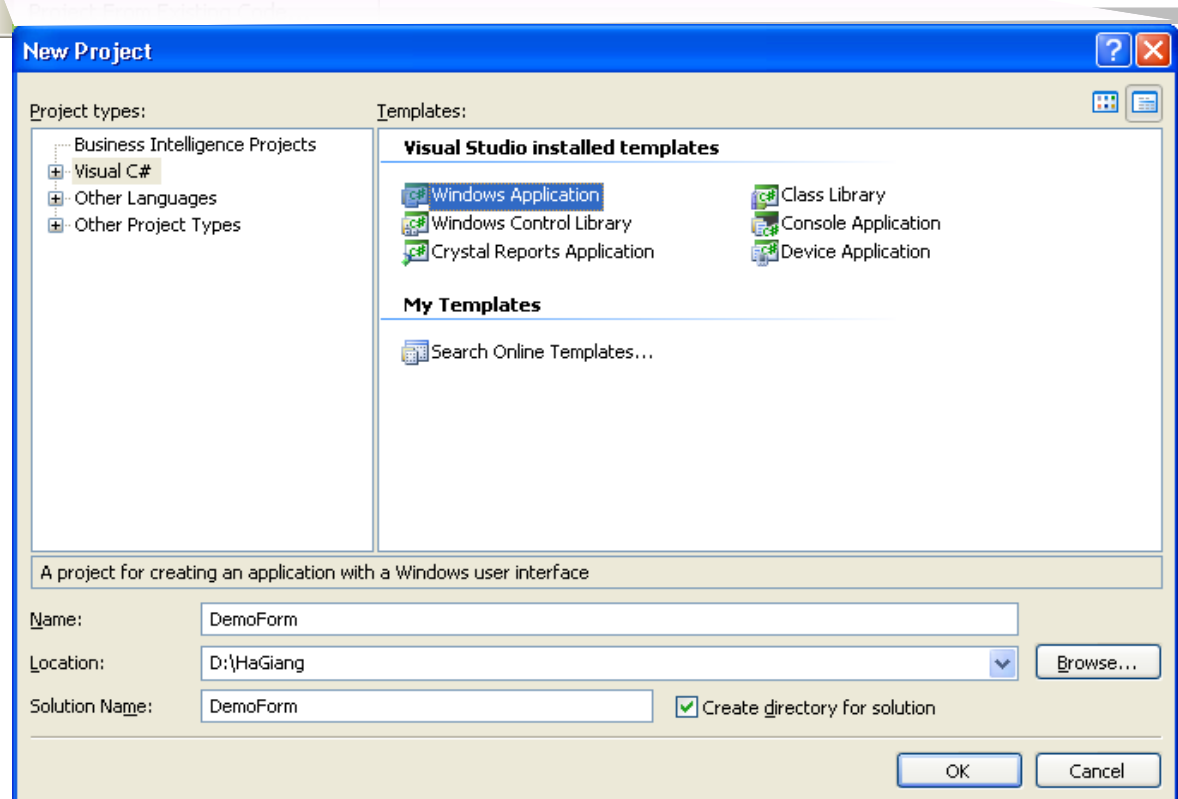
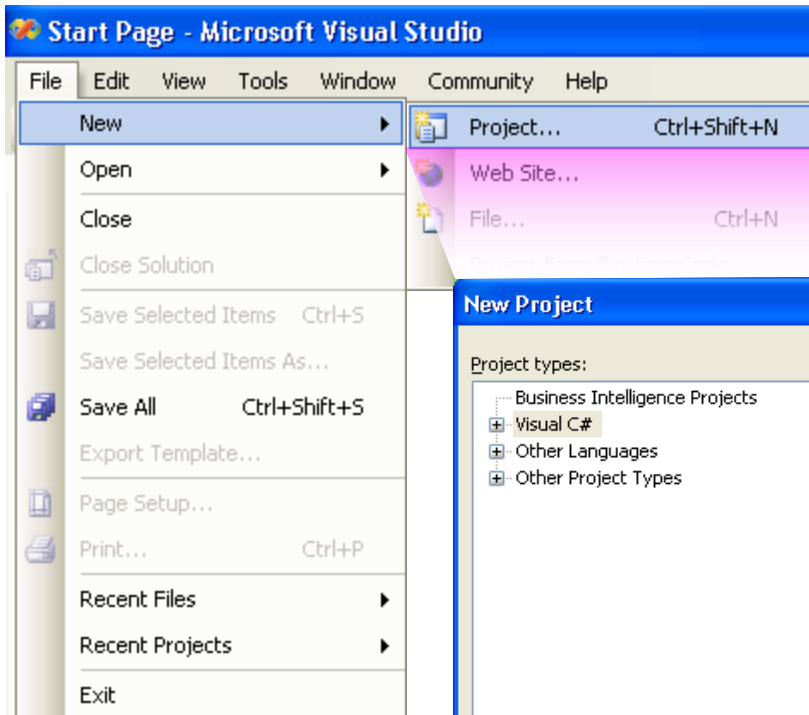
Hỗ trợ WYSISYG cho GUI design



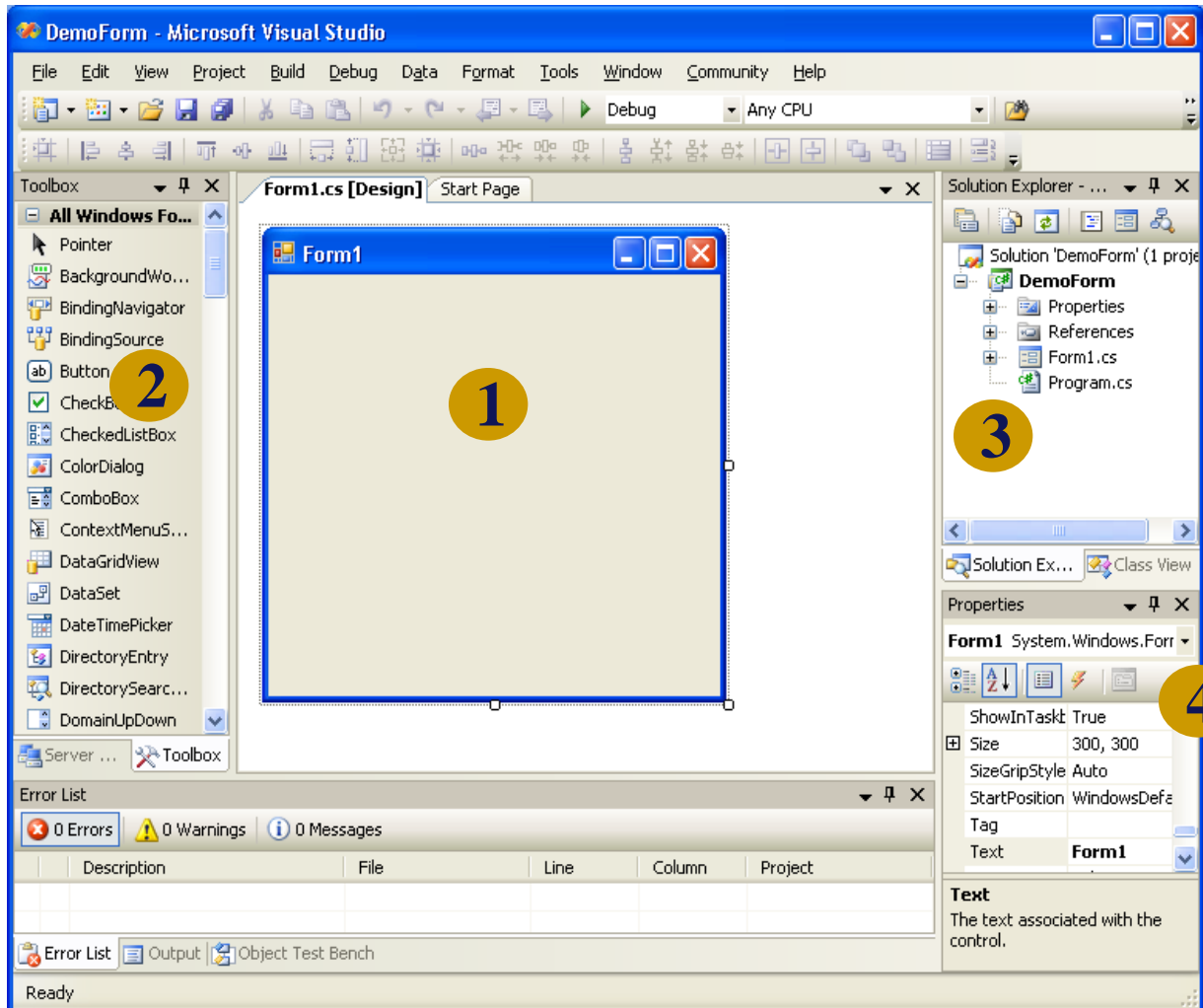
Nhanh chóng & dễ dàng tạo UD Windows Form

Tạo WinForm App

Tạo project: Windows App



Tạo WinForm App



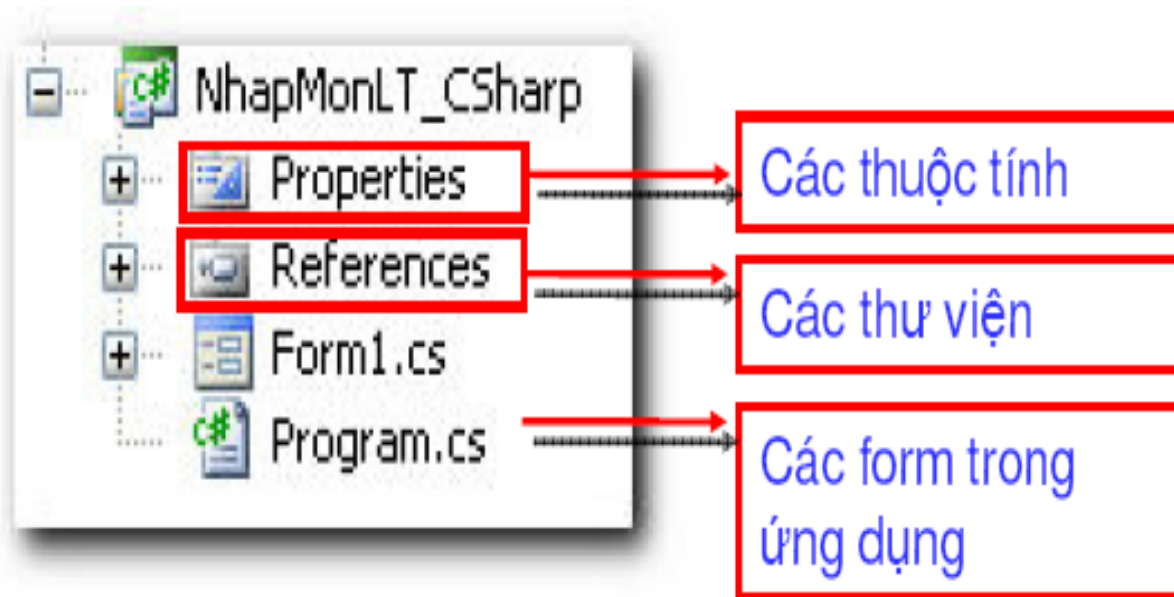
Windows App do

- 1: form ứng dụng
- 2: control toolbox
- 3: Solution Explorer
- 4: Form properties

Tạo WinForm App

Các thành phần của một project: (cửa sổ Solution)

- Khi tạo ứng dụng C#.Net, hệ thống sẽ phát sinh ra cấu trúc thư mục lưu trữ như sau:



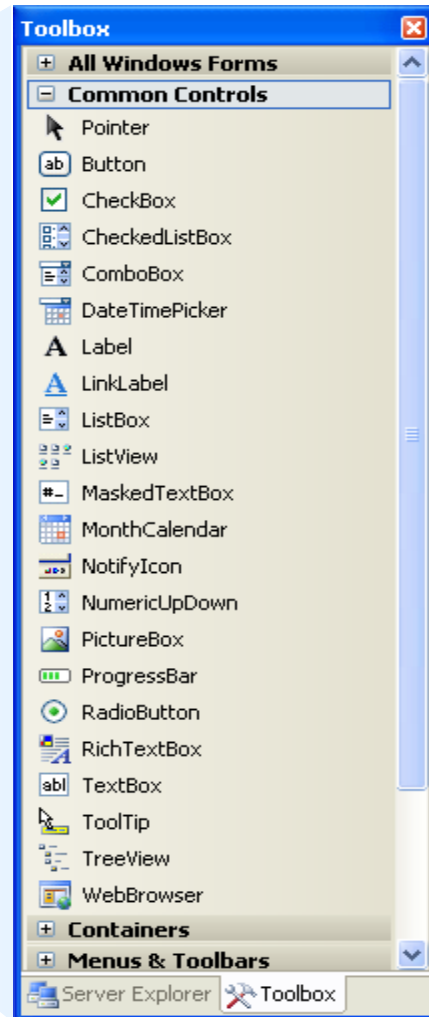
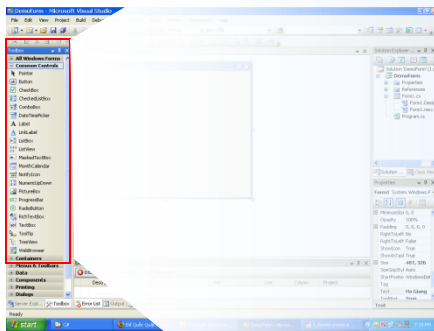
Tạo WinForm App

Các kiểu file của các thành phần trong project

Các file lưu trữ chính

- *.cs: tập tin lưu code của form viết bằng C#
- *.Designer.cs: tập tin lưu phần thiết kế của form
- *.resx: tập tin lưu tài nguyên của form
- *.csproj: tập tin quản lý ứng dụng (file dùng để mở project)
- *.sln: tập tin quản lý đồ án (khi cần phối hợp nhiều project trong ứng dụng)

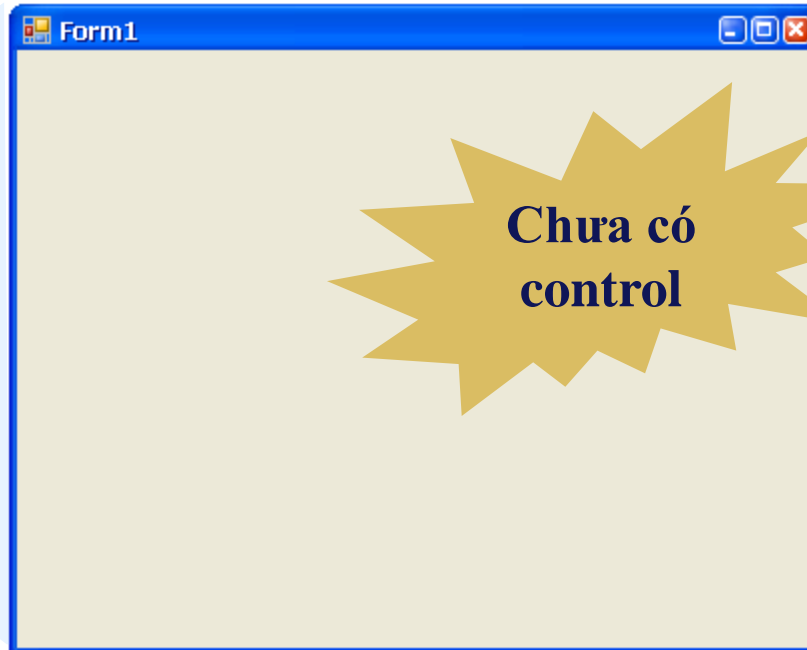
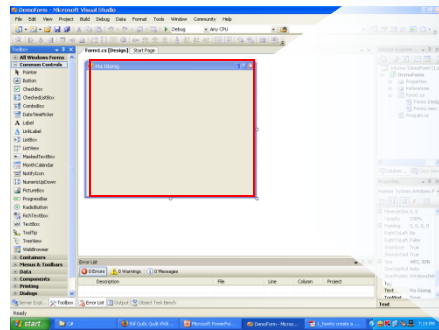
Toolbox



Toolbox

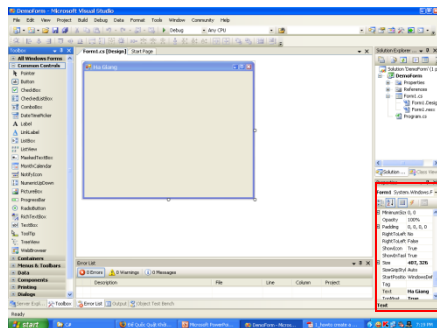
- Kéo thả control lên form
- Code được phát sinh tự động

Giao diện thiết kế form





Form chính của ứng dụng

Cửa sổ properties



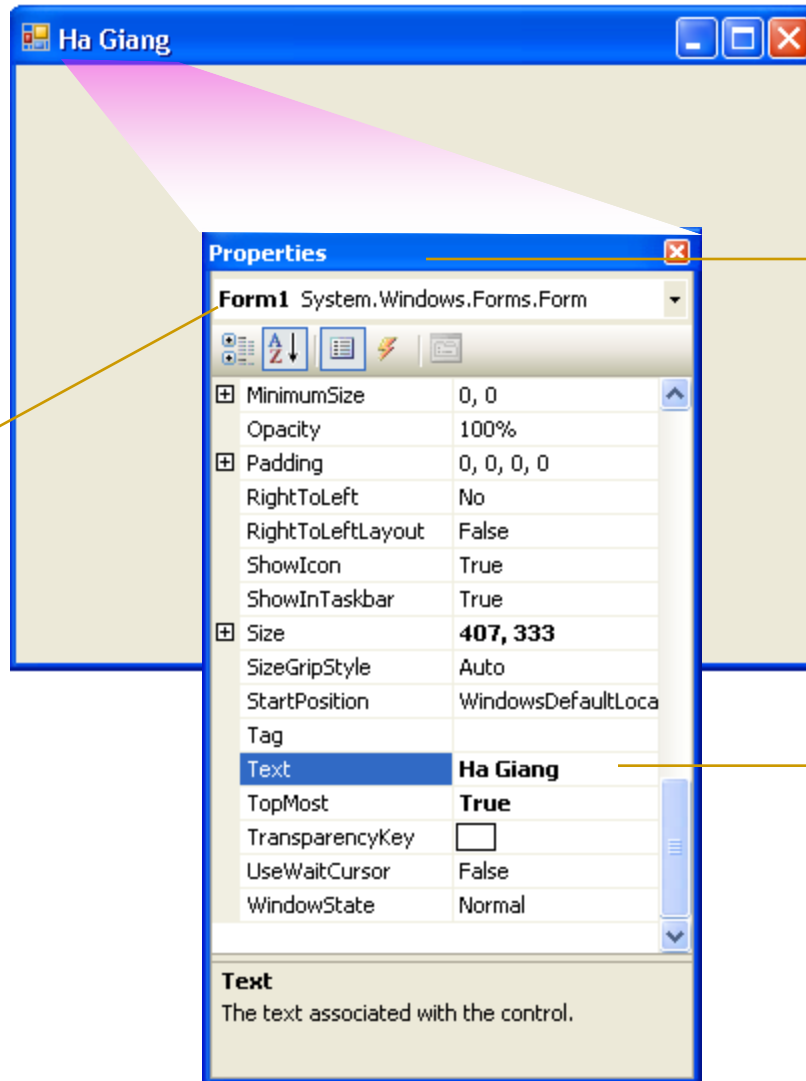
Properties

Form1 System.Windows.Forms.Form

(ApplicationSetting)	
(DataBindings)	
(Name)	Form1
AcceptButton	(none)
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
AutoScaleMode	Font
AutoScroll	False
AutoScrollMargin	0, 0
AutoScrollMinSize	0, 0
AutoSize	False
AutoSizeMode	GrowOnly
AutoValidate	EnablePreventFocus
BackColor	 Control
BackgroundImage	 (none)
BackgroundImageL	Tile
CancelButton	(none)
CausesValidation	True
ContextMenuStrip	(none)

Cửa sổ properties của form

Cửa sổ properties



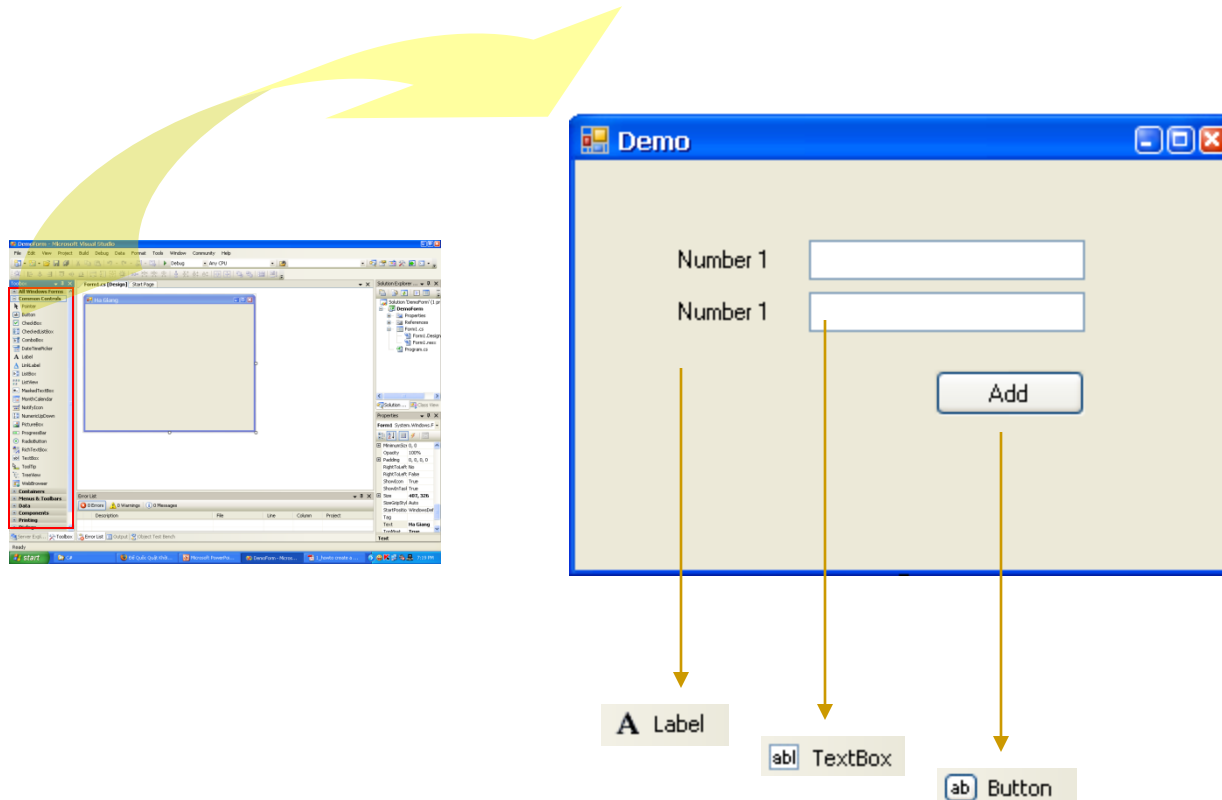
Tên của form
chính là tên
lớp

Dễ dàng hiệu chỉnh
form thông qua cửa
sổ Properties

Thay đổi title

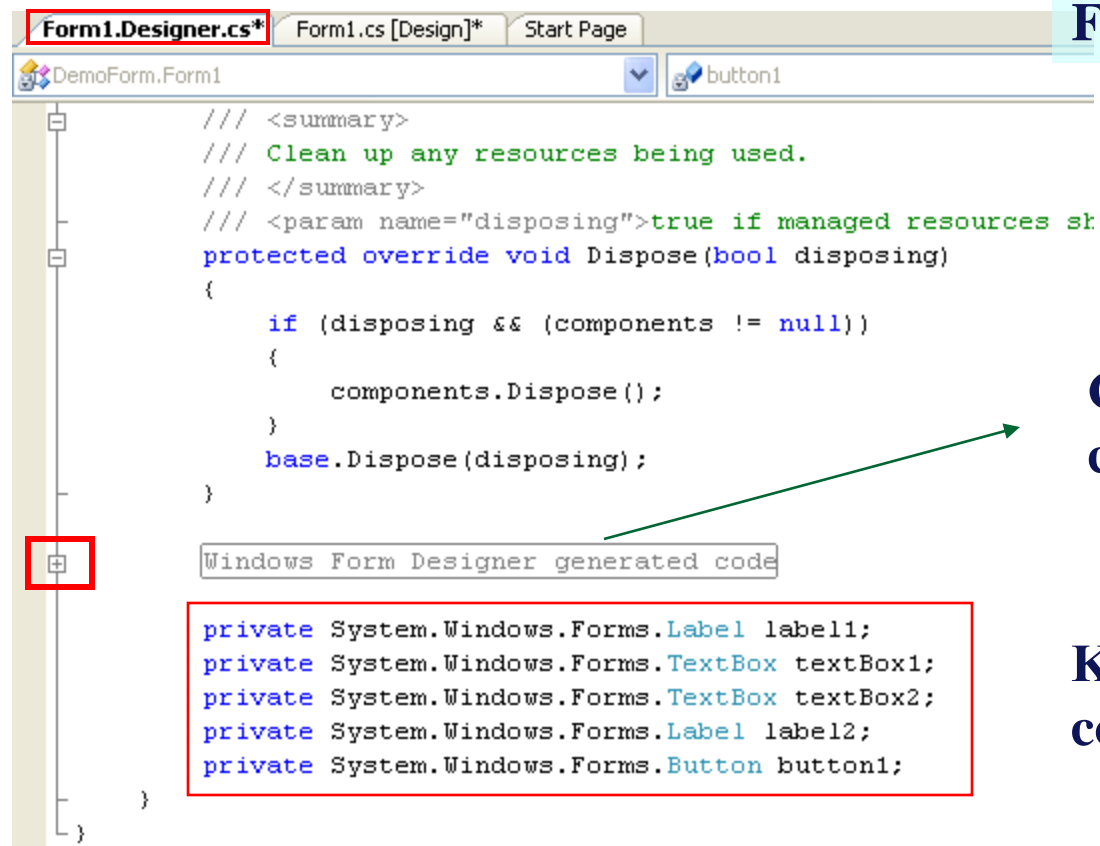
Thêm control vào form

- Kéo thả control vào form



Code của phần design

- Phần code thiết kế Form1 được tạo tự động

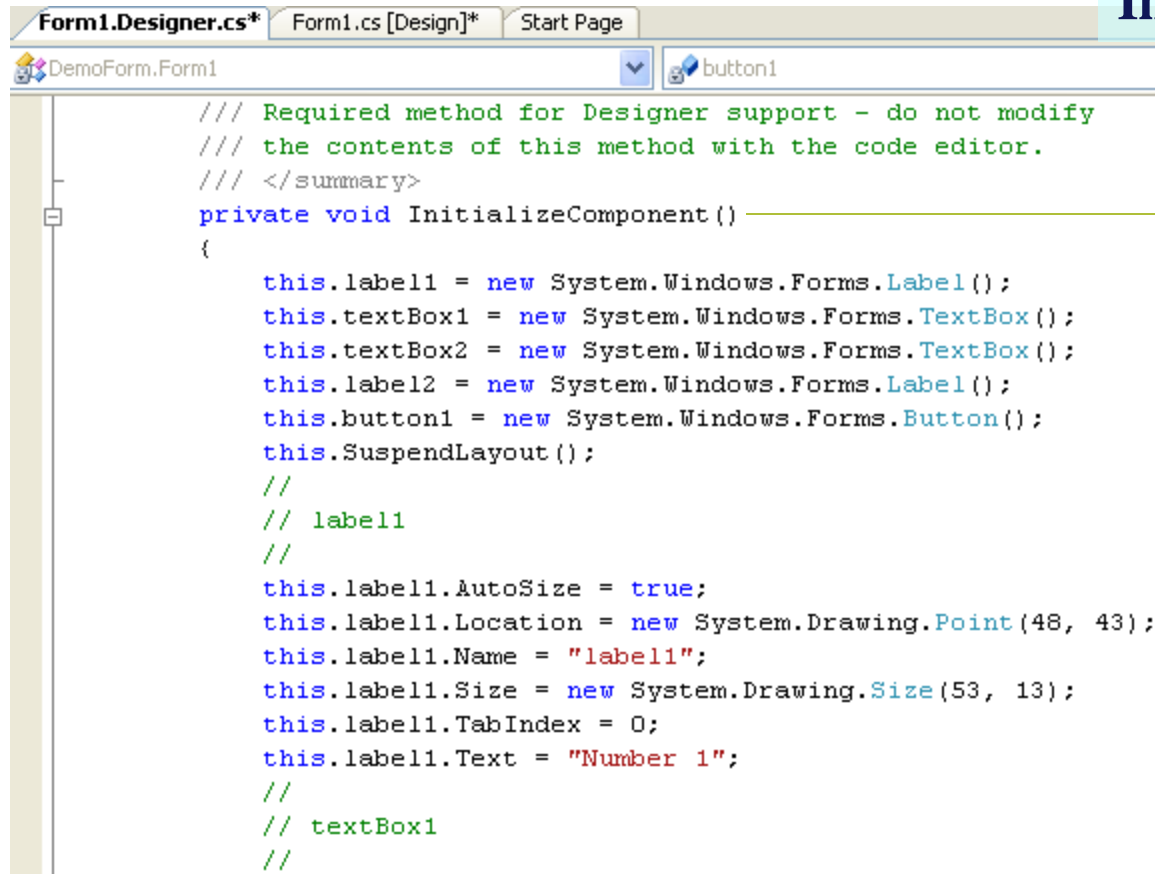


Form1.Designer.cs

Chứa code khởi tạo control

Khai báo các đối tượng control trên Form1

Code của phần design



```
Form1.Designer.cs* Form1.cs [Design]* Start Page
DemoForm.Form1 button1

/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.label1 = new System.Windows.Forms.Label();
    this.textBox1 = new System.Windows.Forms.TextBox();
    this.textBox2 = new System.Windows.Forms.TextBox();
    this.label2 = new System.Windows.Forms.Label();
    this.button1 = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // label1
    //
    this.label1.AutoSize = true;
    this.label1.Location = new System.Drawing.Point(48, 43);
    this.label1.Name = "label1";
    this.label1.Size = new System.Drawing.Size(53, 13);
    this.label1.TabIndex = 0;
    this.label1.Text = "Number 1";
    //
    // textBox1
    //
```

InitializeComponent

Tạo đối tượng

Lần lượt khai
báo các thuộc
tính cho các
control

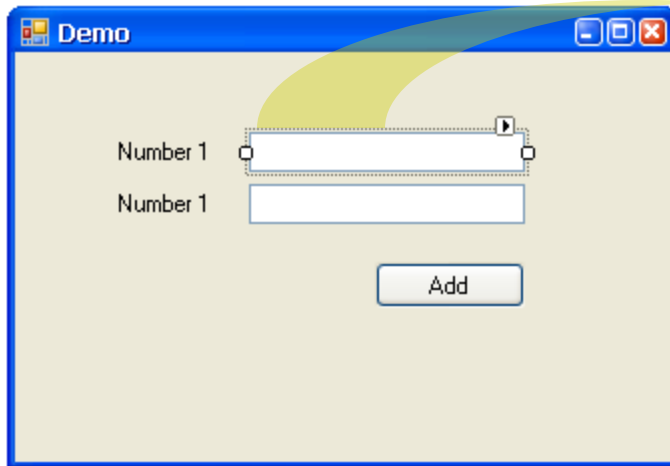
Code của phần design

InitializeComponent

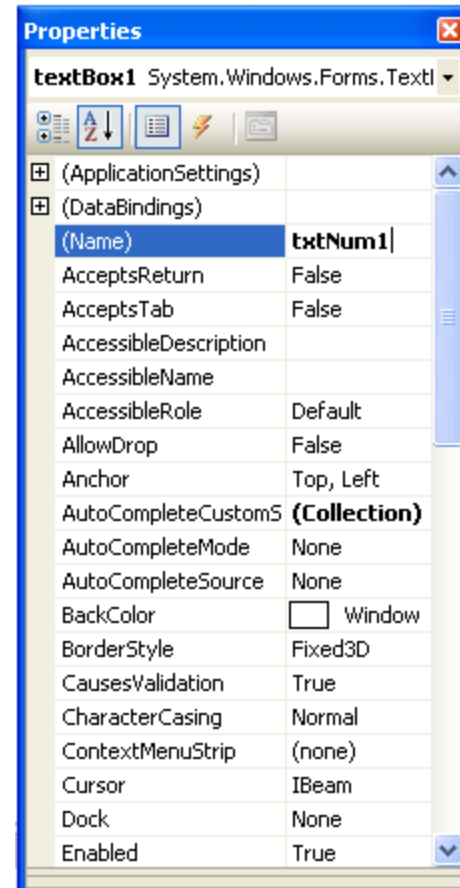
```
//  
// Form1  
//  
this.AutoScaleMode = new System.Drawing.SizeF(6F, 13F);  
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;  
this.ClientSize = new System.Drawing.Size(328, 205);  
this.Controls.Add(this.BtnAdd);  
this.Controls.Add(this.txtNum2);  
this.Controls.Add(this.label2);  
this.Controls.Add(this.txtNum1);  
this.Controls.Add(this.label1);  
this.Name = "Form1";  
this.Text = "Demo";  
this.TopMost = true;  
this.ResumeLayout(false);  
this.PerformLayout();
```

**Đưa các control vào danh sách
control của Form1**

Sửa thuộc tính của control



Thay đổi các giá trị qua cửa sổ
properties -> VS tự cập nhật
code



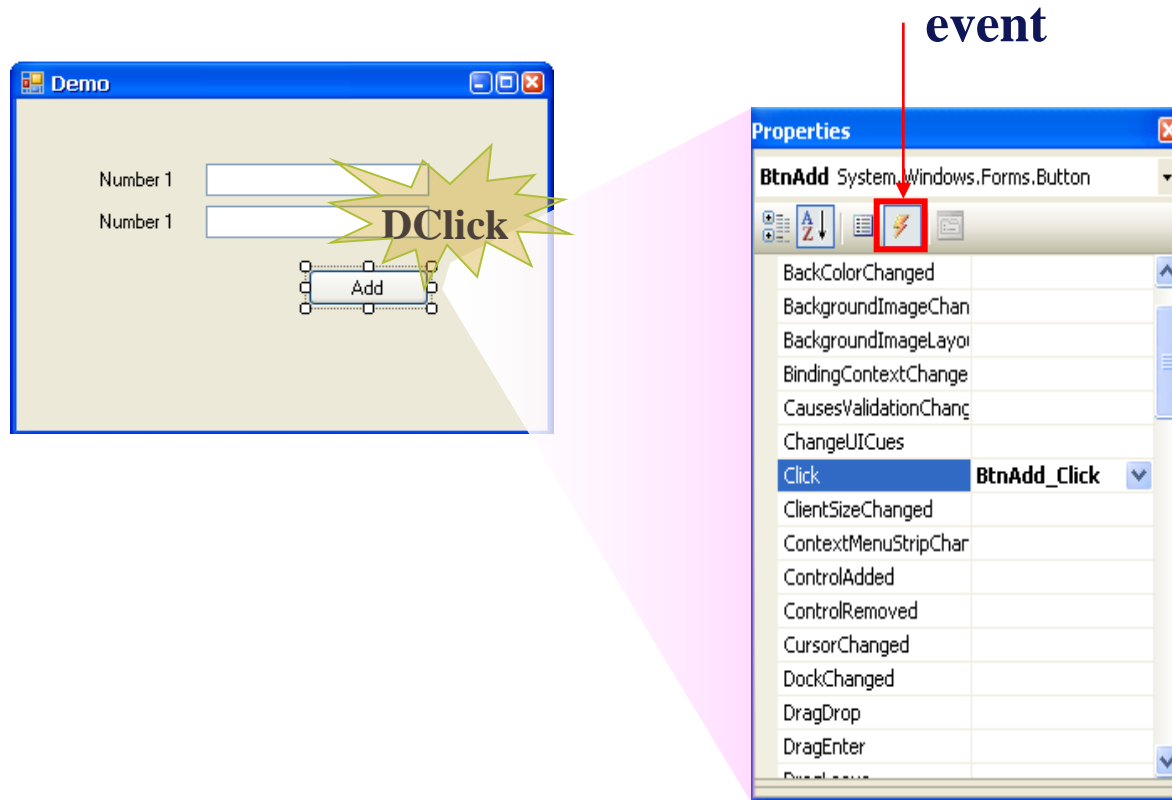
Đổi tên thành
txtNum1

Phần xử lý

- Khi click vào Add -> cộng 2 giá trị và xuất kết quả
- Thực hiện
 - Button Add cung cấp sự kiện click
 - Form sẽ được cảnh báo khi Add được click
 - Form sẽ lấy dữ liệu từ 2 textbox và cộng -> kết quả
- Cơ chế event
 - Button đưa ra sự kiện click: đối tượng publish
 - Form quan tâm đến sự kiện click của button, Form có sẽ phần xử lý ngay khi button click.
 - Phần xử lý của form gọi là Event Handler
 - Form đóng vai trò là lớp subscribe

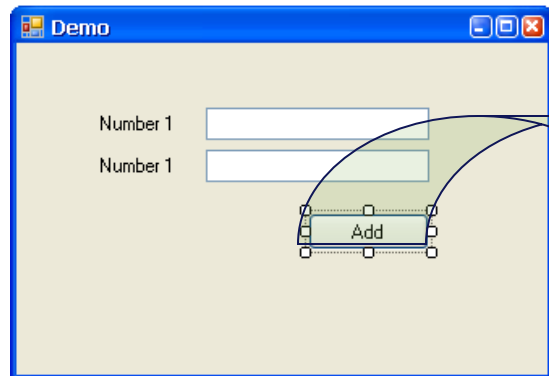
Khai báo event handler

- Kích đúp vào button Add trên màn hình thiết kế cho phép tạo event handler cho sự kiện này.



Cửa sổ quản lý
event của BtnAdd

Khai báo event handler



**Event handler cho
button Add**

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

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

Cùng signature method với System.EventHandler

Khai báo event handler

InitializeComponent

```
//  
// BtnAdd  
//  
this.BtnAdd.Location = new System.Drawing.Point(180, 105);  
this.BtnAdd.Name = "BtnAdd";  
this.BtnAdd.Size = new System.Drawing.Size(75, 23);  
this.BtnAdd.TabIndex = 4;  
this.BtnAdd.Text = "Add";  
this.BtnAdd.UseVisualStyleBackColor = true;  
this.BtnAdd.Click += new System.EventHandler(this.BtnAdd_Click);
```

↓
Sự kiện click

↓
**Trình xử lý được gọi
khi event xảy ra**

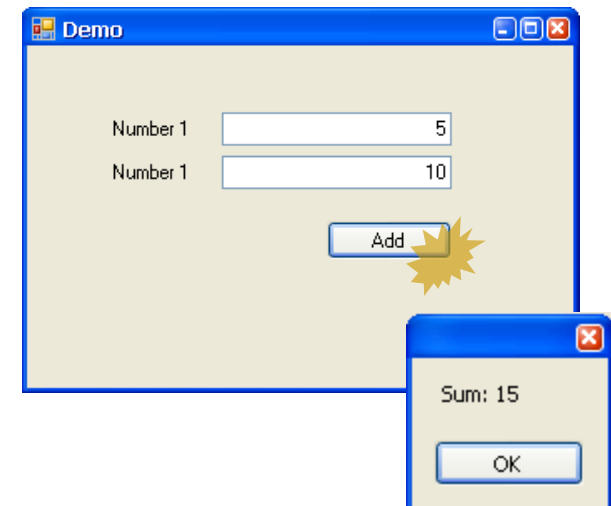
↓
Delegate chuẩn cho event handler

Viết phần xử lý

- Phần xử lý của Form1 khi button click
 - Lấy giá trị của 2 textbox, cộng kết quả và xuất ra MessageBox

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

    private void BtnAdd_Click(object sender, EventArgs e)
    {
        int sum;
        sum = int.Parse(txtNum1.Text) + int.Parse(txtNum2.Text);
        MessageBox.Show("Sum: " + sum.ToString());
    }
}
```



Tóm tắt

- C# là một ngôn ngữ lập trình của .NET
 - Là ngôn ngữ hiện đại
 - Hướng đối tượng, hướng thành phần
 - An toàn kiểu, mạnh mẽ, bền bỉ
- Có ba loại chương trình C#
 - Console, Windows Form, Web Form
- Tạo ra một chương trình C# đơn giản
 - Tạo một ứng dụng Console
 - Viết các câu lệnh trong thân của phương thức Main
 - Tạo ứng dụng Winform

Câu hỏi và bài tập

Câu hỏi 1: Một chương trình C# có thể chạy trên bất cứ máy nào?

Trả lời 1:

- Không phải tất cả. Một chương trình C# chỉ chạy trên máy có Common Language Runtime (CLR) được cài đặt.
- Nếu chúng ta copy một chương trình exe của C# qua một máy không có CLR thì chúng ta sẽ nhận được một lỗi.
- Trong những phiên bản của Windows không có CLR chúng ta sẽ được báo rằng thiếu tập tin DLL.

Câu hỏi và bài tập

Câu hỏi 2: Nếu muốn đưa chương trình mà ta viết cho một người bạn thì tập tin nào mà chúng ta cần đưa?

Trả lời 2:

- Thường cách tốt nhất là đưa chương trình đã biên dịch. Điều này có nghĩa rằng sau khi mã nguồn được biên dịch, chúng ta sẽ có một chương trình thực thi (tập tin có phần mở rộng *.exe).
- Và những người bạn của chúng ta không cần thiết phải có trình biên dịch C#. Họ chỉ cần có C# runtime trên máy tính (như CLR của Microsoft) là có thể chạy được chương trình của chúng ta.

Câu hỏi và bài tập

Câu hỏi 3: Sau khi tạo ra được tập tin thực thi .exe. Có cần thiết giữ lại tập tin nguồn không?

Trả lời 3:

- Nếu chúng ta từ bỏ tập tin mã nguồn thì sau này sẽ rất khó khăn cho việc mở rộng hay thay đổi chương trình, do đó cần thiết phải giữ lại các tập tin nguồn.
- Hầu hết các IDE tạo ra các các tập tin nguồn (.cs) và các tập tin thực thi.
- Cũng như giữ các tập tin nguồn chúng ta cũng cần thiết phải giữ các tập tin khác như là các tài nguyên bên ngoài các icon, image, form.. Chúng ta sẽ lưu giữ những tập tin này trong trường hợp chúng ta cần thay đổi hay tạo lại tập tin thực thi.

Câu hỏi và bài tập

Câu hỏi 4: Nếu trình biên dịch C# đưa ra một trình soạn thảo, có phải nhất thiết phải sử dụng nó?

Trả lời 4:

- Không hoàn toàn như vậy. Chúng ta có thể sử dụng bất cứ trình soạn thảo văn bản nào và lưu mã nguồn dưới dạng tập tin văn bản. Nếu trình biên dịch đưa ra một trình soạn thảo thì chúng ta nên sử dụng nó.
- Nếu chúng ta có một trình soạn thảo khác tốt hơn chúng ta có thể sử dụng nó. Một số các tiện ích soạn thảo mã nguồn có thể giúp cho ta dễ dàng tìm các lỗi cú pháp, giúp tạo một số mã nguồn tự động đơn giản...

Câu hỏi và bài tập

Câu hỏi 5: Có thể không quan tâm đến những cảnh báo khi biên dịch mã nguồn

Trả lời 5:

- Một vài cảnh báo không ảnh hưởng đến chương trình khi chạy, nhưng một số khác có thể ảnh hưởng đến chương trình chạy.
- Nếu trình biên dịch đưa ra cảnh báo, tức là tín hiệu cho một thứ gì đó không đúng.
- Hầu hết các trình biên dịch cho phép chúng ta thiết lập mức độ cảnh báo. Bằng cách thiết lập mức độ cảnh báo chúng ta có thể chỉ quan tâm đến những cảnh báo nguy hiểm, hay nhận hết tất cả những cảnh báo.
- Nói chung cách tốt nhất là chúng ta nên xem tất cả những cảnh báo để sửa chữa chúng, một chương trình tạm gọi là đạt yêu cầu khi không có lỗi biên dịch và cũng không có cảnh báo nhưng chưa chắc đã chạy đúng kết quả!

Câu hỏi và bài tập

Câu hỏi thêm

- Hãy đưa ra 3 lý do tại sao ngôn ngữ C# là một ngôn ngữ lập trình tốt?
- IL và CLR viết tắt cho từ nào và ý nghĩa của nó?
- Đưa ra các bước cơ bản trong chu trình xây dựng chương trình?
- Trong biên dịch dòng lệnh thì lệnh nào được sử dụng để biên dịch mã nguồn .cs và lệnh này gọi chương trình nào?
- Phần mở rộng nào mà chúng ta nên sử dụng cho tập tin mã nguồn C#?

Câu hỏi và bài tập

Câu hỏi thêm

6. Một tập tin .txt chứa mã nguồn C# có phải là một tập tin mã nguồn C# hợp lệ hay không? Có thể biên dịch được hay không?
7. Ngôn ngữ máy là gì? Khi biên dịch mã nguồn C# ra tập tin .exe thì tập tin này là ngôn ngữ gì?
8. Nếu thực thi một chương trình đã biên dịch và nó không thực hiện đúng như mong đợi của chúng ta, thì điều gì chúng ta cần phải làm?
9. Một lỗi tương tự như bên dưới thường xuất hiện khi nào?
10. Tại sao phải khai báo static cho hàm Main của lớp?

Câu hỏi và bài tập

Câu hỏi thêm

11. Một mã nguồn C# có phải chứa trong các lớp hay là có thể tồn tại bên ngoài lớp như C/C++?
12. So sánh sự khác nhau cơ bản giữa C# và C/C++, C# với Java, hay bất cứ ngôn ngữ cấp cao nào mà bạn đã biết?
13. Con trỏ có còn được sử dụng trong C# hay không? Nếu có thì nó được quản lý như thế nào?
14. Khái niệm và ý nghĩa của namespace trong C#? Điều gì xảy ra nếu như ngôn ngữ lập trình không hỗ trợ namespace?

Bài tập

- *Bài tập 1: Nhập vào chương trình sau và biên dịch nó. Cho biết chương trình thực hiện điều gì?*

using System;

class variables

```
{ public static void Main()
```

```
{ int radius = 4;
```

```
    const double PI = 3.14159;
```

```
    double circum, area;
```

```
    area = PI * radius* radius;
```

```
    circum = 2 * PI * radius;
```

```
    // in kết quả
```

```
    Console.WriteLine("Ban kinh = {0}, PI = {1}", radius, PI);
```

```
    Console.WriteLine("Dien tich {0}", area);
```

```
    Console.WriteLine("Chu vi {0}", circum);
```

```
}
```

```
}
```


Bài tập

- *Bài tập 2: Nhập vào chương trình sau và biên dịch. Cho biết chương trình thực hiện điều gì?*

```
class AClass
{
    static void Main()
    {
        int x, y;
        for( x = 0; x < 10; x++, System.Console.Write("\n"));
        for( y = 0 ; y < 10; y++,
            System.Console.WriteLine("{0}",y));
    }
}
```

Bài tập

Bài tập 3: Sửa lỗi và biên dịch chương trình sau

```
class Test
{
    public static void Main()
    {
        Console.WriteLine("Xin chao");
        Consoile.WriteLine("Tam biet");
    }
}
```

Bài tập

Bài tập 4: Sửa lỗi và biên dịch chương trình sau

```
class Test
{
    public void Main()
    {
        Console.WriteLine('Xin chao');
        Consoile.WriteLine('Tam biet');
    }
}
```

Tài liệu tham khảo

1. Professional C#, 2nd Edition, Wrox Press Ltd.
2. A programmer's Introduction to C#, Eric Gunnerson, Apress, 2000
3. Programming C#, Jesse Liberty, O'Reilly, First Edition, 2001
4. C# bible, Jeff Ferguson et al, Wiley Publishing, 2002
5. Thinking in C#, Larry O'Brien, Bruce Eckel, Prentice Hall.
6. Presenting C#, Sams Publishing, 2002
7. C# Language Reference, Anders Hejlsberg and Scott Wiltamuth, Microsoft Corp.