

Lab 1

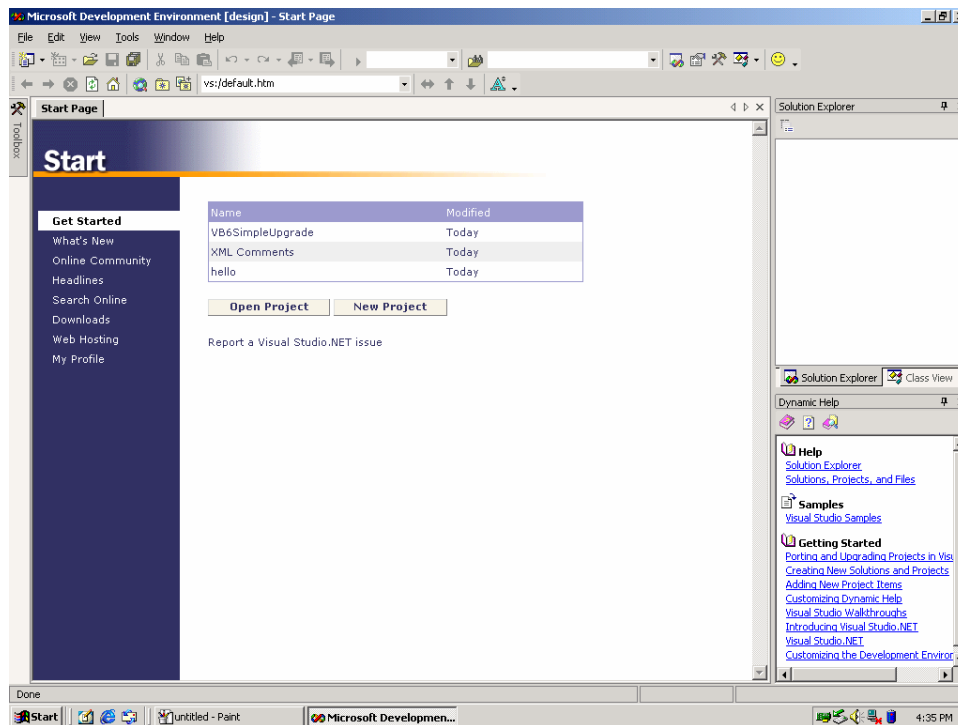
I. Làm quen với Visual Studio .NET

Thực hiện các nhiệm vụ sau.

- Tạo một Visual Studio .NET Solution
- Tạo một C#/VB .NET project
- Thiết lập các properties cho project
- Sử dụng Server Explorer, Object browser, Task list

Exercise 1 – Bắt đầu với Visual Studio .NET

1. Để sử dụng VS.NET thực hiện như sau: **Start | Programs | Microsoft Visual Studio .NET | Microsoft Visual Studio .NET**. Start Page xuất hiện như hình dưới đây.

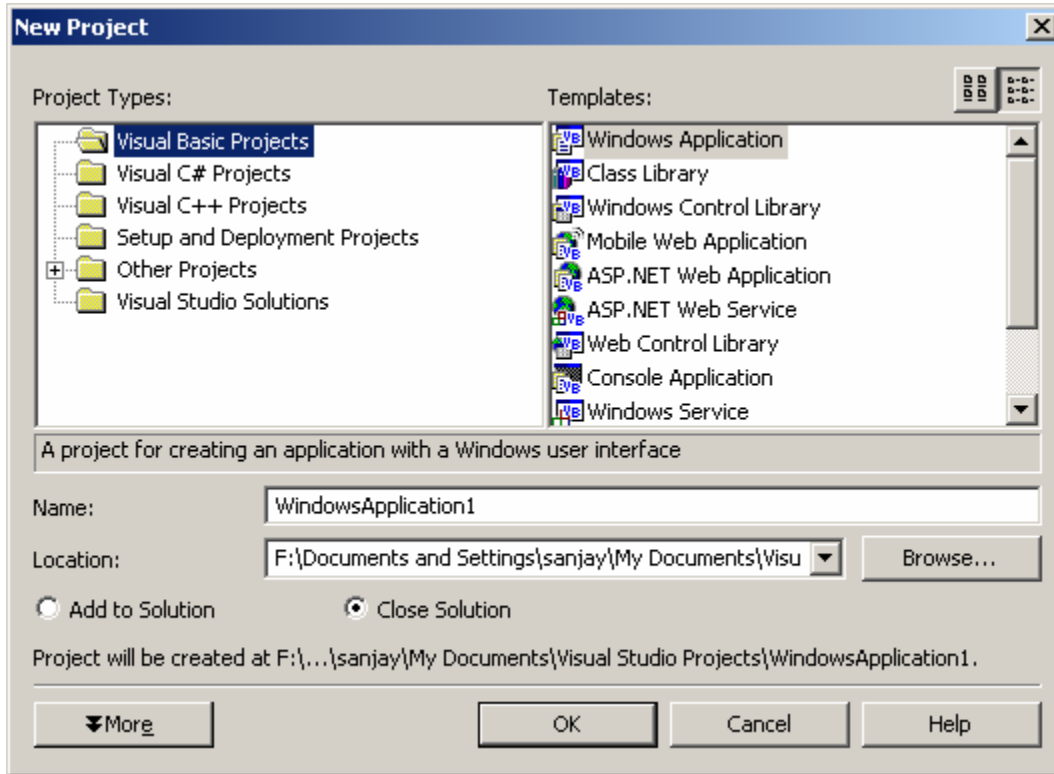


2. Xuất hiện các vùng làm việc sau:

- Start page
- Dynamic Help
- Solution Explorer

Exercise 2 – Tạo một Solution và Project file

1. Từ menu **File | New | Project**. Cửa sổ New project xuất hiện như hình dưới đây.



2. Hộp thoại new project mở ra, instructor mô tả các đặc trưng sau.

VB projects, Templates

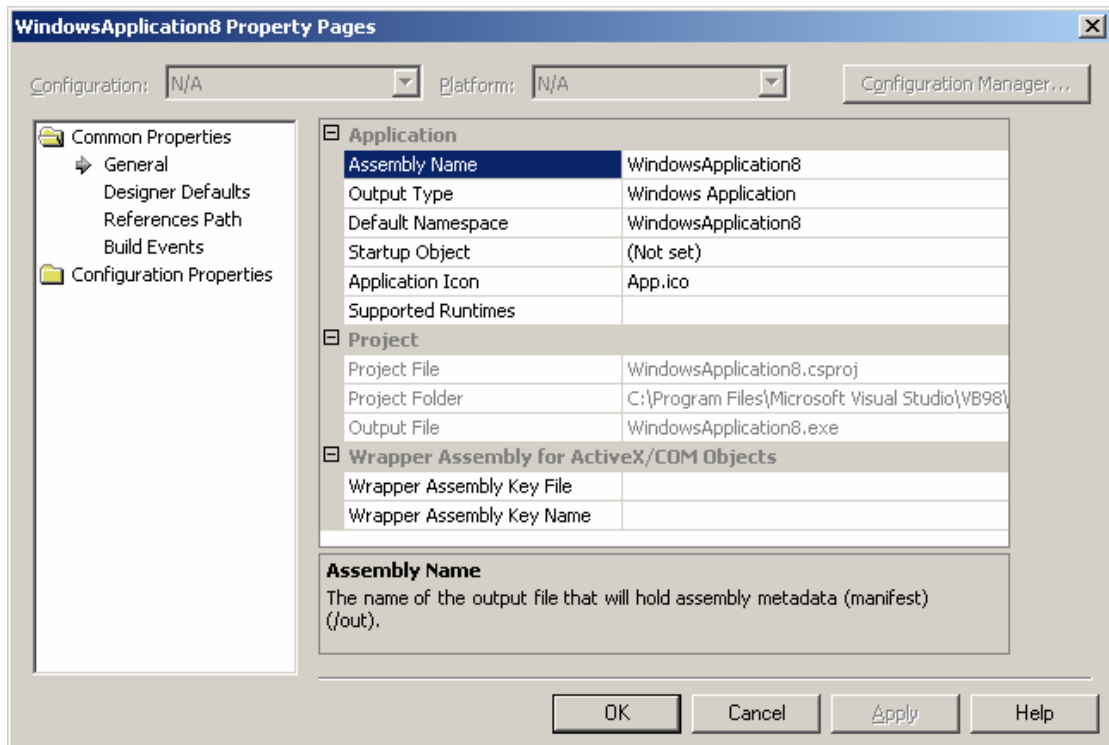
VC Projects, Templates

Một môi trường phát triển tích hợp hỗ trợ các mẫu Enterprise Templates đa ngôn ngữ.

3. Trong hộp thoại **New Project**, ở vùng **Project Types**, click **Visual C#/Visual Basic Projects**. Ở vùng **Templates**, click **Windows Application**, đặt tên của project là “MYFirstDotNetApp” sau đó click OK để tạo một project mới.
4. Instructor đưa ra các đặc tính sau:
 - Command window
 - Task List
 - Enhanced IntelliSense
 - Enhanced Toolbox

Exercise 3 – Thiết lập các properties cho project

Click chuột phải vào tên của project trong **Solutions Explorer** ở góc phải của màn hình và chọn **Properties**. Với một **C# Project**.



Tại đó, bạn có thể thay đổi các thông tin sau:

- Common properties
- Configuration Properties
- Configuration Manager\

II. Làm quen với ngôn ngữ C#

Exercise 1: Tạo chương trình C# đầu tiên.

Hướng dẫn cách tạo chương trình HelloWorld sử dụng ngôn ngữ C# và môi trường phát triển MS VS.NET

- Tạo một empty project mới đặt tên là HelloWorld và lưu nó vào 1 vị trí tùy ý.
- Viết một chương trình để hiển thị một chuỗi ký tự lên màn hình

```
using System;

class Hello
{
    static void Main()
    {
        System.Console.WriteLine("Hello Character World");
        System.Console.ReadLine();
    }
}
```

- Dịch và chạy chương trình

- o Build chương trình bằng cách **Build -> Build Solution**. Hoặc sử dụng phím tắt Ctrl+Shift+B.
- o Chạy chương trình sử dụng **Debug->Start Without Debugging**. Hoặc có thể sử dụng phím tắt Ctrl+F5.
- o Kết quả hiển thị ra màn hình như sau:

```
Hello Character World
Press any key to continue
```

Exercise 2: Common C# Value Types.

Giới thiệu các loại dữ liệu cơ bản, cách chuyển đổi giữa các loại dữ liệu và các toán tử string cơ bản.

- Thực hiện theo hướng dẫn sau:

1. Tạo một empty project Windows Application.	a. Thực hiện như các bước đã nêu trong phần đầu, và đặt tên project tùy ý.
2. Tạo một biến kiểu byte , xác định giá trị lớn nhất của kiểu đó, chuyển giá trị đó sang dạng xâu, rồi hiển thị lên màn hình.	b. Tại hàm Main, gõ lệnh sau: <code>byte b;</code> c. Đặt giá trị cho b là giá trị lớn nhất của kiểu này như sau: <code>b = byte.MaxValue;</code> d. Chuyển sang xâu bằng cách sử dụng phương thức tĩnh <code>byte.ToString()</code> . Hiển thị kết quả ra console: <code>Console.WriteLine("Maximum Byte is " + b.ToString());</code>

- Tương tự như thế để thực hiện với các kiểu biến 32-bit loại **int**, biến 64-bit loại **long**, ...
- Tạo 2 xâu, sau đó kết nối chúng sử dụng toán tử “+” – tạo ra xâu kết quả. Hiển thị xâu kết quả đó.
- Xác định độ dài của một xâu sử dụng property **String.Length**. Gọi phương thức **String.Substring** để cắt ra một vài ký tự ở giữa của một xâu bất kỳ. Hiển thị kết quả của cả 2 thao tác đó.
- Tạo một object **DateTime**, khởi tạo là ngày hôm nay, sau đó hiển thị ra màn hình.

Exercise 3: Containers.

Bài tập này giới thiệu 3 loại containers (collection classes) fixed-length arrays, variable-length arrays, and queues – available to .NET programmers.

Mục đích

- Viết code để allocate, initialize, và truy nhập dữ liệu trong mảng fixed-length.
- Viết code để allocate, initialize, và truy nhập dữ liệu trong mảng variable-length.
- Viết code để allocate, initialize, và truy nhập dữ liệu trong một queue.

Steps

Mục tiêu 1 – Write code to allocate, initialize, and access data in fixed-length arrays.

- Tạo một C# project Console Application và đặt tên tùy ý.
- Tạo và gán giá trị cho một mảng 5 số nguyên.

```
int[] i = new int[5] {10, 20, 30, 40, 50};
int j;

for (j = 0; j < 5; j++)
{
    Console.WriteLine("Index= " + j.ToString() +
                      " & Value = " + i[j].ToString());
}
```

- Hiển thị thông tin lên màn hình

```
Index= 0 & Value = 10
Index= 1 & Value = 20
Index= 2 & Value = 30
Index= 3 & Value = 40
Index= 4 & Value = 50
```

- Tiếp đó, tạo và thiết lập giá trị cho một mảng gồm 3 xâu.

```
string[] str = new string[3] {"One", "Two", "Three" };
int iStr;

for (iStr = 0; iStr < 3; iStr++)
{
    Console.WriteLine("Index= " + iStr.ToString() +
                      " & Value = " + str[iStr]);
}
```

- Chạy chương trình đó để xem kết quả..
- Sau đó, tạo và thiết lập giá trị cho một mảng gồm 2 dates.

```
DateTime[] dt = new DateTime[2]
{
    new DateTime(2002,5,1), new DateTime(2002, 6,1)
};
int iDate;
```

```

for (iDate = 0; iDate < 2; iDate++)
{
    Console.WriteLine("Index= " + iDate.ToString() +
        " & Date = " + dt[iDate].ToShortDateString());
}

```

- Chạy chương trình đó để xem kết quả..

Mục tiêu 2 – Write code to allocate, initialize, and access data in fvariable-length arrays.

Tạo một ArrayList và thêm 3 phần tử thuộc 3 kiểu khác nhau: string, integer và date. Hiển thị nội dung của mảng đó lên màn hình

```

ArrayList al = new ArrayList();

al.Add("Hello");
al.Add(new DateTime(2002,10,23));
al.Add(15);

Console.WriteLine(al[0]);
Console.WriteLine(al[1]);
Console.WriteLine(al[2]);

```

Chú ý

- Mảng có độ dài cố định phải chứa các đối tượng cùng loại. Một mảng ArrayList có thể lưu trữ các đối tượng thuộc các loại khác nhau.

Mục tiêu 3 – Write code to allocate, initialize, and access data in a queue.

Tạo một Queue và thêm 3 phần tử thuộc 3 loại khác nhau. Hiển thị nội dung lên màn hình.

```

Queue q = new Queue();
q.Enqueue("Compact Framework");
q.Enqueue(new Decimal(123.456) );
q.Enqueue(654.321);

Console.WriteLine(q.Dequeue());
Console.WriteLine(q.Dequeue());
Console.WriteLine(q.Dequeue());

```

Chú ý:

- Một .NET Queue có thể lưu trữ bất kỳ kiểu đối tượng nào.
- Việc thêm phần tử cho một queue được thực hiện bằng cách gọi Queue.Enqueue.
- Xóa phần tử khỏi một queue bằng cách gọi Queue.Dequeue.

Exercise 3: Định nghĩa lớp.

Giới thiệu một số đặc tính của lớp trong C#

Mục tiêu 1: *Mô tả sự khác biệt giữa phương thức tĩnh (static method) và phương thức thể hiện (instance method)*

- Tạo một project mới.
- Tạo một lớp bất kỳ A chứa 1 static method và 1 instance method.

```
class A
{
    static public void Static_Member()
    {
        Console.WriteLine("Calling from Static_Member");
    }
    public void Instance_Member()
    {
        Console.WriteLine("Calling from Instance_Member");
    }
}
```

- Trong hàm Main(), sử dụng như sau.

```
A myA = new A();
A.Static_Member();    // Call static methods with class name.
myA.Instance_Member(); // Use instance to call instance method.
```

- Dịch và chạy project đó để xem kết quả.

Mục tiêu 2: *Thừa kế lớp có sẵn*

- Tạo một lớp B thừa kế từ lớp A, tạo 3 biến private trong lớp đó

```
class B : A
{
    private int m_x;
    private int m_y;
    private int m_z;
}
```

- Tạo một instance của B, gọi hàm thành viên của nó và cố truy cập các biến private xem sao?

Mục tiêu 3: Thêm các properties: read-only properties, write-only properties, và read/write properties

- Chỉnh sửa lớp B để thêm vào read-only property (X), write-only property (Y), và read/write property (Z).

```
public int X
{
    get { return m_x; }
}

public int Y
{
    set { m_y = value; }
}

public int Z
{
    get { return m_z; }
    set { m_z = value; }
}
```

- Viết code để sử dụng các properties này.