MICROSOFT .NET (C#)

PROFESSIONAL PROGRAMMING FOR REAL LIFE

ITGATEVN V 2.0
Speed of Technology
Trung tâm đào tạo CNTT chất lượng cao
CỔNG CNTT VIỆT NAM. WWW.ITGATEVN.COM.VN

LỜI NGỎ

Toàn tập giáo trình này là kết quả của sự cô đọng những kiến thức cần thiết giúp bạn làm chủ nền tảng .NET cho công việc. Chúng tôi đã đúc kết những kinh nghiệm thực tế, rút gọn những điểm cần lưu ý, những kỹ thuật mà hầu hết các công ty phần mềm phát triển ứng dụng trên nền tảng .NET đều yêu cầu thực hiện. Hơn nữa đây còn là tài liệu mà bạn có thể sử dụng để ôn tập lại những kiến thức sau những giờ lên lớp và đặc biệt trong công việc của các bạn.

Xuất thân từ một nhà phát triển, chuyên gia phân tích hệ thống, tôi đã gói những kiến thức cốt lõi và cần thiết nhất trong lập trình trên công nghệ .NET vào tài liệu này. Những ví dụ trong tài liệu là những ví dụ thực tế được tôi trích lọc từ những dự án mà tôi và đồng sự đã thực hiện trong suốt thời gian tham gia làm việc tài nhiều công ty phần mềm lớn, đó là các tình huống cụ thể mà chúng tôi gặp phải, và giờ đây tôi tổng kết lại để giới thiệu đến các bạn như một sự chia sẻ kinh nghiệm.

Sau thời gian làm việc và hoạt động trên mạng thông tin Việt Nam - www.itgatevn.com.vn - đã có nhiều người bạn làm việc tại các công ty tin học trong nước cũng như các bạn bè của tôi trên mạng gởi email, hỏi đáp và yêu cầu tôi viết một tài liệu đầy đủ về lập trình .NET, chia sẻ những kinh nghiệm thực tế trong công việc để mọi người cùng tham khảo, phải ngắn gọn và thật sự thực tế nhưng lại phải chuyên sâu và thể hiện những kỹ thuật cao trong lập trình. Tôi đã nghĩ về những yêu cầu đó, và quyết định viết tài liệu này vào mỗi buổi tối sau giờ làm việc. Đây như một món quà đáp lại sự tín nhiệm, yêu mến mà các bạn đã, đang và sẽ dành cho tôi.

Tôi rất mong nhận được nhiều ý kiến đóng góp của các bạn cho tài liệu này, và tôi sẽ tổng hợp lại để cập nhật cho tài liệu một tốt hơn.

Tôi mong rằng tài liệu này sẽ đồng hành cùng các bạn trong công việc.

Tác giả

Phạm Tuấn Anh

BẢN QUYỀN TÁC GIẢ

Tác giả

Phạm Tuấn Anh

MỤC LỤC

LẬP TRÌNH .NET (C#)

.NET là nền tảng cho phép phát triển những ứng dụng mới hoàn toàn trên cả hai môi trường Win và Web. Khi sử dụng .NET, đòi hỏi phải sử dụng một ngôn ngữ để khai thác hết sức mạnh của nó. C# là ngôn ngữ chúng tôi lựa chọn để sử dụng và giới thiệu đến bạn. C# được phát triển từ C/C++ và giữ nguyên tên trong gia đình C, ký tự # được sử dụng như một sự khẳng định về tính sắc bén của ngôn ngữ này, do đó C# được phát âm là **C sharp**

Cấu trúc lập trình C# căn bản

Ứng dụng "C# Hello World"

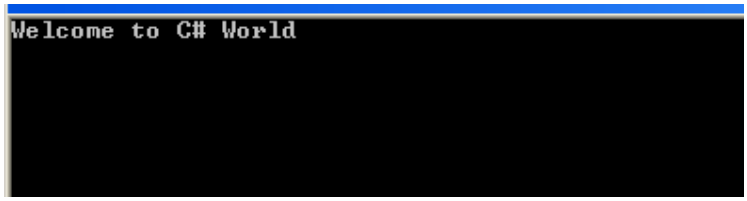
Hellow World là chương trình đầu tiên để mở đầu cho việc học một ngôn ngữ lập trình nào đó, với C# cũng thế, hãy bắt đầu với "C# Hello World"

Ví dụ 1:

Sau đây là chương trình C# Hello World, mã nguồn như sau:

```
/*This is Hellow World C# Program*/
using System;
using System.Collections.Generic;
using System.Text;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to C# World");
        }
    }
}
```

Kết quả xuất hiện của chương trình như sau



Tiếp cận C#

Những chủ đề chính

Tại phần này, bạn sẽ học:

- Biến trong lập trình C#.
- Kiểu dữ liệu
- Cấu trúc điều kiện
- Cấu trúc vòng lặp
- Mảng trong C#

Khai báo biến trong C#

Các biến trong C# được khai báo theo công thức như sau:

AccessModifier DataType **VariableName**;

Trong đó,

AccessModifier: xác định ưu tiên truy xuất tới biến

Datatype: định nghĩa kiểu lưu trữ dữ liệu của biến

VariableName: là tên biến

Cấp độ truy xuất tới biến được mô tả như bảng dưới đây

Access Modifier	Mô tả
public	Truy cập tại bất kỳ nơi đâu
protected	Cho phép truy xuất bên trong một lớp nơi biến này được định nghĩa, hoặc từ các lớp con của lớp đó.
private	Chỉ truy xuất ở bên trong lớp nơi mà biến được định nghĩa.

Kiểu dữ liệu trong C#

Các kiểu dữ liệu được sử dụng trong ngôn ngữ C# được mô tả theo bảng dưới đây

C# Data Type	Mô tả	Ví dụ
object	kiểu dữ liệu cơ bản của tất cả các kiểu khác	`object obj = null;`
string	Được sử dụng để lưu trữ những giá trị kiểu chữ cho biến	`string str = "Welcome";`
int	Sử dụng để lưu trữ giá trị kiểu số nguyên	`int ival = 12;`
byte	Sử dụng để lưu trữ giá byte	`byte val = 12;`
float	Sử dụng để lưu trữ giá trị số thực	`float val = 1.23F;`
bool	Cho phép một biến lưu trữ giá trị đúng hoặc sai	`bool val1 = false;` `bool val2 = true;`
char	Cho phép một biến lưu trữ một ký tự	`char cval = 'a';`

Input/Output trong C# căn bản

Input /output trong C# được thực hiện thông qua việc sử dụng hàm của lớp *Console* trong namespace *System*.

Hai hàm thường sử dụng nhất cho thao tác Input/Output là:

```
Console.WriteLine();
Console.ReadLine();
```

Trong đó,

Console.WriteLine(): được sử dụng để xuất hiện kết quả

Console.ReadLine(): được sử dụng để đọc kết quả nhận vào.

Xem tại mã nguồn HelloWorld kèm theo tại liệu này

Cấu trúc điều khiển trong lập trình C#

C# cung cấp hai cấu trúc điều khiển thực hiện việc lựa chọn điều kiện thực thi chương trình

Cấu trúc if

Cấu trúc if trong C# được mô tả như sau:

```
if (biểu thức điều kiện)
{
     // câu lệnh thực thi nếu biểu thức điều kiện đúng
}
[else
{
     // câu lệnh thực thi nếu biểu thức điều kiện sai
}]
```

Ví dụ:

```
if (20 % 4 > 0)
        {
            Console.WriteLine("Số 20 không chia hết cho 4");
        }
        else
        {
            Console.WriteLine("Số 20 chia hết cho số 4");
        }
```

Cấu trúc switch … case

Cấu trúc swtich….case có cấu trúc như sau:

```
// switch ... case
switch (variable)
{
      case value:
            Câu lệnh thực thi
            break;
      case value:
            Câu lệnh thực thi
            break;
      case value:
            Câu lệnh thực thi
            break;
      default:
            Câu lệnh thực thi
      break;
}
```

Ví dụ:

```
int x = 20 % 4;
        switch (x)
        {
            case 1:
                Console.WriteLine("20 chia cho 4 được số dư là 1");
                break;
            case 0:
                Console.WriteLine("20 chia hết cho 4");
                break;
            default:
                Console.WriteLine("Không thuộc tất cả các trường hợp trên");
                break;
        }
```

Cấu trúc vòng lặp trong lập trình C#

C# cung cấp các cấu trúc vòng lặp chương trình

- While
- Do… while
- For
- Foreach

Sau đây, tôi xin giới thiệu công thức và ví dụ sử dụng các vòn lặp trên

- **Vòng lặp *While***

  Cấu trúc vòng lặp while

  ```
  while    (condition)
  {
           // câu lệnh
  }
  ```

  Thực thi câu lệnh hoặc một loạt những câu lệnh đến khi điều kiện không được thỏa mãn.

Ví dụ:

```
using System;
class WhileTest
{
   public static void Main()
   {
      int n = 1;

      while (n < 6)
      {
         Console.WriteLine("Current value of n is {0}", n);
         n++;
      }
   }
```

```
}
```

- **Vòng lặp *do***

  Cấu trúc vòng lặp while

  ```
  do
  {
          // câu lệnh
  }
  white    (condition)
  ```

  Thực thi câu lệnh ***ít nhất một lần*** đến khi điều kiện không được thỏa mãn.

Ví dụ:

```
using System;
public class TestDoWhile
{
   public static void Main ()
   {
      int x;
      int y = 0;

      do
      {
         x = y++;
         Console.WriteLine(x);
      }

      while(y < 5);
   }
}
```

- **Vòng lặp *for***

  Cấu trúc vòng lặp for

  ```
  for (initialization; condition; increment / decrement)
          {
              // thực thi câu lệnh
          }
  ```

Ví dụ:

```
using System;
public class ForLoopTest
{
    public static void Main()
    {
            for (int i = 1; i <= 5; i++)
                Console.WriteLine(i);
    }
}
```

- **Vòng lặp *foreach***

  Cấu trúc vòng lặp *foreach*

```
for (initialization; condition; increment / decrement)
        {
            // thực thi câu lệnh
       }
```

Arrays - Mảng trong C#

Mảng là một nhóm những biến có cùng một kiểu dữ liệu. Những biến này được lưu trữ trong bộ những vùng bộ nhớ kế tiếp do đó mảng cho phép truy xuất và thực thi đến từng phần tử trong mảng.

Công thức khai báo một mảng như sau:

Datatype [] variableName = new *Datatype* [number of elements];

Trong đó,

number of elements: là số phần tử của mảng

Datatype: kiểu dữ liệu mà mảng lưu trữ

variableName: là tên mảng.

Ví dụ:

```
// mảng kiểu int
int[] iarray = new int[5];
// mảng kiểu string
string[] sarray = new string[6];
```

Ví dụ: cách khai báo khác

```
string[] sarray2 = { "Welcome", "to", "C# Array" };
```

Khi lập trình, tùy theo điều kiện chương trình mà bạn có thể chọn lựa một trong hai cách trên.

Chúng ta đã học

Tới đây, bạn đã tìm hiểu và làm quen với lập trình trên nền tảng .NET với ngôn ngữ C#. Những kiến thức sau bạn cần nắm vững:

- C# là một ngôn ngữ mạnh, được biên dịch và thực thi dựa trên nền tảng .NET của Microsoft.
- Những kiểu dữ liệu cơ bản trong C#, cách khai báo biến, mảng trong C#
- Cấu trúc điều kiện, lựa chọn if … else và switch… case
- Cấu trúc vòng lặp while, do…while, for, foreach

Bài tập tự thực hiện

Để củng cố kiến thức đã học, Những bài tập sau đây được yêu cầu thực hiện

1. Viết chương trình cho phép nhật vào 1 số nguyên dương N, và hiển thị giá trị từ 0 đến N ra màn hình
2. Viết chương trình máy tính cá nhân cho phép nhập vào hai số và thực hiện tính toán: nhân, chia, cộng, trừ, lỹ thừa
3. Viết chương trình giải bài toán phương trình bậc hai: $aX^2 + bX + c = 0$ với a,b,c là các tham số

Hiện thực khái niệm hướng đối tượng (OOP) trong C#

Chúng ta sẽ học

Những nội dung trong phần này tổng kết hóa những điểm quan trọng nhất về khái niệm OOP trong C#. Nội dung bao gồm

- Định nghĩa lớp, đối tượng
- Hàm tạo (Constructor), hàm hủy (Destructor)
- Function Overloading.
- Thừa kế trong lập trình C#.
- Overriding Method.
- Polymorphism
- Abstract Class trong C#
- Namespaces
- Enumerators

Lớp (class) trong C#

Class

Một Class là một khái niệm mô tả cho những thực thể có chung tính chất và hành vi. Class định nghĩa những thuộc tính và hành vi được dùng cho những đối tượng của lớp đó. Do đó có thể nói Class là một khuôn mẫu cho các đối tượng.

Công thức để tạo một class

```
AccessModifier  class className
{
      // thân class
}
```

Đối tượng (Objects)

Đối tượng là một đại diện, hay có thể nói là một sản phẩm của một class. Tất cả các đối tượng đều có chung những thuộc tính và hành vi mà class định nghĩa. Cách tạo đối tượng giống như cách tạo một biến có kiểu dữ liệu là Class.

```
AccessModifier  ClassName   ObjectName = new ClassName();
```

Ưu điểm của việc sử dụng Class và Đối tượng

Có một số những ưu điểm của việc sử dụng Class và đối tượng trong phát triển phần mềm. Những ưu điểm nổi bật nhất được liệt kê như sau:

- Duy trì code bằng việc mô hình hóa
- Đóng gói những sự phức tạp trong mã lệnh từ người dùng
- Khả năng sử dụng lại
- Cung cấp đơn kế thừa để thực thi nhiều phương thức.

Hàm tạo (Constructors) và hàm hủy (Destructors) trong C#

Constructors

Constructors là những hàm đặc biệt cho phép thực thi, điều khiển chương trình ngay khi khởi tạo đối tượng. Trong C#, Constructors có tên giống như tên của Class và không trả lại giá trị.

Ví dụ

```
class Library
{
      private int ibooktypes;
      //Constructor
      public Library()
      {
            ibooktypes = 7;
      }
      public Library(int value)
          {
                ibooktypes = value;
```

```
        }
}
```

Destructors

Là một hàm đặc biệt được sử dụng để làm sạch bộ nhớ. Cách khai báo giống như Constructor nhưng không có tham số và được bắt đầu bằng dấu "~".

Ví dụ

```
class Library
{
      private int ibooktypes;
      //Constructor
      public Library()
      {
            ibooktypes = 7;
      }
      public Library(int value)
      {
            ibooktypes = value;
      }
      ~ Library()
      {
            //thực thi câu lệnh
      }
}
```

Fuction Overloading

Method Overloading xuất hiện khi trong một class có từ hai hàm có cùng tên. Có hai kiểu Method Overloading:

- Function Overloading dựa trên số lượng tham số truyền vào
- Function Overloading dựa trên kiểu giá trị tham số truyền vào.

Ví dụ

```
class Library
{
      // Function Overloading
        public void insertbooks(int id)
        {
            //
        }
        public void insertbooks(int id, int type)
        {
            //
        }
        public void insertbooks(string id, int type)
        {
            //
        }
}
```

Ba hàm insertbooks ở trên là một ví dụ về function overloading trong lập trình C#. Trong khi hàm thứ nhất và thứ 2 là overloading theo số lượng tham số, và hàm thứ 3 với hàm thứ 2 là overloading theo kiểu tham số truyền vào.

Thừa kế trong C#

Một trong những ưu điểm nổi bật của lập trình hướng đối tượng đó là thừa kế, đó là sự sử dụng lại những thuộc tính và hành vi của một lớp. Có hai kiểu kế thừa trong lập trình, đơn kế thừa và đa kế thừa.

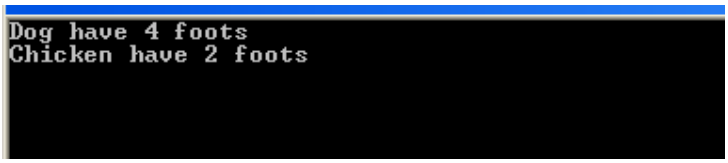
C# cung cấp mô hình đơn kế thừa.

Ví dụ về kế thừa trong C#.

```
/* Ví dụ về thừa kế trong lập trình C# */
using System;
using System.Collections.Generic;
using System.Text;

namespace __OOP_Inheritance
{
    class Program
    {
        static void Main(string[] args)
        {
            Dog objDog = new Dog(4);
            objDog.displayProperties();
            Chicken objChicken = new Chicken(2);
            objChicken.displayProperties();
            Console.Read();
        }
    }
    class Animal
    {
        protected int ifoots;
        protected string sName;

        protected void setFoot(int ival)
        {
            ifoots = ival;
        }
        protected void setName(string sVal)
        {
            sName = sVal;
        }
        public void displayProperties()
        {
            Console.WriteLine(sName + " have " + ifoots.ToString()+ "
foots");
        }
    }
    class Dog : Animal
    {
        public Dog(int ival)
        {
            setName("Dog");
            ifoots = ival;
        }
    }
    class Chicken : Animal
    {
        public Chicken(int ival)
```

```
        {
            setName("Chicken");
            setFoot(ival);
        }
    }
}
```

Kết quả khi thực thi chương trình



Ở ví dụ trên, Dog và Chicken là hai lớp kế thừa từ lớp Animal, do đó các thuộc tính như số chân, ifoots và tên sName đương nhiên xuất hiện trong hai lớp này và cho phép sử dụng. Tương tự, các hàm như *setName()*, *setFoot()*, *displayProperties()* tại lớp Animal cũng được kế thừa xuống hai lớp Dog và Chicken. Do đó ta có thể gọi những hàm này, và kết quả hiển thị khi gọi hàm *displayProperties()* theo đối tượng objDog và objChicken khác nhau như hình trên.

Overriding, Polymorphism trong C#

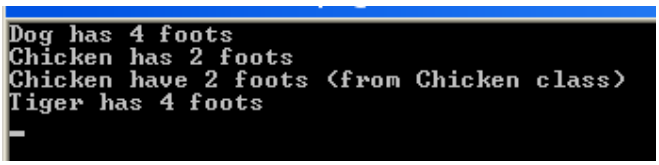
Overriding

Ví dụ

```
/* Ví dụ về thừa kế,overrding trong lập trình C# */
using System;
using System.Collections.Generic;
using System.Text;

namespace __OOP_Inheritance
{
    class Program
    {
        static void Main(string[] args)
        {
            Dog objDog = new Dog(4);
            objDog.displayProperties();
            Chicken objChicken = new Chicken(2);
            objChicken.displayProperties();
            Tiger objTiger = new Tiger(4);
            objTiger.displayProperties();
            Console.Read();
        }
    }
    class Animal
    {
        protected int ifoots;
        protected string sName;

        protected void setFoot(int ival)
        {
            ifoots = ival;
        }
        protected void setName(string sVal)
```

```
        {
            sName = sVal;
        }
        public virtual void displayProperties() // chú ý hàm này
        {
            Console.WriteLine(sName + " has " + ifoots.ToString()+ " foots");
        }
    }
    class Dog : Animal
    {
        public Dog(int ival)
        {
            setName("Dog");
            ifoots = ival;
        }
    }
    class Chicken : Animal
    {
        public Chicken(int ival)
        {
            setName("Chicken");
            setFoot(ival);
        }
        public void displayProperties()
        {
            base.displayProperties();
            Console.WriteLine(sName + " have " + ifoots.ToString() + " foots
(from Chicken class)");
        }
    }

    class Tiger : Animal
    {
        public Tiger(int ival)
        {
            setFoot(ival);
        }
        public override void displayProperties() // chú ý hàm này
        {
            Console.WriteLine("Tiger has " + ifoots.ToString()+ " foots");
        }
    }
}
```

Kết quả thực hiện chương trình



Hàm displayProperties() trong lớp Tiger overrides hàm displayProperties() trong lớp Animal

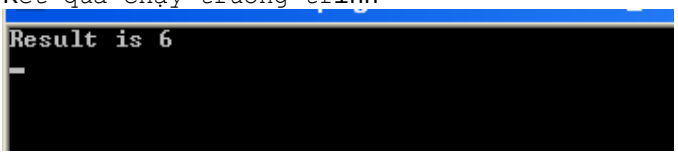
Nếu một hàm được định nghĩa trong lớp con có cùng tên, kiểu với hàm trong lớp cha, khi ấy hàm trong lớp con sẽ overrides (làm ẩn) hàm trong lớp cha. Đó được gọi là overriding.

Polymorphism

Ví dụ

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

namespace __OOP_polymorphism
{
    class Program
    {
        static void Main(string[] args)
        {
            Child objchild = new Child();
            Console.WriteLine("Result is " + objchild.methodA().ToString());
            Console.Read();
        }
    }
    class Parent
    {
        public int methodA()
        {
            return methodB() * methodC();
        }
        public virtual int methodB()
        {
            return 1;
        }
        public int methodC()
        {
            return 2;
        }
    }
    class Child : Parent
    {
        public override int methodB()
        {
            return 3;
        }

    }

}
```

Kết quả chạy trương trình



Như bình thường của mô hình kế thừa, kết quả trả về khi gọi hàm methodA() từ đối tượng của lớp Child phải là "Result is 2". Nhưng trong kết quả trên, kết quả là "Result is 6". Kết quả này do hàm methodB() tại lớp Child đã override hàm methodB() tại lớp Parent.

Vậy ta có thể khái quát Polymorphism như sau:

- Polymorphism không chỉ đơn giản là overriding, mà nó là overrding thông minh.
- Khác biệt giữ Overriding và Polymorphism đó là trong Polymorphism, sự quyết định gọi hàm được thực hiện khi chương trình chạy.

Abstract Class trong C#

Abstract Class là lớp dùng để định nghĩa những thuộc tính và hành vi chung của những lớp khác. Một Abstract class được dùng như một lớp cha của các lớp khác. Từ khóa abstract được dùng để định nghĩa một abstract class. Những lớp được định nghĩa bằng cách dùng từ khóa abstract thì không cho phép khởi tạo đối tượng của lớp ấy.

```
abstract class Shape
{
      public abstract float calculateArea();
      public void displaySomething()
      {
            Console.WriteLine("Something is displayed");
      }
}
class Circle:Shape
{
     float radius;
     public override float calculateArea()
     {
          return radius * 22 / 7;
     }
}
```

Khi thực thi chương trình, bạn không thể tạo đối tượng cho lớp Shape, vì nó là abstract class.

Namespaces

Khái niệm Namespace

Đường mang tên vị tướng danh tiếng Trần Hưng đạo đều có tại Sài Gòn và Hà Nội, vậy làm sao để phân biệt khi người nước ngoài muốn hỏi về đường Trần Hưng Đạo. Cách đơn giản nhất đó là khi muốn gọi tên đường Trần Hưng Đạo tại Hà Nội thì ta gọi "đường Trần Hưng Đạo tại Hà Nội" và tương tự tại Sài Gòn là "đường Trần Hưng Đạo tại Sài Gòn" và chắc chắn chúng ta sẽ có câu trả lời cho vị khách đó.

Hà Nội, Sài Gòn trong ví dụ trên là một ví dụ cho Namespace.

Vậy có thể hiểu Namespace là một gói những thực thể có thuộc tính và hành vi độc lập với bên ngoài. Những ưu điểm của namespace được liệt kê như sau:

- Tránh được sự trùng lặp tên giữa các class.
- Cho phép tổ chức mã nguồn một cách có khoa học và hợp lý.

Khai báo một Namespace

```
namespace NamespaceName
{
```

```
      // nơi chứa đựng tất cả các class
}
```
Trong đó,

Namespace: là từ khóa khai báo một NameSpace
NamespaceName: là tên của một Namespace

Ví dụ

```
namespace CSharpProgram
{
    class Basic
    {
    }
    class Advance
    {
    }
}
```

Enumerator trong C#

Enums là một loạt tên của những hằng số. Được sử dụng để định nghĩa những kiểu dữ liệu có một loạt những giá trị xác định.

Ví dụ sau mô tả về Enumerator

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

namespace __OOP_polymorphism
{
    class Program
    {
        static void Main(string[] args)
        {
            // Enummerator
            EnumDemo eobj = newEnumDemo();
            eobj.getWeekDay(DayinWeek.Saturday);
            Console.Read();
        }
    }

    public enum DayinWeek
    {
        Monday,
        Tuesday,
        Wednesday,
        Thursday,
        Friday,
        Saturday,
        Sunday
    }
    public class EnumDemo
    {
        public void getWeekDay(DayinWeek dayoff)
        {
            Console.WriteLine("My weekday is " + dayoff.ToString());
        }
    }
```

```
}
```

BÀI TẬP CÓ HƯỚNG DẪN

Namespace System.IO

IO Namespace chứa những lớp cho phép thao tác đọc và ghi dữ liệu đến những luồng dữ liệu và file. Một số lớp của System.IO được liệt kê như sau:

- BinaryReader
- Binary Writer
- Stream
- TextReader
- TextWriter
- Directory
- File
- FileSystemInfo

CƠ SỞ DỮ LIỆU

Mục tiêu khóa học này

Sau phần này, bạn có thể

- Hiểu về các hệ cơ sở dữ liệu
- Thao tác dữ liệu với CSDL MSSQL Server bằng các câu lệnh insert, update, delete, select cơ bản
- Truy vấn dữ liệu từ nhiều bảng cùng lúc.

Cơ sở dữ liệu

Cơ sở dữ liệu là tập hợp những dòng dữ liệu được sắp xếp theo một cấu trúc nhất định.

Hệ quản trị cơ sở dữ liệu

Là phần mềm được sử dụng để quản trị các cơ sở dữ liệu.

Bảng (Table)

Bảng có thể được hiểu là nơi thực sự chứa dữ liệu ở mức độ vật lý. Bảng là tập hợp những dòng dữ liệu có cùng cấu trúc được định nghĩa bởi những cột (Column), mỗi dòng dữ liệu trong bảng chứa những kiểu dữ liệu được qui định bởi những cột của bảng.

Thao tác với hệ quản trị dữ liệu MSSQL Server

Chúng ta sẽ cùng đi thẳng vào vấn đề thao tác với một hệ cơ sở dữ liệu sử dụng mã lệnh chương trình.

Khởi tạo một hệ cơ sở dữ liệu

Cấu trúc câu lệnh đơn giản để khởi tạo một hệ cơ sở dữ liệu như sau:

```
CREATE DATABASE database_name
```

Trong đó

- CREATE DATABASE là từ khóa tạo cơ sở dữ liệu
- *database_name* là tên cơ sở dữ liệu.

Ví dụ: tạo cơ sở dữ liệu để học tập .NET có tên _NETDB

```
CREATE DATABASE _NETDB
```

Tạo bảng

Câu trúc câu lệnh đơn giản để khởi tạo một bảng trong một CSDL như sau

```
CREATE TABLE
(
      Tên cột 1          Kiểu dữ liệu,
      Tên cột 2          Kiểu dữ liệu,
      ……                 …..
)
```

Ví dụ: tạo bảng MyTable trong cơ sở dữ liệu _NETDB

```
CREATE TABLE MyTable
(
          id int PRIMARY KEY,
          vName   varchar(150),
          dbirthday datetime,
          igender tinyint
)
```

Truy vấn dữ liệu từ một bảng

Câu lệnh truy vấn dữ liệu (SELECT) trả về dữ liệu có trong bảng. Cấu trúc câu lệnh đơn giản để truy vấn dữ liệu từ một bảng như sau:

SELECT * | [Cột 1], [Cột 2], … FROM [Table Name]

Ví dụ:

```
SELECT * FROM MyTable

SELECT vname FROM MyTable.
```

Truy vấn dữ liệu có điều kiện

Như câu lệnh truy vấn dữ liệu, nhưng câu truy vấn dữ liệu từ bảng cho phép chúng ta chỉ lấy những dữ liệu cần thiết theo một điều kiện nào đó

Cấu trúc câu truy vấn dữ liệu có điều kiện như sau:

SELECT * | [Cột 1], [Cột 2], … FROM [Table Name]

WHERE [Điều kiện 1] AND | OR [Điều kiện 2] ….

Ví dụ:

```
SELECT * FROM MyTable where vName = 'Phạm Tuấn Anh'
Select * from MyTable where igender = 1
```

Truy vấn dữ liệu từ nhiều bảng

MSSQL cung cấp cho chúng ta khả năng truy vấn dữ liệu cùng lúc từ nhiều bảng khác nhau có chung điều kiện. Để truy vấn dữ liệu từ nhiều bảng chúng ta sẽ sử dụng từ khóa JOIN.

Ví dụ: Chúng ta có hai bảng dữ liệu như sau

Category:

icid	vname
1	A
2	B
3	C
4	D

Items

iid	icid	vvalue
1	1	V1
2	1	V2
3	2	V1
4	3	V1
5	5	V1

Chúng ta có thể thấy hai bảng này đều có trường **icid**, và chúng ta có thể lấy dữ liệu từ hai bảng dựa vào việc so sánh dữ giá trị trong trường **icid** ở hai bảng.

Câu lệnh như sau:

```
select
c.*,
i.iid,
vvalue
from category c
JOIN
Items i
ON c.icid = i.icid
```

Kết quả đạt được như bảng sau

1	A	1	V1
1	A	2	V2
2	B	3	V1
3	C	4	V1

Chúng ta có thể thấy trong bảng items có dòng dữ liệu với icid = 5, và dòng này không thỏa điều kiện so sánh (không tồn tại trong bảng category) do đó bị loại khỏi kết quả.

Chúng ta tiếp tục thực hiện câu truy vấn thứ hai.

```
select
c.*,
i.iid,
vvalue
from category c
LEFT JOIN  items i
ON c.icid = i.icid
```

Kết quả đạt được là

icid	vname	iid	vvalue
1	A	1	V1
1	A	2	V2
2	B	3	V1
3	C	4	V1
4	D	NULL	NULL

Chúng ta có được 5 dòng kết quả với dòng thứ 5 có giá trị là NULL, như thế ở câu lệnh trên, việc so sánh chỉ dựa trên các giá trị của cột **icid** trên bảng **category** hay nói cách khác câu truy vấn trên lấy tất cả các dòng của bảng category (bảng bên trái). Người ta định nghĩa đó là LEFT JOIN

Ngược với LEFT JOIN đó là RIGHT JOIN. Chúng ta cùng xét ví dụ

```
select
c.*,
i.iid,
vvalue
from  category c
RIGHT JOIN  items i
ON c.icid = i.icid
```

Kết quả đạt được ngược lại so với khi dùng LEFT JOIN như sau.

icid	vname	iid	vvalue
1	A	1	V1
1	A	2	V2
2	B	3	V1
3	C	4	V1
NULL	NULL	5	V1

Thêm dữ liệu vào bảng

Cấu trúc câu lệnh để thêm dữ liệu vào bảng như sau

INSERT [INTO] TableName [(*column_list*)] VALUES *data_values*

Ví dụ

```
insert into category values('A')
insert into items values(1,'V1')
```

Trong hai câu lệnh trên, chúng ta có thể thấy, không có dữ liệu thêm cho cột mã chính, điều này xảy ra khi cột mã chính là cột tự tăng, khi ấy hệ thống tự gán giá trị ứng với dòng được thêm.

Cập nhật dữ liệu trong bảng

Cấu trúc câu lệnh để cập nhật dữ liệu trong bảng như sau

UPDATE TableName SET *[Cột 1]*=[Giá trị 1], *[Cột 2]*=[Giá trị 2],… [WHERE [ĐIỀU KIỆN]]

Ví dụ

```
update items set vvalue='V5' where iid=5
```

Lưu ý, khi sử dụng câu lệnh UPDATE, nên sử dụng điều kiện phía sau để đảm bảo rằng chỉ có những dòng thỏa điều kiện mới được cập nhật.

Xóa dữ liệu từ bảng

Cấu trúc câu lệnh để cập nhật dữ liệu trong bảng như sau

DELETE TableName [WHERE [ĐIỀU KIỆN]]

Ví dụ

```
DELETE items where iid=5
```

Lưu ý, khi sử dụng câu lệnh DELETE, nên sử dụng điều kiện phía sau để đảm bảo rằng chỉ có những dòng thỏa điều kiện mới được xóa.

LẬP TRÌNH DÀNH CHO CÔNG VIỆC

Mục tiêu khóa học này

Sau phần này, bạn có thể

- Thao tác trên công cụ phát triển Visual Studio .NET (IDE)
- Sử dụng những điều khiển cơ bản của ứng dụng Window
- Sử dụng các lớp của lớp CommonDialog
- Gia tăng tiện ích cho ứng dụng bằng thanh thực đơn
- Tạo hệ thống ứng dụng với MDI Form.
- Xây dựng website, hệ thống ứng dụng trên nền tảng web.
- Thực hiện ứng dụng phân tán với Webservices.
- Thực hiện hệ thống ứng dụng theo yêu cầu của doanh nghiệp.

ADO.NET và thao tác với cơ sở dữ liệu

Sau khi hoàn thành, bạn có thể

- Nắm vững kiến trúc ADO.NET
- Hiểu rõ những đối tượng thuộc ADO.NET
- Kết nối cơ sở dữ liệu sử dụng các hàm thuộc thư viện OleDb
- Thực hiện thêm, cập nhật, xóa dữ liệu

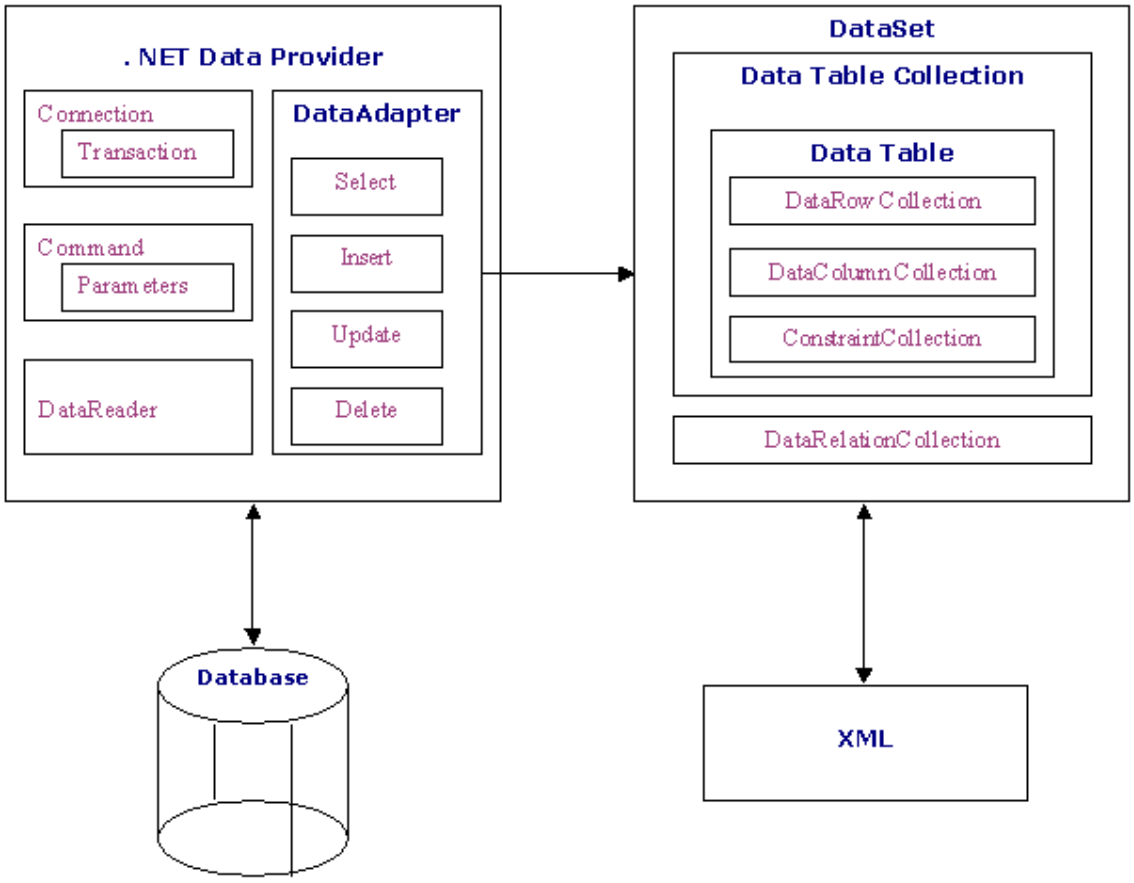
Giới thiệu về ADO.NET

Ngày nay, những ứng dụng thương mại cần thiết để quản lý khối lượng dữ liệu khổng lồ. Dữ liệu thường được lưu trữ trong những bảng quan hệ trong cơ sở dữ liệu. Nhận dữ liệu và thao tác dữ liệu trực tiếp từ một cơ sở dữ liệu đòi hỏi kiến thức về những câu lệnh thao tác trên cơ sở dữ liệu để truy cập đến dữ liệu. Và ứng dụng cần giao tiếp với Cơ sở dữ liệu để thực hiện những công việc sau:

- Nhận dữ liệu được lưu trữ trong cơ sở dữ liệu và hiển thị chúng ra giao diện người dùng.
- Cập nhật dữ liệu, thực hiện thêm, hiệu chỉnh, và xóa dữ liệu.

ADO.NET là một mô hình được những ứng dụng .NET sử dụng để giao tiếp với cơ sở dữ liệu cho việc nhận, truy cập, và cập nhật dữ liệu.

Mô hình ADO.NET



ADO .NET Data Architecture

Data Provider

Một data provider được sử dụng cho việc kết nối đến cơ sở dữ liệu, nhận, lưu trữ dữ liệu trong dataset, đọc, nhận và cập nhật dữ liệu trong cơ sở dữ liệu.

Có hai loại data provider:

- Ole DB data provider - loại này làm việc với tất cả OleDB Provider như Sql OleDB Provider, Oracle OleDb provider, và Jet OleDb Provider. Được biết đến trong môi trường .NET với namespace System.Data.OleDb.
- Sql Server data provider - loại này làm việc chỉ với Microsoft SQL Server. Cho phép thao tác với tốc độ tối ưu với hệ cơ sở dữ liệu MSSQL Server, được biết đến với namespace System.Data.SqlClient.

Kết nối

Kết nối là một component được sử dụng để thiết lập một kết nối đến cơ sở dữ liệu từ một data source. Có hai kiểu đối tượng kết nối thường sử dụng nhất là OleDbConnection và SqlConnection. Bảng sau đây hiển thị những thuộc tính và phương thức hay sử dụng nhất của một đối tượng kết nối.

Tên	Mô tả
ConnectionString	Cung cấp thông tin như datasource, tên cơ sở dữ liệu, được sử dụng để thiết lập kết nối với một cơ sở dữ liệu
Open()	Mở một kết nói với datasource được khai báo tại ConnectionString
Close()	Được sử dụng để đóng kết nối với data source
State	Được sử dụng để kiểm tra trạng thái của một kết nối. 0: kết nối đang đóng, 1: kết nối đang mở.

Ví dụ về việc sử dụng đối tượng kết nối

```
string connectionstring       = "PROVIDER=SQLOLEDB;
server=(local);uid=_net;pwd=;database=_NET";
            OleDbConnection conObj = new OleDbConnection(connectionstring);
            conObj.Open();
```

Data Adapter

Data Adapter là thành phần của ADO.NET, có tác dụng chuyển tiếp dữ liệu từ và đến cơ sở dữ liệu. data Adapter nhận dữ liệu từ cở sở dữ liệu vào một DataSet. Khi bạn thay đổi dữ liệu trong dataset, cũng là thay đổi trong cơ sở dữ liệu bởi dataadapter. Có hai kiểu data adapter thường dùng để cấu hình kết nối đến cơ sở dữ liệu trong Visual Studio .NET:

- SqlDataAdapter - làm việc chỉ với hệ cơ sở dữ liệu MS SQL Server
- OleDbDataAdapter - kiểu này được cấu hình để làm việc với hầu hết các hệ cơ sở dữ liệu được hỗ trợ bởi OleDb data provider.

Data Adapter sử dụng đối tượng kết nối OleDbConnection và SqlConnection được cung cấp bởi data provider để giao tiếp với cơ sở dữ liệu.

Thuộc tính và phương thức của Data Adapter
Data Adapter giao tiếp với cơ sở dữ liệu trong khi nhận, thêm mới, xóa và cập nhật dữ liệu. Những thuộc tính sau đây được thiết lập để thực hiện những tác vụ khác nhau trên một hệ cơ sở dữ liệu

- SelectCommand - nhận dữ liệu từ CSDL thông qua một câu truy vấn hoặc stored procedures
- InsertCommand - thêm dữ liệu vào CSDL qua câu lệnh insert.
- UpdateCommand - cập nhật cơ sở dữ liệu với câu lệnh update
- DeleteCommand - xóa dữ liệu khỏi cơ sở dữ liệu thông qua câu lệnh delete
- Fill() - là phương thức đẩy dữ liệu từ CSDL vào một dataset.
- Update() - là phương thức thực thi InsertComand, Update Command, hoặc DeleteCommand cho mỗi câu lệnh thêm, hiệu chỉnh hoặc xóa dòng để thực hiện thay đổi trong CSDL.

Data Command

DataCommand là đối tượng thực thi những câu lệnh SQL hoặc stored procedure được sử dụng để thao tác với CSDL. Data Command là đối tượng của lớp SqlCommand và OleDbCommand.

Ví dụ sử dụng Data Command

```
string connectionstring       = "PROVIDER=SQLOLEDB;
server=(local);uid=_net;pwd=;database=_NET";
            OleDbConnection conObj = new OleDbConnection(connectionstring);
            conObj.Open();
            OleDbCommand cmd = new OleDbCommand("delete from books where
ibookid='003'",conObj);
            int result = cmd.ExecuteNonQuery();
            conObj.Close();
```

Data Reader

Data reader được sử dụng để chỉ nhận dữ liệu từ một data source. Sử dụng data reader nhận được kết quả nhanh hơn và không tốn bộ nhớ tại mọi thời điểm vì chỉ một hàng dữ liệu thực sự được lưu trữ trong bộ nhớ.

Các thuộc tính của đối tượng DataReader:

Tên	Mô tả
Read()	Phương thức được sử dụng để đọc mọt dòng.
Close()	Được sử dụng để đóng đối tượng DataReader.
NextResult	Được sử dụng để di chuyển đến hàng dữ liệu tiếp theo trong trường hợp câu lệnh truy vấn trả lại nhiều dòng kết quả

Ví dụ sử dụng Data Reader

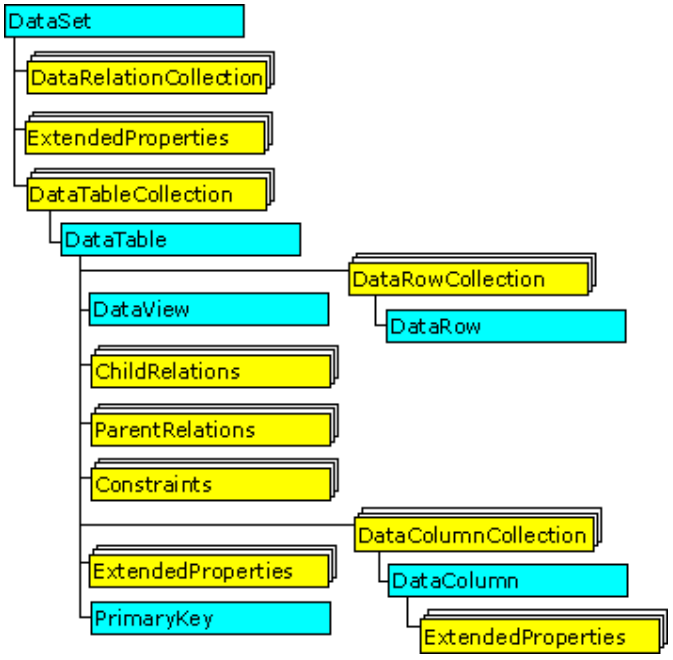
```
private void button2_Click(object sender, EventArgs e)
```

```
        {
            string connectionstring = "PROVIDER=SQLOLEDB;
server=(local);uid=_net;pwd=;database=_NET";
            OleDbConnection conObj = new OleDbConnection(connectionstring);
            conObj.Open();
            OleDbCommand cmd = new OleDbCommand("select * from books");
            OleDbDataReader dr;
            dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                lbltestdatareader.Text = dr["BookID"].ToString();
            }
            conObj.Close();
        }
```

DataSet

Là đối tượng nhận dữ liệu từ cơ sở dữ liệu thông qua DataAdapter, DataSet hoạt động như một cơ sở dữ liệu ảo chứa những bảng, dòng, và cột.

Mô hình DataSet như sau



Mô hình Dataset

Các đối tượng thuộc tính cần quan tâm với đối tượng DataSet bao gồm: DatatableCollection, DataRelationCollection, ExtendedProperties, DataTable, DataRelation, DataRowCollection, DataView, PrimaryKey, DataColumnCollection. Chi tiết những đối tượng thuộc tính này sẽ được đề cập trong các phần sau.

Ví dụ sử dụng DataAdapter đẩy dữ liệu từ cơ sở dữ liệu vào DataSet.

```
private void button3_Click(object sender, EventArgs e)
        {
            string connectionstring =
"PROVIDER=SQLOLEDB;server=(local);uid=_net;pwd=;database=_NET";
            OleDbConnection conObj = new OleDbConnection(connectionstring);
```

```
            conObj.Open();
            OleDbCommand cmd = new OleDbCommand("select * from books");
            OleDbDataAdapter da = new OleDbDataAdapter(cmd);
            DataSet ds = new DataSet();
            da.Fill(ds, "books");
            conObj.Close();
        }
```

BÀI TẬP CÓ HƯỚNG DẪN

Yêu cầu
Ứng dụng quản lý cuộc gọi cần cung cấp khả năng xem chi tiết khách hàng cho bộ phận Bán hàng. Hãy tạo một ứng dụng để hiển thị chi tiết khách hàng cho phòng Kinh doanh.

Giải quyết

Bạn thực hiện những bước sau

1. Tạo một kết nối
2. Tạo một Data Adapter
3. Tạo một Dataset
4. Nhận và hiển thị dữ liệu từ cơ sở dữ liệu.

BÀI TẬP TỰ RÈN LUYỆN

Yêu cầu
Một ứng dụng cần được xây dựng để cho phép người quản lý của công ty NetNam Telecommunication xem những đơn đặt hàng của khách từ cơ sở dữ liệu. Thêm vào đó, với mỗi đặt hàng, phần trăm chi phí của một sản phẩm cần được liệt kê.

Hướng dẫn

1. Tạo một data adapter
2. Tạo một Dataset
3. Nhận, xem dữ liệu từ cơ sở dữ liệu.

Data Binding

Khái niệm Data Binding

Sau khi nhận dữ liệu từ cơ sở dữ liệu, dữ liệu cần được hiển thị trên những control của Windows Form để người sử dụng thấy. Quá trình đó gọi là Data Binding.

Thực hiện Data Binding thông qua câu lệnh truy vấn

Yêu cầu
Giám đốc nhân sự công ty HHO cần xem danh sách nhân viên trong các phòng ban của công ty. Một ứng dụng được yêu cầu xây dựng để thực hiện việc này.

Hướng dẫn:

Thực hiện theo video hướng dẫn. Tên video: PTA_NET_2_0_ADO_Databinding.WMV

Lọc và Sắp xếp dữ liệu

Dữ liệu lấy về từ cơ sở dữ liệu bởi câu lệnh truy vấn select bao gồm tất cả các dòng dữ liệu trong một bảng. Yêu cầu đặt ra là chỉ lấy một số dòng xác định tương ứng với một mã số nào đó. Để giải quyết vấn đề này, ADO.NET cung cấp hai cách thức để thực hiện: lọc dữ liệu trong DataSet và thực hiện lấy dữ liệu cần thiết qua câu lệnh SQL có tham số.

Lọc một Dataset

Dữ liệu từ một table trong cơ sở dữ liệu được lất toàn bộ và đổ vào Dataset, sau đó tiến hành lọc những dòng dữ liệu cần thiết.

Ví dụ sau mô tả cách thực hiện này.

```
private void button4_Click(object sender, EventArgs e)
        {
            string connectionstring = "PROVIDER=SQLOLEDB;
server=(local);uid=_net;pwd=;database=_NET";
            OleDbConnection conObj = new OleDbConnection(connectionstring);
            conObj.Open();
            OleDbCommand cmd = new OleDbCommand("select * from books",conObj);
            OleDbDataAdapter da = new OleDbDataAdapter(cmd);
            DataSet ds = new DataSet();
            da.Fill(ds, "books");
            MessageBox.Show( ds.Tables["books"].Rows.Count.ToString());

            // filter data
            DataView dv = new DataView(ds.Tables["books"]);
            dv.RowFilter = "fprice>20000";
            dv.Sort = "fprice DESC";
            MessageBox.Show(dv.Table.Rows.Count.ToString());
            conObj.Close();
        }
```

Sử dụng câu lệnh SQL có tham số

Ví dụ sau sẽ lấy về danh sách những quyển sách có nhà xuất bản là công ty phát hành sách nhà xuất bản trẻ. (mã số inxb=2);

```
string connectionstring = "PROVIDER=SQLOLEDB;
server=(local);uid=_net;pwd=;database=_NET";
            OleDbConnection conObj = new OleDbConnection(connectionstring);
            conObj.Open();
            OleDbCommand cmd = new OleDbCommand(" select * from books where
inxb=?", conObj);
            cmd.CommandType = CommandType.Text;

            // truyền tham số vào câu lệnh SQL

            cmd.Parameters.Add("@ibookid", 2);
            OleDbDataAdapter da = new OleDbDataAdapter(cmd);
            DataSet ds = new DataSet();
            da.Fill(ds, "books");
            MessageBox.Show(ds.Tables["books"].Rows.Count.ToString());
            conObj.Close();
```

Bạn lưu ý với dòng lệnh

```
cmd.Parameters.Add("@ibookid", 2);
```

Đây là cách sử dụng ngắn gọn khi truyền tham số mà những chuyên gia phát triển phần mềm thực tế thường sử dụng, nguyên mẫu câu lệnh có công thức như sau:

```
OleDbParameter param = new
OleDbParameter("@paramname",OleDbType.VarChar);
param.Value = "value";
```

trong đó, OleDbType là thư viện những kiểu dữ liệu tương ứng với những kiểu dữ liệu lưu trữ trong hệ cơ sở dữ liệu.

Thêm, Cập nhật, xóa dữ liệu trong cơ sở dữ liệu

Thêm mới dữ liệu vào cơ sở dữ liệu

Ví dụ

```
string connectionstring = "PROVIDER=SQLOLEDB;
server=(local);uid=_net;pwd=;database=_NET";
            OleDbConnection conObj = new OleDbConnection(connectionstring);
            conObj.Open();
            OleDbCommand cmd = new OleDbCommand("insert books values(?,?,?,?)",
conObj);
            cmd.CommandType = CommandType.Text;
            cmd.Parameters.Add("@inxb", 2);// Nha xuat ban tre
            cmd.Parameters.Add("@vmasach", "MS0002");
            cmd.Parameters.Add("@vtensach", "Lập trình .NET cho cuộc sống");
            cmd.Parameters.Add("@fgia", 75000);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Đã thêm một sách vào CSDL");
            conObj.Close();
```

Ở ví dụ trên, câu lệnh SQL insert được thực hiện kết hợp với đối tượng tham số Parameters để đẩy dữ liệu vào cơ sở dữ liệu thông qua đối tượng Command.

Cập nhật, xóa dữ liệu trong cơ sở dữ liệu

Quá trình cập nhật và xóa dữ liệu tương tự như với thêm mới cơ sở dữ liệu.

Ví dụ

```
private void button7_Click(object sender, EventArgs e)
        {
            string connectionstring = "PROVIDER=SQLOLEDB;
server=(local);uid=_net;pwd=;database=_NET";
            OleDbConnection conObj = new OleDbConnection(connectionstring);
            conObj.Open();
            OleDbCommand cmd = new OleDbCommand("update books set
vtensach=?,fgia=? where vmasach=?",conObj);
            cmd.CommandType = CommandType.Text;
            cmd.Parameters.Add("@vtensach", ".NET for Real Life");
            cmd.Parameters.Add("@fgia", 85000);
            cmd.Parameters.Add("@vmasach", "MS0002");
            cmd.ExecuteNonQuery();
            MessageBox.Show("Đã sửa đổi nội dung sách trong CSDL");
            conObj.Close();
        }
```

Xóa dữ liệu

```
private void button8_Click(object sender, EventArgs e)
        {
            string connectionstring = "PROVIDER=SQLOLEDB;
server=(local);uid=_net;pwd=;database=_NET";
            OleDbConnection conObj = new OleDbConnection(connectionstring);
            conObj.Open();
            OleDbCommand cmd = new OleDbCommand("delete from books where
vmasach=?", conObj);
            cmd.CommandType = CommandType.Text;
            cmd.Parameters.Add("@vmasach", "MS0002");
            cmd.ExecuteNonQuery();
            MessageBox.Show("Đã sửa đổi nội dung sách trong CSDL");
            conObj.Close();
        }
```

Chúng ta đã học

Bây giờ, bạn đã có thể hoàn toàn thao tác trên cơ sở dữ liệu từ ứng dụng bạn phát triển. Một số điểm sau bạn cần lưu ý để ghi nhớ

- Kết nối luôn phải được đóng lại sau mỗi lần truy xuất đến cơ sở dữ liệu.
- Sử dụng đối tượng Data Adapter, DataSet và đối tượng Command trong thao tác nhận dữ liệu từ CSDL
- Sử dụng Command để thao tác cập nhật, thêm mới, xóa dữ liệu
- Sử dụng DataReader để có tốc độ truy xuất (chỉ nhận) dữ liệu nhanh nhất.
- Đối tượng Parameters được sử dụng để truyền giá trị cho tham số trong câu lệnh SQL

Xây dựng một lớp CSDL dùng chung

Khi thao tác với CSDL, việc tạo kết nối, mở kết nối, thực hiện câu truy vấn dữ liệu, … và đóng kết nối được thực hiện rất nhiều lần, Sẽ là rất tốt thời gian nếu chúng ta cứ lập đi lập lại quá trình này.

Chúng ta sẽ cùng xây dựng một lớp chuyên biệt để thực hiện việc giao tiếp với CSDL, và người lập trình sẽ không phải mất công thực hiện lại các bước thao tác với các đối tượng như Connection, DataAdapter… và cuối cùng là giúp giảm thiểu thời gian cho việc xây dựng ứng dụng.

Xây dựng lớp giao tiếp với CSDL - DBClass

Mã lệnh của lớp DBClass như sau

```
using System;
using System.Data;
using System.Data.OleDb;

public class OleDBClass
{
    static string strConStr = "";

    public static string ConnectingString
    {
        get
        {
            if (!strConStr.Equals(""))
                return strConStr;
            else
                return null;
        }
        set
        {
            strConStr = value;
        }
    }
    public static OleDbConnection GetConnection()
    {
        OleDbConnection con = new OleDbConnection(strConStr);
        con.Open();
        return con;
    }

    public static DataTable GetData(OleDbCommand cmd)
    {
```

```
        try
        {
            if (cmd.Connection != null)
            {
                using (DataSet ds = new DataSet())
                {
                    using (OleDbDataAdapter da = new OleDbDataAdapter())
                    {
                        da.SelectCommand = cmd;
                        da.Fill(ds);
                        return ds.Tables[0];
                    }
                }
            }
            else
            {
                using (OleDbConnection conn = GetConnection())
                {
                    using (DataSet ds = new DataSet())
                    {
                        using (OleDbDataAdapter da = new OleDbDataAdapter())
                        {
                            da.SelectCommand = cmd;
                            da.SelectCommand.Connection = conn;
                            da.Fill(ds);
                            return ds.Tables[0];
                        }
                    }
                }
            }
        }
        finally
        {

        }
    }

    public static OleDbDataReader GetReadOnlyData(OleDbCommand cmd)
    {
        try
        {
            if (cmd.Connection != null)
            {
                return cmd.ExecuteReader();
            }
            else
            {
                using (OleDbConnection conn = GetConnection())
                {
                    cmd.Connection = conn;
                    datareader = cmd.ExecuteReader();
                    return cmd.ExecuteReader();
                }
            }
        }
        finally
        {

        }
    }

    public static DataSet GetDsData(OleDbCommand cmd)
    {
```

```
        try
        {
            if (cmd.Connection != null)
            {
                using (DataSet ds = new DataSet())
                {
                    using (OleDbDataAdapter da = new OleDbDataAdapter())
                    {
                        da.SelectCommand = cmd;
                        da.Fill(ds);
                        return ds;
                    }
                }
            }
            else
            {
                using (OleDbConnection conn = GetConnection())
                {
                    using (DataSet ds = new DataSet())
                    {
                        using (OleDbDataAdapter da = new OleDbDataAdapter())
                        {
                            da.SelectCommand = cmd;
                            da.SelectCommand.Connection = conn;
                            da.Fill(ds);
                            return ds;
                        }
                    }
                }
            }
        }
        finally { }
    }

    public static DataTable GetData(string sql)
    {
        try
        {
            using (OleDbConnection conn = GetConnection())
            {
                using (OleDbCommand cmd = conn.CreateCommand())
                {
                    cmd.CommandType = CommandType.Text;
                    cmd.CommandText = sql;
                    using (DataSet ds = new DataSet())
                    {
                        using (OleDbDataAdapter da = new OleDbDataAdapter())
                        {
                            da.SelectCommand = cmd;
                            da.SelectCommand.Connection = conn;
                            da.Fill(ds);
                            return ds.Tables[0];
                        }
                    }
                }
            }
        }
        finally
        {

        }
    }
    public static void ExecuteNonQuery(OleDbCommand cmd)
```

```
    {
        try
        {
            using (OleDbConnection conn = GetConnection())
            {
                cmd.Connection = conn;
                cmd.ExecuteNonQuery();
            }
        }
        finally
        {

        }
    }

    public static object ExecuteScalar(OleDbCommand cmd)
    {
        try
        {
            using (OleDbConnection conn = GetConnection())
            {
                cmd.Connection = conn;
                return cmd.ExecuteScalar();
            }
        }
        finally
        {

        }
    }

}
```

Sử dụng lớp giao tiếp với CSDL - DBClass

Việc bây giờ là chúng ta sẽ sử dụng lớp CSDL như thế nào. Sau đây là mã lệnh mẫu để sử dụng.

```
DBClass.ConnectingString = "PROVIDER=SQLOLEDB;
server=(local);uid=_net;pwd=;database=_NET";
        // việc gọi một câu truy vấn bây giờ đơn giản là việc gọi hàm trong lớp
DBClass
        DataTable dt = DBClass.GetData("select * from items");
```

Chúng ta có thể thấy đó là việc thực hiện câu truy vấn dữ liệu từ bảng items đã trở nên cực kỳ đơn giản hơn bao giờ hết. Không cần phải mở kết nối Connection, sử dụng DataAdapter… bạn vẫn có được dữ liệu từ bảng items. Nói cho đúng thì chúng ta đã đưa những công doạn này vào trong lớp OleDBClass và giờ đây chúng ta chỉ quan tâm tới mục đích sử dụng câu truy vấn mà không mất quá nhiều mã lệnh để thực hiện các bước chuẩn bị.

Xây dựng Ứng dụng Windows Form

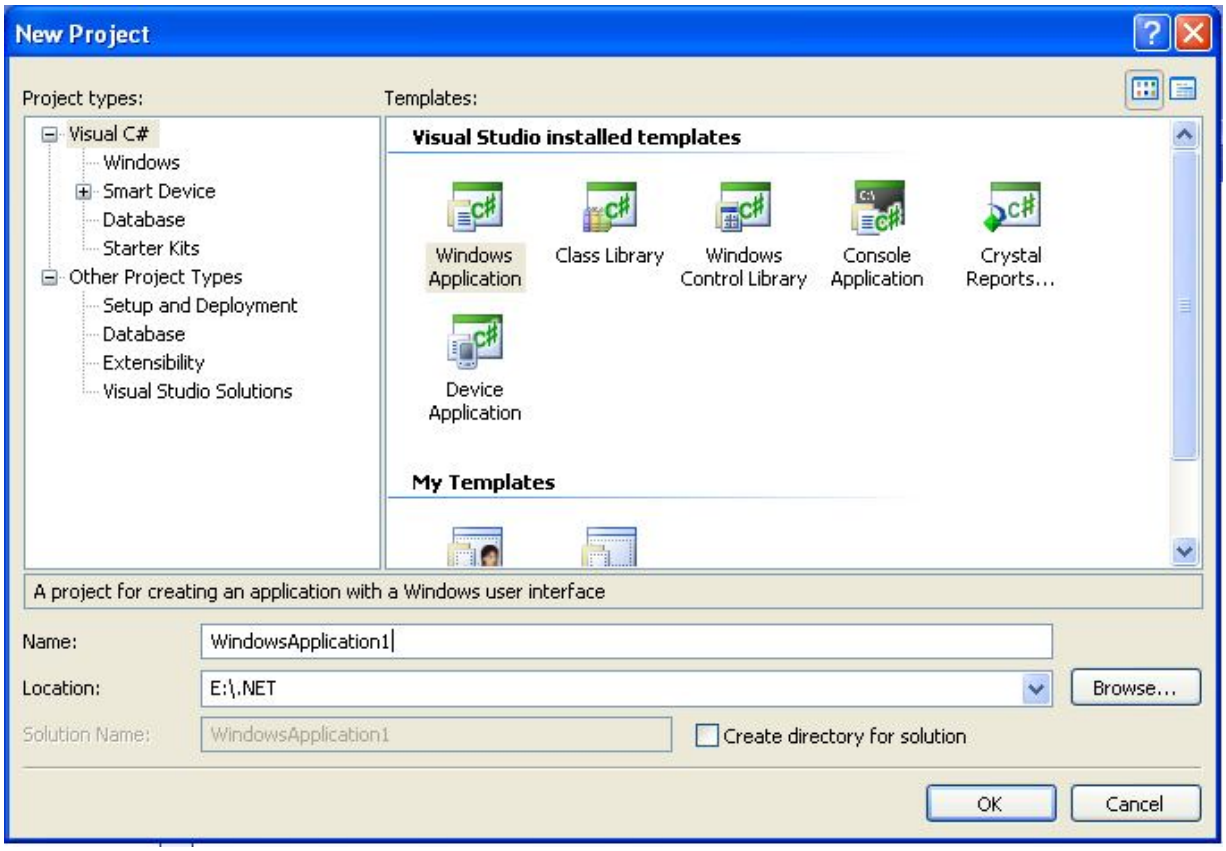
Xây dựng Windows Forms

Visual Studio .NET Integrated Development Environment (IDE).

Cung cấp cho bạn những giao diện chung cho việc phát triển nhiều loại dự án khác nhau trên nền tảng .NET. IDE cho phép khả năng thiết kế giao diện người dùng cho ứng dụng, viết mã lệnh, biên dịch, và kiểm lỗi cho ứng dụng. Visual Studio .NET cung cấp nhiều ngôn ngữ để phát triển ứng dụng trong bộ .NET của Microsoft như: Visual Basic, Visual C#, Visual C++….

Tạo một dự án trong Visual Studio .Net

Chạy ứng dụng Visual Studio 2005 từ thực đơn **Start → Program → Microsoft Visual Studio 2005 → Microsoft Visual Studio 2005.** Tạo mới ứng dụng bằng cách nhấn vào menu **File → New → Project**. Cửa sổ tạo mới Project xuất hiện.



Trong cửa sổ **New Project**, **Project Types pane** hiển thị danh mục những kiểu project mà bạn có thể tạo trong VS. Chúng ta quan tâm tới hai loại project đó là **Visual C#** và **Setup and Deployment.** Trong khi Visual C# là kiểu dự án cho phép tạo ra ứng dụng bằng ngôn ngữ C#, thì Setup and Deployment là kiểu project để triển khai dự án đến người dùng cuối.

Chọn **Visual C# → Windows** tại Project Types.

Ở cửa sổ Templates, một số mẫu ứng dụng có sẵn để giúp người phát triển nhanh chóng tạo ra ứng dụng phù hợp theo yêu cầu. Có các kiểu project template sau cần chú ý nhất:

- Windows Application: được dùng để tạo những ứng dụng Windows.

- Class Library: Sử dụng để tạo ra những compnent sử dụng lại trong các dự án khác
- Windows Controls Library: tạo những công cụ cho môi trường ứng dụng Window
- Console Application: tạo ra ứng dụng console chạy từ dòng lệnh, giao diện ký tự.

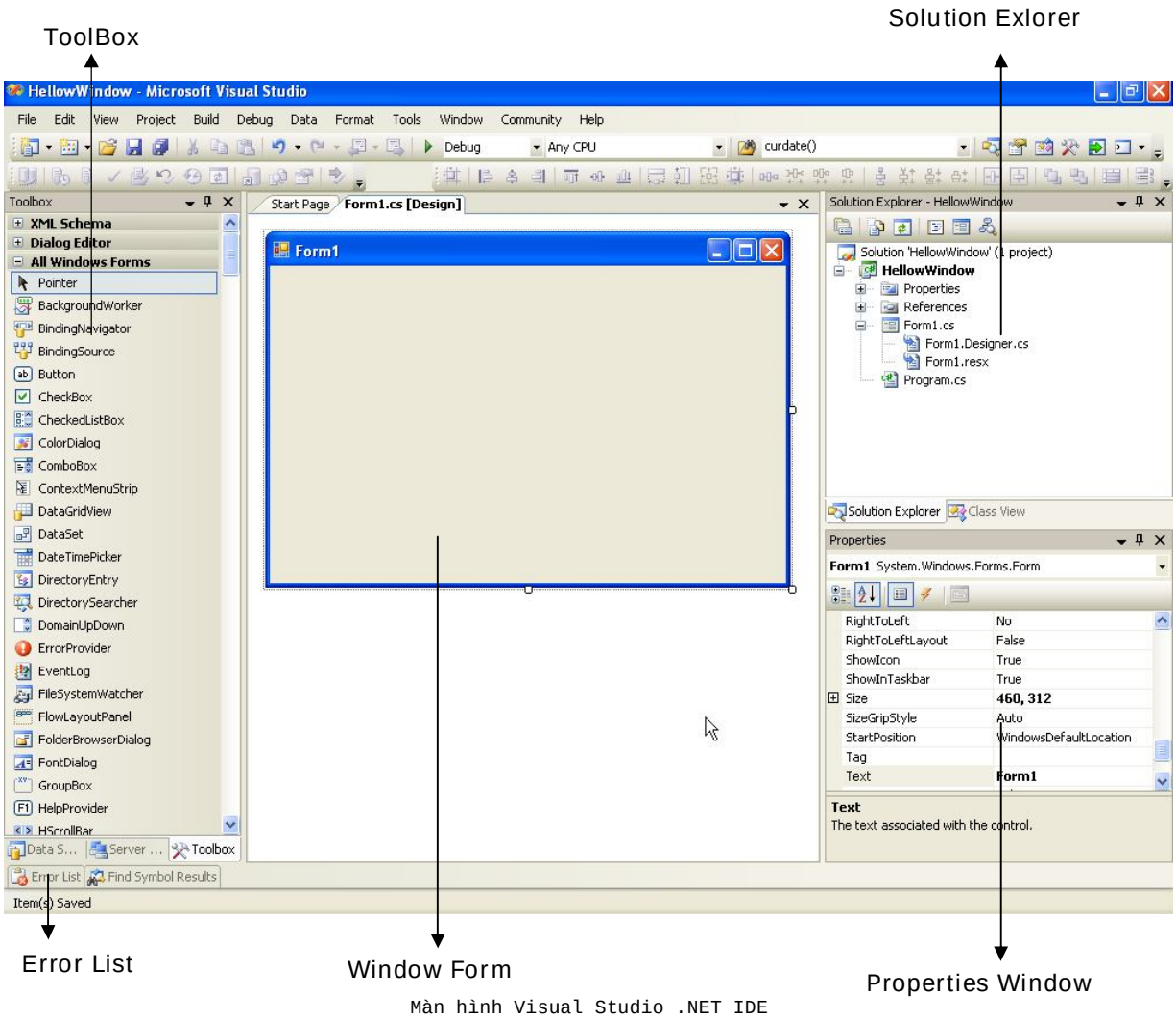
Chọn **Windows Application**

Đặt tên cho dự án đầu tiên là **HelloWindow** tại **Name**

Chọn thư mục lưu trữ dự án tại **Location**

Click **OK**

Màn hình dự án xuất hiện như sau



Màn hình Visual Studio .NET IDE

Chúng ta quan tâm đến những cửa sổ sau:

Window Form: nơi sử dụng để thiết kế giao diện chương trình.

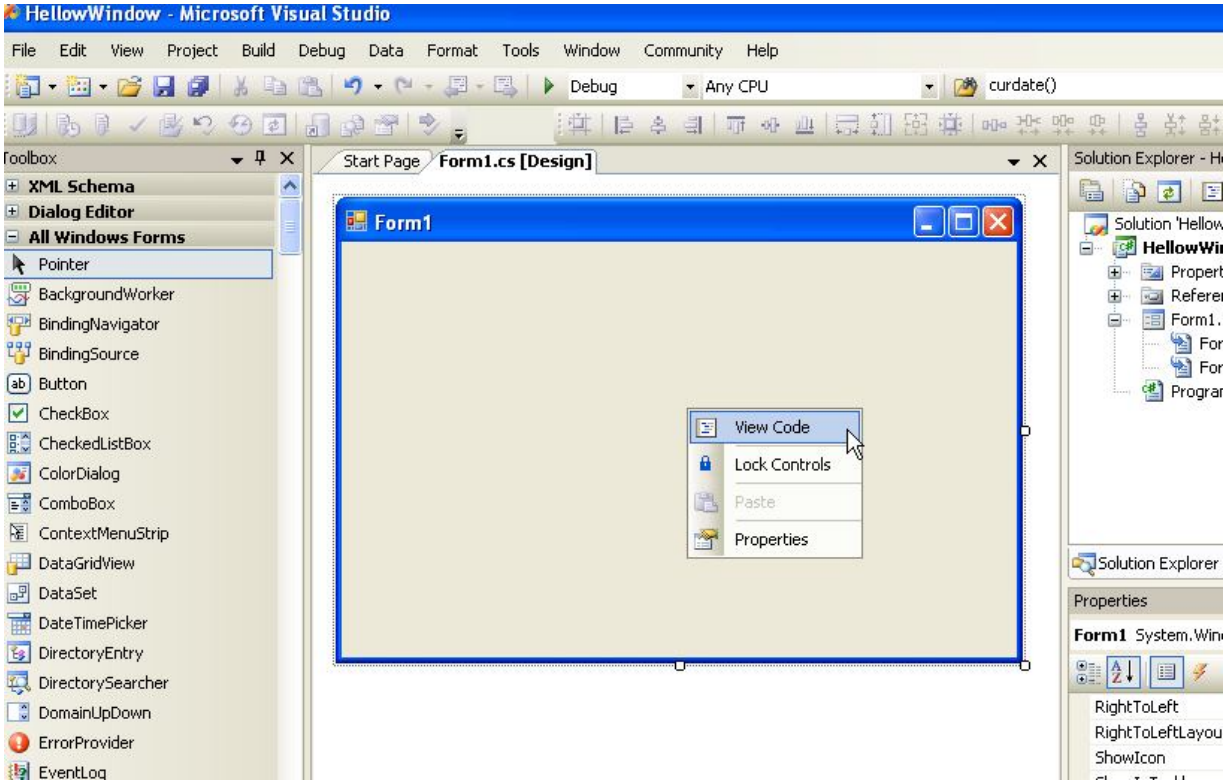
ToolBox: chứa đựng những controls được xây dựng sẵn. Có nhiều tab trong cửa sổ ToolBox liệt kê những control theo những loại khác nhau.

Solution Explorer: liệt kê tên dự án, những file thiết kế, mã nguồn, … của ứng dụng

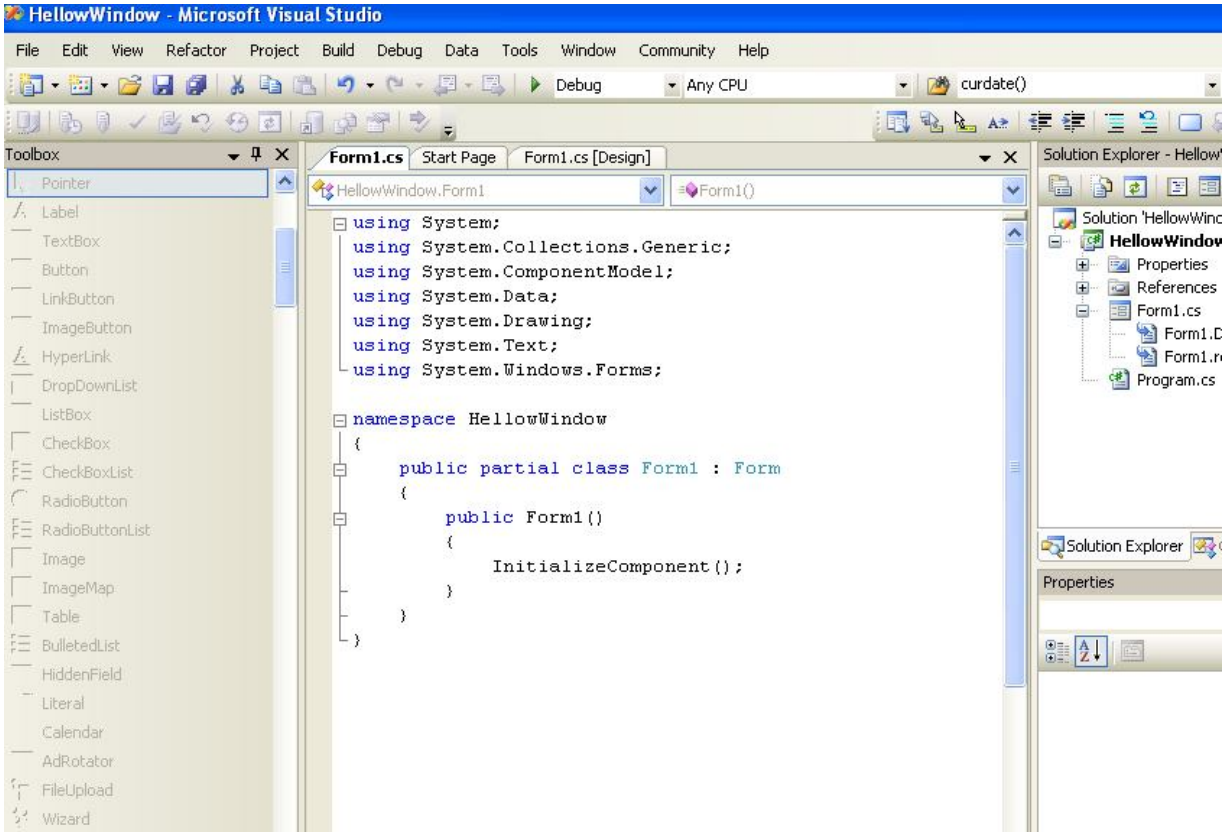
Properties Window: Cửa sổ chứa đựng những thuộc tính của một control nào đó, ví dụ Form1 có các thuộc tính như Text, StartPosition, ShowinTaskBar…

Erros List: nơi xuất hiển những lỗi của chương trình

Tiếp tục thực hiện ứng dụng…, tại Window Form, click chuột phải, một popup Menu xuất hiện, trong đó có mục View Code, mục này cho phép xuất hiện cửa sổ soạn thảo mã lệnh điều khiển chương trình



Cửa sổ mã lệnh chương trình hiển thị như sau:



Cửa sổ mã lệnh cho phép nhà phát triển soạn thảo mã lệnh điều khiển chương trình

Tiếp tục thực hiện ứng dụng…, trở lại màn hình Window Form, double click vào Form1, và soạn thảo mã lệnh như sau vào hàm Form_Load() do chương trình khởi tạo.

```
private void Form1_Load(object sender, EventArgs e)
{
    MessageBox.Show("Welcome to Window Form");
}
```

Chọn menu **Debug → Start Debugging** (hoặc nhấn F5) để chạy ứng dụng.

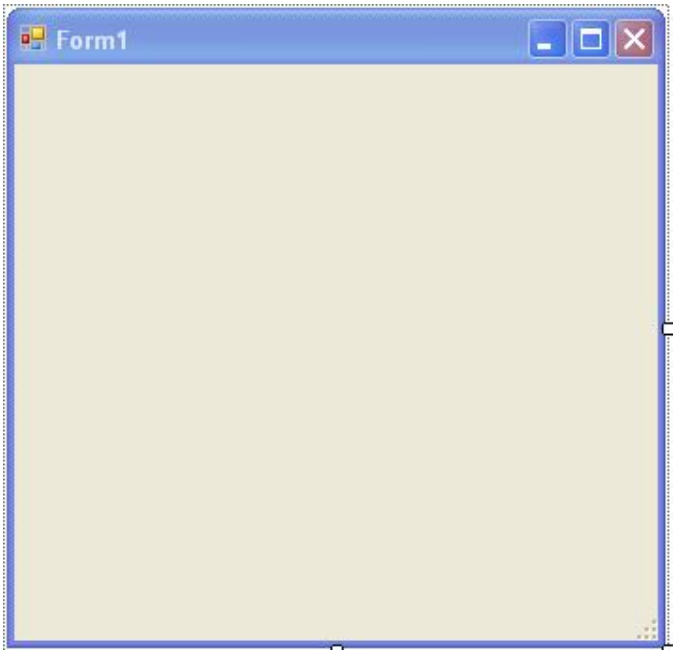
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace HellowWindow
{
    public parti                                    rm
    {
        public F
        {
            Init
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            MessageBox.Show("Welcome to Window Form");
        }
    }
}
```

Bạn đã hoàn tất ứng dụng Window Form đầu tiên với C#. bây giờ, tôi cùng bạn tiếp cận đến những điều khiển trong thư viện WindowsForm được cung cấ sẵn trong C#.

Window Form Controls

Windows Form

Một Windows Form là một cửa sổ được xuất hiện trong một ứng dụng. Mỗi Windows Form là một lớp được kế thừa từ lớp Form nằm trong Namespace System.Windows.Forms



Thuộc tính Windows Form

Những thuộc tính chung của Windows Form được liệt kê theo bảng sau

Properties	Mô tả
Name	Là thuộc tính để xác định tên của form, mặc định, thuộc tính **Name** của form đầu tiên trong ứng dụng là **Form1**
Backcolor	Thuộc tính xác định màu nền của form
BackgroundImage	Thuộc tính xác định hình nền cho form
Font	Thuộc tính xác định kiểu, kích thước, và loại font được hiển thị trên form và trong những controls trong form.
Size	Kích thước của form bao gồm: Width và Height
Start Position	Thuộc tính xác định vị trí mặc định xuất hiện của Form trên màn hình máy tính người sử dụng, có các thuộc tính sau: - **Manual** - vị trí và kích thước của form phục thuộc vào vị trí xuất hiện của nó - **CenterScreen -** xuất hiện ở chính giữa màn hình - **WindowsDefaultLocation** - form xuất hiện tại vị trí mặc định của Windows theo kích thước của form. - **Windows DefaultBounds** - form được hiển thị tại vị trí mặc định của Windows và các chiều của chúng phụ thuộc vào hệ điều hành Windows. - **Center Parent** - form được mở như một cửa sổ con của một form khác và xuất hiện tại vị trí chính giữa so với form cha.
Text	Xác định tiêu đề của form tại Title Bar
WindowState	Xác định trạng thái xuất hiện của form: normal, maximized, hay minimized.

Sự kiện trong Windows Form

Những sự kiện trong Windows Form được liệt kê như bảng sau

Events	Mô tả
Click	Sự kiện này xảy ra khi người dùng click vào bất kỳ nơi nào trên Windows Form
Closed	Sự kiện này xảy ra khi một form được đóng lại
Deactivate	Sự kiện xảy ra khi một form bị mất trạng thái sử dụng
Load	Sự kiện xảy ra khi một form được tải trong bộ nhớ cho lần đầu tiên.
MouseMove	Sự kiện này xuất hiện khi chuột được rê trên một form

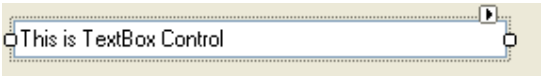
Events	Mô tả
MouseDown	Sự kiện xảy ra khi chuột được nhấn trên form
MouseUp	Sự kiện xảy ra khi chuột được thả trên form.

Hàm thao tác Với Windows Form

Methods	Mô tả	Ví dụ
Show()	Được sử dụng để xuất hiện một form bằng cách set thuộc tính **Visible** của form ấy là True	`Form1 frmObj = new Form1(); frmObj.Show();`
Activate()	Sử dụng để kích hoạt trạng thái sử dụng của Form và đưa trạng thái sử dụng về Form ấy.	`frmObj.Activate();`
Close()	Dùng để đóng một Form	`frmObj.Close();`
SetDesktopLocation()	Hàm này dùng để định vị trí của Form trên màn hình	`SetDesktopLocation(100,150)`

TextBox Control

TextBox là điều khiển cho phép nhận giá trị từ người dùng trên một Form. Mặc định giá trị lớn nhất mà TextBox nhận là 2048 ký tự.

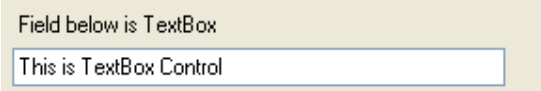


TextBox Control

Property	Mô tả	Ví dụ
Text	Xác định giá trị hiển thị bên trong TextBox	`txtUserName.Text=”pta30000”;`
Multiline	Cho phép TextBox hiển thị nhiều dòng chữ.	`txtContent.Multiline = true;`
PasswordChar	Thuộc tính này cho phép lấy một ký tự làm đại diện cho tất cả các ký tự khác được nhập vào từ người dùng	`txtPassword.PasswordChar=”*”;`

Label Control

Control được sử dụng để hiển thị chữ trên form và không cho phép người dùng thay đổi. Label được sử dụng để mô tả thông tin cho những control khác trên Form.



Label Control

Dòng chữ xuất hiện bên trên TextBox đó là Label, có mục đích giải thích cho TextBox.

LinkLabel Control

LinkLabel được sử dụng để hiển thị một chuỗi như một liên kết. Khi bạn nhấn vào liên kết, nó sẽ mở ra một form khác hoặc một Website.

Để mở ra một form khác, .NET cung cấp sự kiện LinkClicked, bạn thực hiện như sau:

```
private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
    Form1 frmobj = new Form1();
    frmobj.Show();
}
```

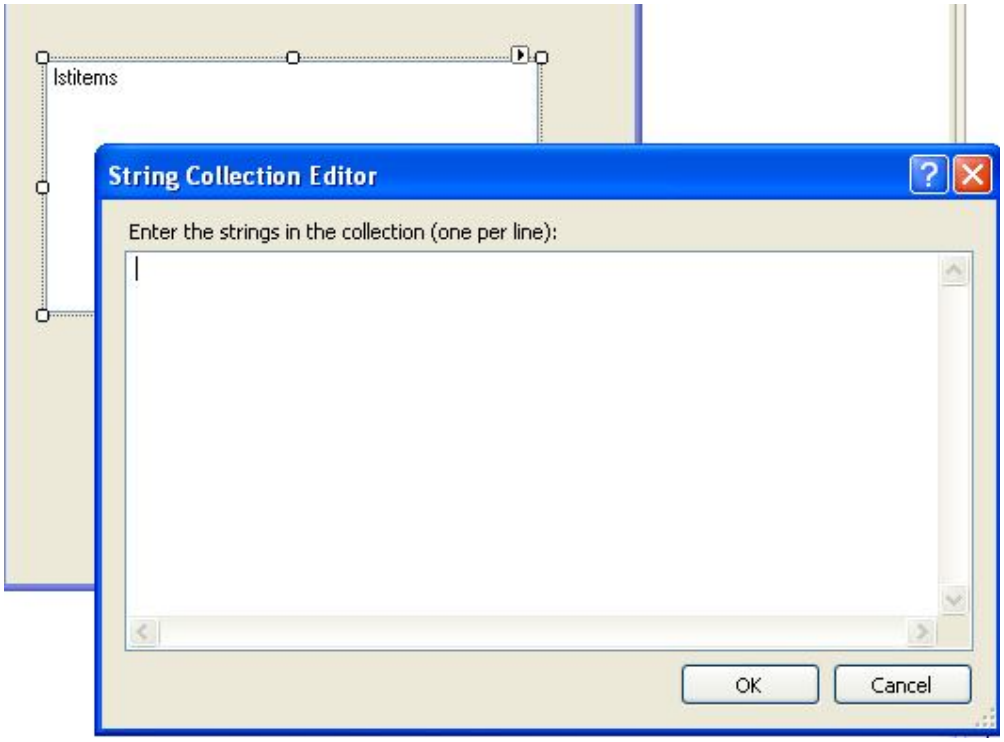
Để mở một website, bạn có thể dụng đoạn mã sau đặt bên trong Sự kiện Link

```
private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
     System.Diagnostics.Process.Start( "http://www.itgatevn.com.vn");
}
```

ListBox Control

ListBox được sử dụng để hiển thị một danh sách phần tử đến người dùng. Một người dùng có thể chọn một trong những phần tử này.

Bạn có thể thêm danh mục những phần tử vào trong một ListBox bằng cách, chọn ListBox, tại cửa sổ Properties Window, chọn thuộc tính Items, nhấn vào nút  và cửa sổ String Collection Editor xuất hiện như hình dưới.



Nhập vào VietNam, gõ Enter, nhập vào USA, gõ Enter. Sau đó nhấn OK. ListBox của bạn sẽ có hai phần tử VietNam và USA.

Bạn cũng có thể thêm những phần tử vào ListBox tại lúc chạy chương trình bằng cách sử dụng phương thức Add() của thuộc tính Items trong ListBox.

```
lstitems.Items.Add("German");
lstitems.Items.Add("China");
```

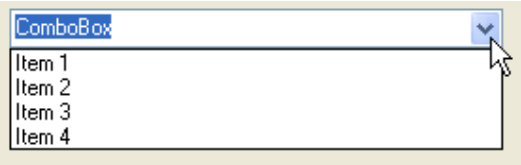
Những thuộc tính thường sử dụng nhất của ListBox được liệt kê theo bảng dưới đây

Property	**Mô tả**	**Ví dụ**
SelectionMode	Xác định cách thức mà người dùng lựa chọn những phần tử trong ListBox. Có 4 giá trị - None: người dùng không thể chọn bất cứ phần tử nào trong ListBox - One: người sử dụng chỉ chọn một giá trị từ ListBox - MultiSimple: cho phép người sử dụng chọn nhiều phần tử từ ListBox. - MultiExtended: người dùng có thể chọn nhiều phần tử và sử dụng phím SHIFT, CTRL và phím mũi tên để chọn những phần tử từ ListBox.	ListBox1.SelectionMode = SelectionMode. MultiSimple;

Property	Mô tả	Ví dụ
Sorted	Thuộc tính xác định khi nào những phần tử trong ListBox có thể được sắp xếp	ListBox1.Sorted = true;
SelectedIndex	Thuộc tính cho phép gán hoặc nhận vị trí được chọn hiện tại trong ListBox.	ListBox1.SelectedIndex = 2;
SelectedItem	Thuộc tính này được dùng để gán hoặc nhận phần tử đang được chọn	MessageBox.Show(ListBox1.SelectedItem)

ComboBox Control

ComboBox được sử dụng để hiển thị danh sách những phần tử thả xuống. ComboBox là sự kết hợp của TextBox cho phép người dùng nhập giá trị và một danh sách thả xuống cho phép người sử dụng chọn phần tử.



ComboBox

Hầu hết những thuộc tính của ComboBox giống như thuộc tính của ListBox, nhưng ComboBox có thể thuộc tính Text.

Thuộc tính Text: cho phép gán hoặc nhận giá trị được người sử dụng nhập vào từ ComboBox.

Để thêm những phần tử vào ComboBox, tương tự như với ListBox, bằng cách sử dụng phương thức Add() của thuộc tính Items.

```
ComboBox1.Items.Add(“VietNam”);
ComboBox1.Items.Add(“Thailand”);
```

CheckBox Control

Control này được dùng để gán tùy chọn Yes/No hoặc True/False. Những thuộc tính thường sử dụng của CheckBox

Property	Mô tả	Ví dụ
Text	Thuộc tính này sử dụng để nhận hoặc gán chuỗi ký tự là tiêu đề của Checkbox	CheckBox1.Text = “Yes”;
Checked	Là thuộc tính được sử dụng để xác định checkbox được chọn	checkBox1.Checked = true;

Property	Mô tả	Ví dụ

RadioButton Control

RadioButton được dùng để cung cấp sự lựa chọn một trong một nhóm tiêu chí cho người dùng. Chỉ một RadioButton được chọn trong một nhóm. Những thuộc tính thường sử dụng của RadioButton giống như CheckBox, bao gồm: Text, Checked.

GroupBox Control

Được sử dụng để nhóm những control liên quan lại với nhau. GroupBox thường được sử dụng để nhóm hai hay nhiều RadioButton để cung cấp một sự lựa chọn duy nhất giữa chúng.

Button Control

Button được sử dụng để thực hiện một tác vụ khi người sử dụng nhấn vào nó.

```
private void button1_Click(object sender, EventArgs e)
        {
            this.button1.Text = this.Text;
        }
```

Đoạn mã lệnh trên thực hiện việc gán Caption của button1 bởi Caption của Form.

Tạo control động trong Windows Form

Bên cạnh việc tạo những control khi thiết kế, bạn cũng có thể tải những điều khiển khi chương trình đang thực thi. Ví dụ sau minh họa việc tải một TextBox vào Form.

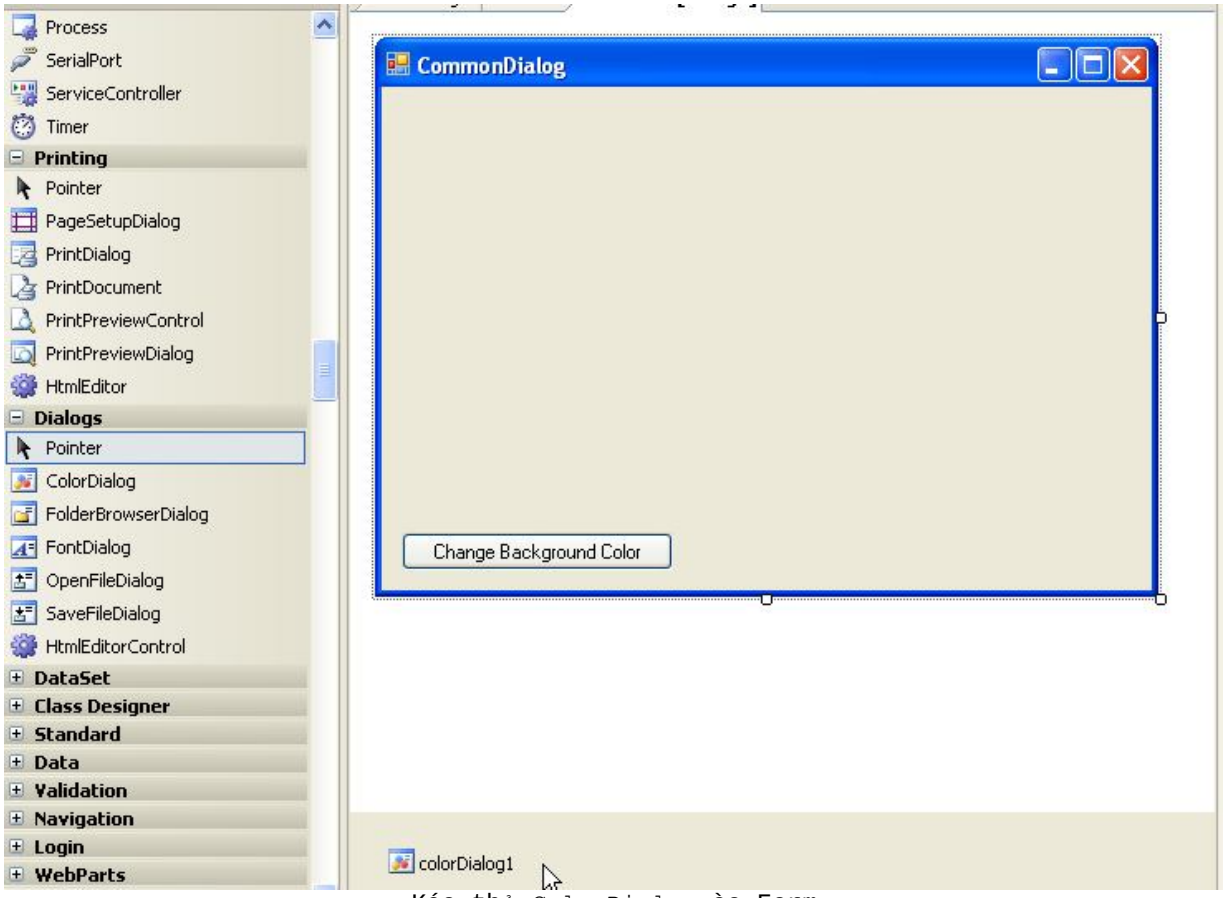
```
TextBox txtBox = new TextBox();
txtBox.Text = "Dynamic TextBox";
this.Controls.Add(txtBox);
```

Sử dụng những lớp thừa kế CommonDialog

Lớp ColorDialog

ColorDialog là lớp công cụ màu sắc cho phép người sử dụng lựa chọn màu sắc từ bảng màu. Sử dụng công cụ này bằng cách kéo thả **ColorDialog** từ **ToolBox** vào trong Form.

Bạn mong muốn rằng sau khi nhấn Button "Change Backgroud Color" và lựa chọn màu sắc thì màu nền của Form sẽ thay đổi theo màu được chọn, bạn thực hiện như sau:

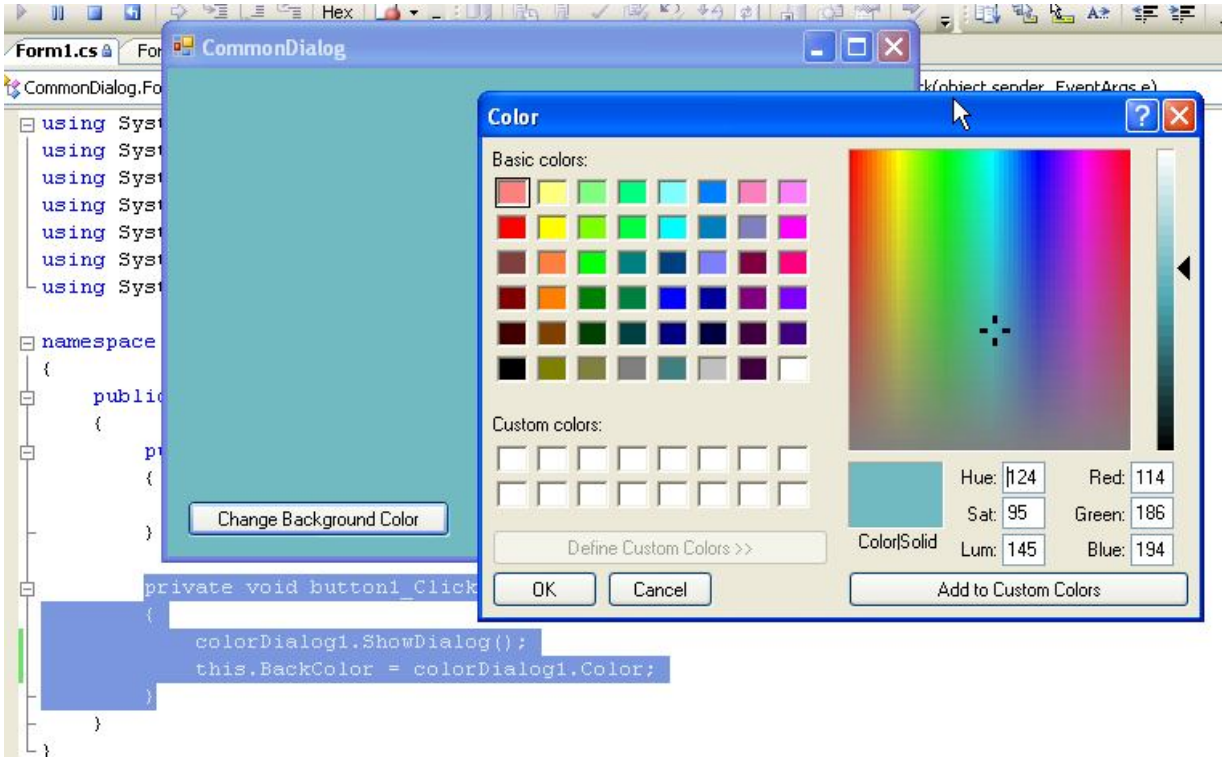


Kéo thả ColorDialg vào Form

Tại sự kiện Click() của Button, thực hiện đoạn mã lệnh

```
private void button1_Click(object sender, EventArgs e)
        {
            colorDialog1.ShowDialog();
            this.BackColor = colorDialog1.Color;
        }
```

Kết quả được hiển thị như hình sau



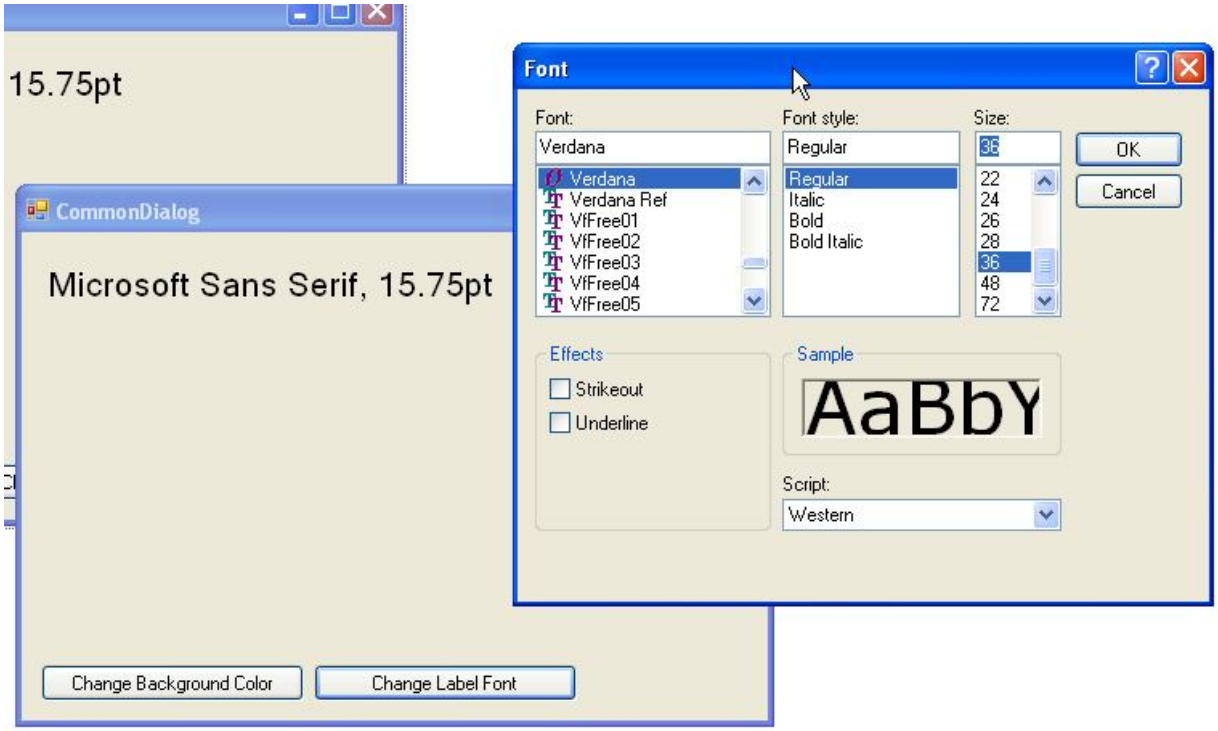
Màu nền của Form đã được thay đổi khi lựa chọn màu từ bảng màu

Lớp FontDialog

Giống như ColorDialog, FontDialog được sử dụng để người dùng lựa chọn kiểu hiển thị, kích thước của chữ. Để sử dụng FontDialog, bạn thực hiện theo các bước sau đây

Kéo thả FontDialog từ ToolBox vào Form.
Thực hiện đoạn mã lệnh sau

```
private void button2_Click(object sender, EventArgs e){

            fontDialog1.ShowDialog();

            this.label1.Font = fontDialog1.Font;

        }
```

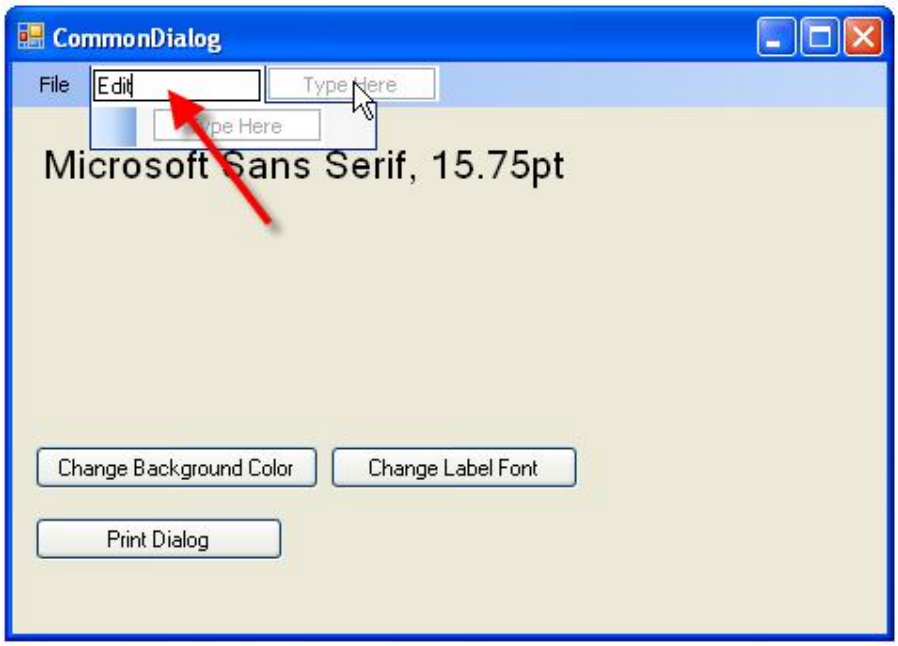


Lựa chọn Font để thay đổi với FontDialog

Làm việc với Menus và xây dựng ứng dụng MDI

Trong môi trường Windows, bạn có thể sử dụng Menu để tăng sự tiện ích cho người sử dụng ứng dụng. Menus có hai dạng, Menu xuất hiện tại thanh Menu và menu xuất hiện khi người sử dụng nhấn chuột phải, được gọi là Context menus.

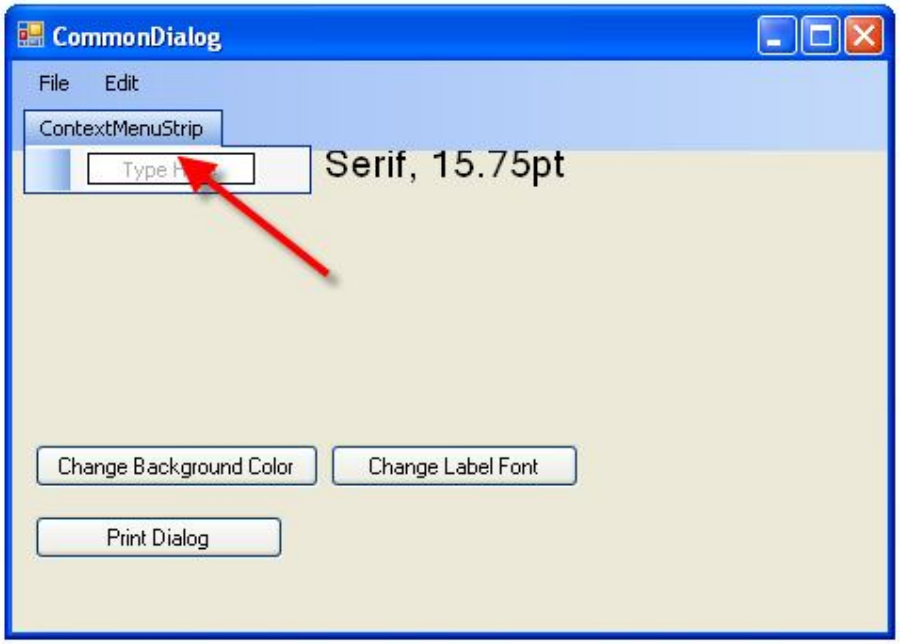
Menus xuất hiện tại Menu Bar được khởi tạo bằng cách kéo thả control MenuStrip từ ToolBox vào Form Design, khi ấy Menu xuất hiện cho phép bạn khởi tạo danh mục menu như hình sau



Khởi tạo Menu bằng công cụ MenuStrip

Context Menus được xuất hiện khi người sử dụng nhấn chuột phải. Để tạo ra context menu, bạn có thể thực hiện như sau:

Kéo thả điều khiển ContextMenuStrip vào cửa sổ thiết kế ứng dụng. ContextMenu xuất hiện như hình sau:



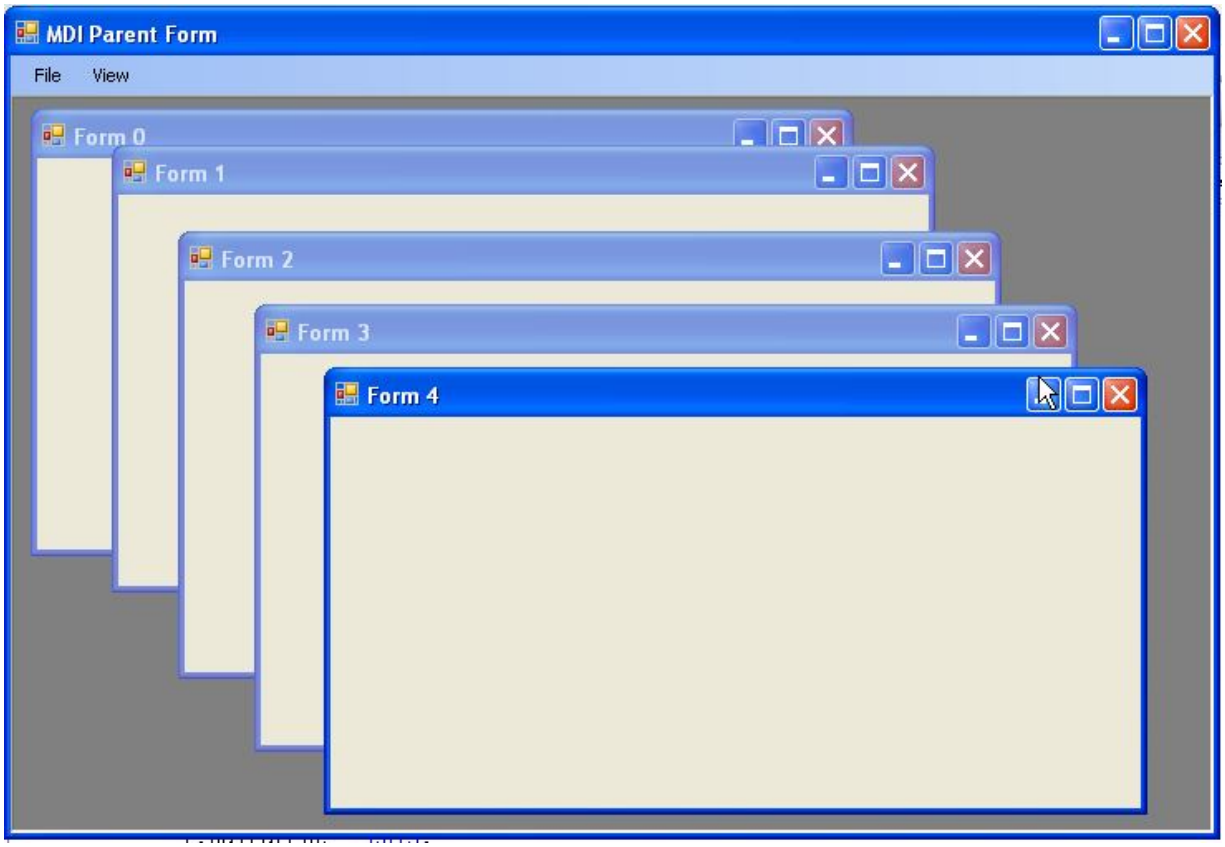
ContextMenu xuất hiện khi được kéo thả từ ToolBox

Xây dựng ứng dụng MDI

Ứng dụng Multiple Document Interface (MDI) là ứng dụng cho phép hiển thị nhiều cửa sổ ứng dụng tại cùng thời điểm. Một ví dụ về ứng dụng MDI đó là chương trình Microsoft Word, bạn có thể tạo cùng lúc nhiều cửa sổ soạn thảo trên cùng một cửa sổ chung của ứng dụng.

Một ứng dụng MDI có thể có một hoặc nhiều form MDI cha, và mỗi form MDI cha này có thể chứa nhiều form MDI con. Một MDI form cha có thể chứa nhiều MDI form con, nhưng một MDI form con không thể thuộc về nhiều MDI form cha.

MDI cha có thể được khởi tại tại thời điểm thiết kế ứng dụng hoặc khi chương trình chạy bằng cách gán thuộc tính **IsMdiContainer** của Form là **true**.



Ứng dụng MDI Form

Bài tập có hướng dẫn

1. Xây dựng ứng dụng máy tính cá nhân thực hiện các phép tính căn bản: cộng, trừ, nhân, chia, lũy thừa

2. Tạo một giao diện cho ứng dụng soạn thảo mô phỏng ứng dụng Microsoft Word sử dụng MDI Form bao gồm các Menu: File, Edit, View, Insert, Format, Window, Help

3. Xây dựng ứng dụng soạn thảo văn bản (Word Pad) sử dụng MDI Form đã tạo ở trên.

Bài tập tự luyện

Xây dựng trò chơi "Tập gõ bàn phím" cho phép xuất hiện những ký tự trên bàn phím và so sánh với phím mà người sử dụng nhập vào. Tính kết quả số lần người chơi gõ đúng phím và số lần người chơi gõ sai.

Quản lý lỗi trong lập trình C#

Khi lập trình, đôi khi người phát triển gặp những trường hợp lỗi xảy ra khi chương trình đang chạy. Để quản lý và điều khiển những lỗi này, C# cung cấp cho người phát triển cách thức bẫy lỗi để chương trình của bạn chạy xuyên suốt thông các từ khóa try, catch và finally. Ví dụ sau mô tả cách sử dụng try…catch, và finally

```
int SafeDivision(int x, int y)
{
    try
    {
        return (x / y);
    }
    catch (System.DivideByZeroException dbz)
    {
        System.Console.WriteLine("Lỗi khi chia một số cho 0!");
        return 0;
    }
}
```

Ở ví dụ trên, bạn có thể thấy với hàm SafeDivision nếu truyền vào hai tham số x=2, y=0 thì chương trình sẽ gặp lỗi. Nhưng khi chạy chương trình, nếu bạn gọi hàm SafeDivision, và truyền vào hai tham số như trên sẽ xuất hiện dòng chữ Division by zero attempted!!, như thế lỗi của đoạn mã lệnh đã được quản lý. Sở dĩ lỗi xảy ra được quản lý là do chúng ta đã sử dụng bẫy lỗi mà C# cung cấp thông qua từ khóa try…catch. Công thức của try…catch như sau:

```
try
{
 // Câu lệnh thực hiện có thể phát sinh lỗi
}
catch(Type x)
{
 // Câu lệnh xử lý khi gặp lỗi
}
finally
{
 // Câu lệnh luôn được thực thi
}
```

Với công thức trên, bạn cần chú ý một điểm, những câu lệnh trong khối finally được xem như luôn luôn được thực thi.

Bài tập có hướng dẫn

1. Xây dựng ứng dụng cho phép quản lý danh sách nhân viên của một công ty theo phòng ban. Chi tiết nhân viên công ty bao gồm: mã số, họ và tên, ngày

sinh, giới tính, địa chỉ, điện thoại, ngày vào làm, ngày kết thúc, chi tiết phòng ban bao gồm: mã số phòng ban, tên phòng ban.

2. Xây dựng chương trình quản lý kho hàng, sản xuất của công ty đồ chơi GlobalToys.

Video hướng dẫn thực hiện chương trình PTA_NET_APP_ADO_NET.WMV

Dự án tự ôn luyện

Xây dựng chương trình quản lý kho hàng với những yêu cầu sau

- Quản lý danh mục sản phẩm hàng hóa trong kho
- Quản lý xuất nhập kho
- Báo cáo thống kê: số lượng hàng hóa trong kho, báo cáo xuất kho, báo cáo nhập kho.
- Phân quyền sử dụng tính năng của chương trình theo các cấp độ sau:
  - Nhân viên nhập/xuất hàng: sử dụng cửa sổ nhập, xuất hàng
  - Quản trị viên: sử dụng tất cả tính năng của chương trình.

Xây dựng hệ thống ứng dụng trên nền tảng Web - ASP.NET

Khóa học, xây dựng hệ thống ứng dụng trên nền tảng web , tập trung phát triển kỹ năng, kỹ thuật phát triển hệ thống ứng dụng Web sử dụng ASP.NET.

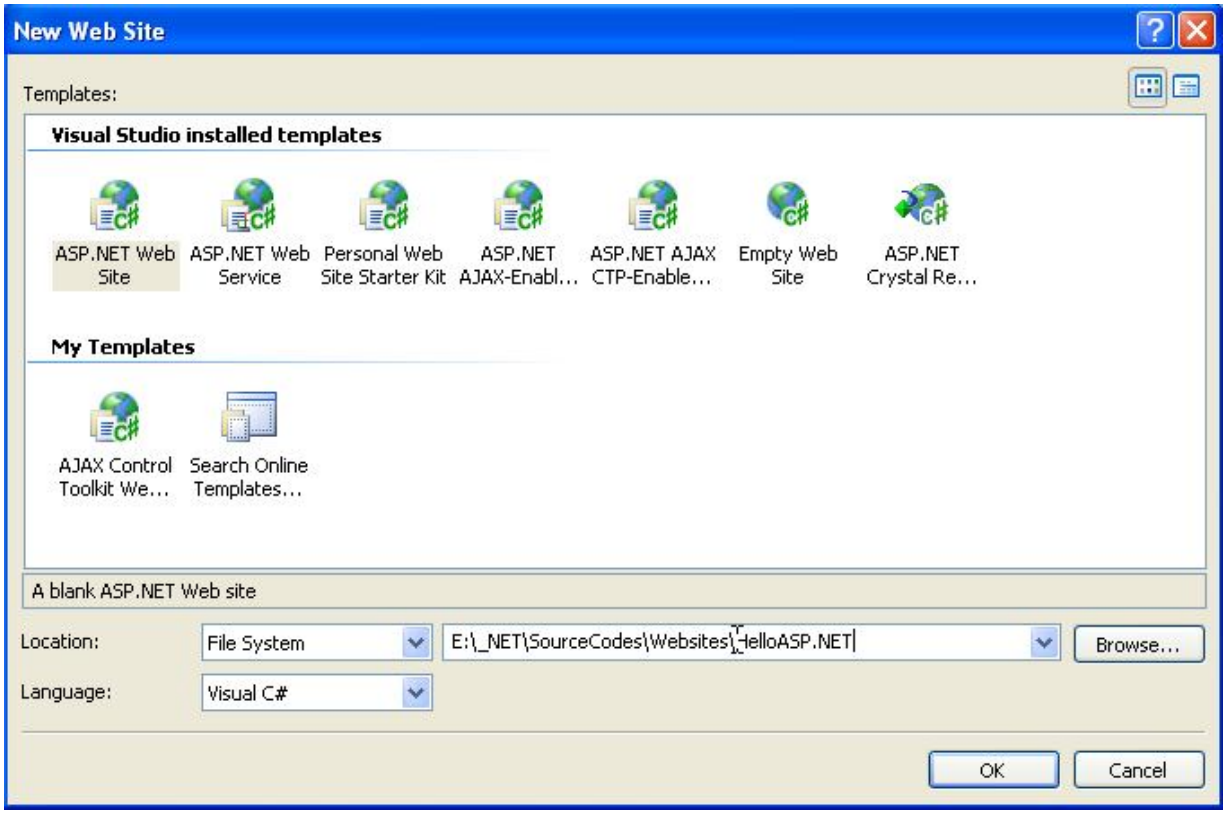
Đi tới đây, bạn đã thành thạo về lập trình OOP với C#, thao tác với cơ sở dữ liệu thông qua ADO.NET. Do vậy, từ phần này, tôi hướng bạn đến những ví dụ cụ thể là những tình huống cần xử lý trong công việc mà không chú trọng nhiều đến lý thuyết.

Mục tiêu sau khi kết thúc

- Nắm vững mô hình phát triển web động với ASP.NET.
- Hiển thị, truy vấn dữ liệu trong CSDL từ trang web
- Xây dựng Website phục vụ yêu cầu khách hàng.
- Xây dựng ứng dụng doanh nghiệp trên nền tảng web.

Xây dựng ứng dụng Hello ASP.NET Sử dụng Visual Studio .NET IDE

Mở ứng dụng Visual Studio .NET 2005 bằng cách vào menu Start → All Programs → Visual Studio 2005 → Visual Studio 2005. Sau đó chọn Menu File → New → Website.



Cửa sổ New Web Site xuất hiện cho phép khởi tạo các loại webiste khác nhau

Visual Studio cung cấp nhiều loại website mẫu, chúng ta quan tâm đến các kiểu mẫu website sau:

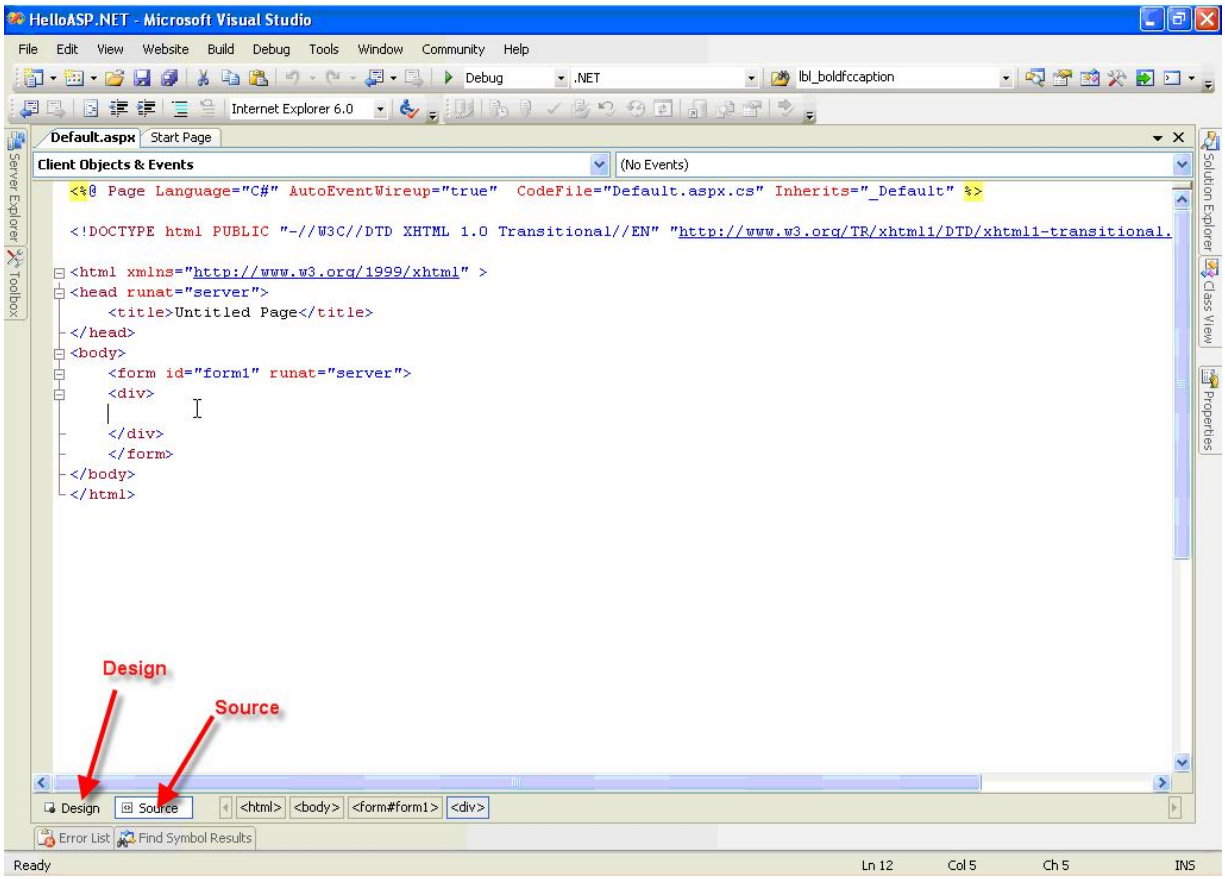
- **ASP.NET Webiste** - đây là kiểu mặc định, kiểu template này cho phép xây dựng một ứng dụng website trên nền tảng ASP.NET.
- **ASP.NET Web Service** - kiểu template này cho phép xây dựng một dịch vụ xử lý trực tuyến trên nền tảng Internet.

Ở khung cửa sổ **Templates**, chọn **ASP.NET Web Site**.

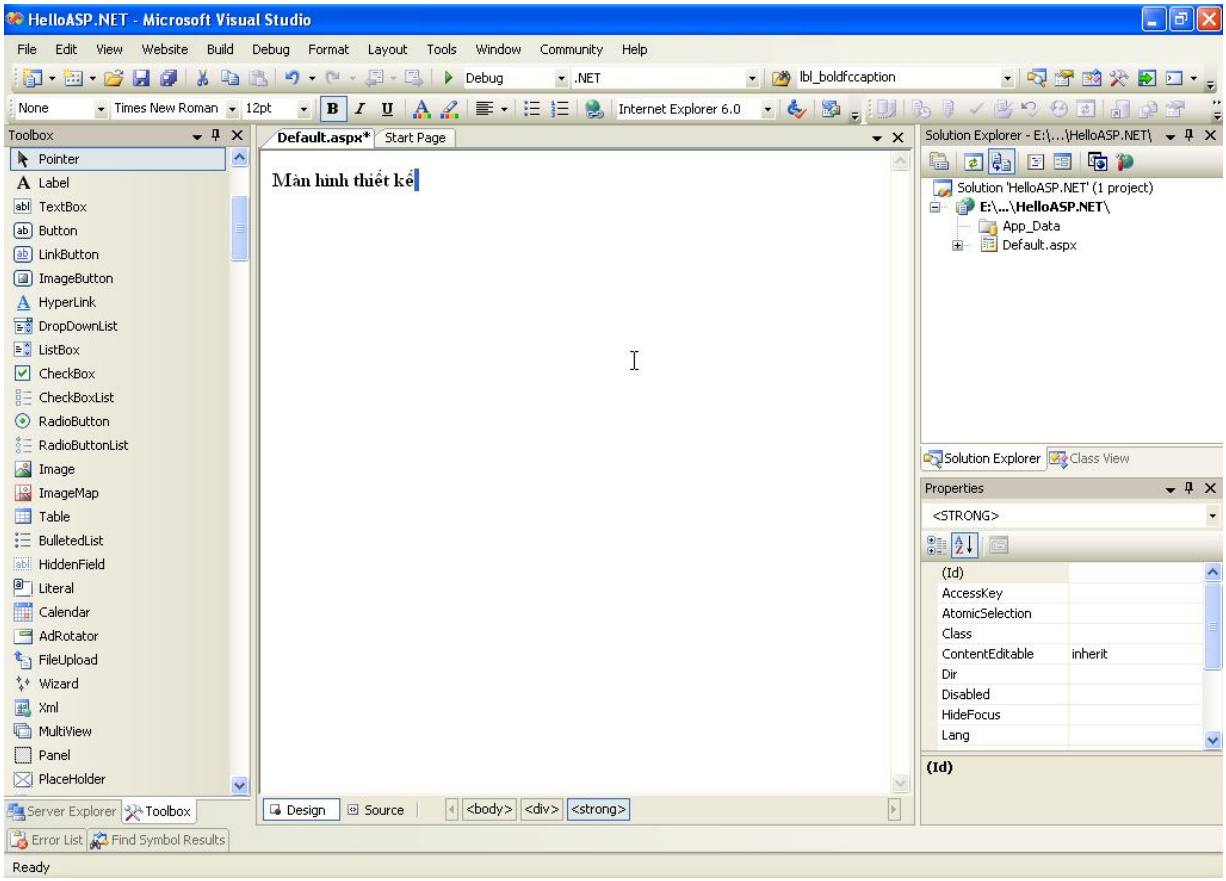
Ở Khung **Location**, chọn thư mục lưu trữ website, bạn có thể thay đổi địa chỉ thư mục lưu trữ bằng cách nhấn vào nút Browse...

Ở khung **Language**, chọn ngôn ngữ lập trình Visual C#.

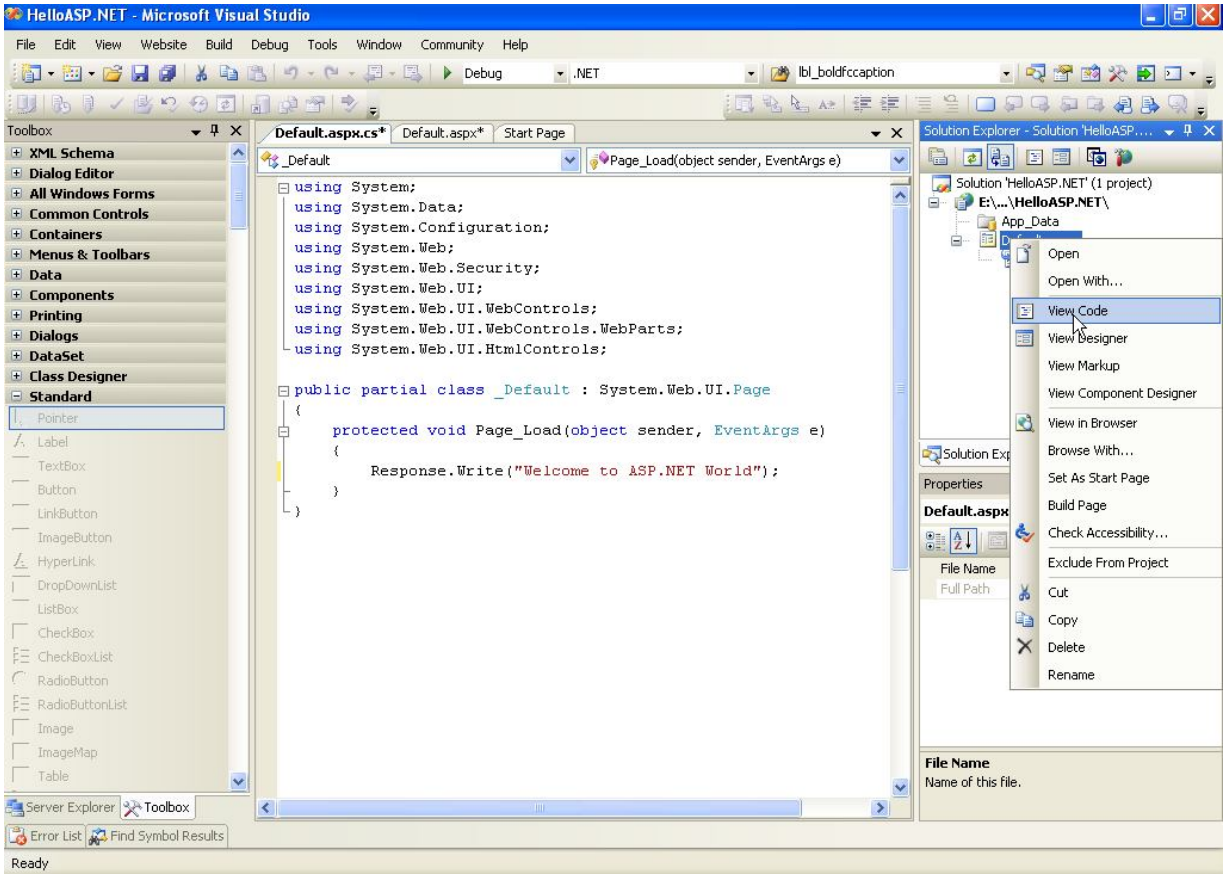
Nhấn **OK** để bắt đầu. Cửa sổ thiết kế website xuất hiện:



Đây là màn hình soạn thảo cấu trúc cho website cần xây dựng (tab **Source**), và Visual Studio cũng cung cấp một màn hình thiết kế web hoàn chỉnh, click vào tab **Design** màn hình thiết kế xuất hiện, với công cụ tại **ToolBox** và giao diện cho phép kéo thả các control vào cửa sổ thiết kế web.



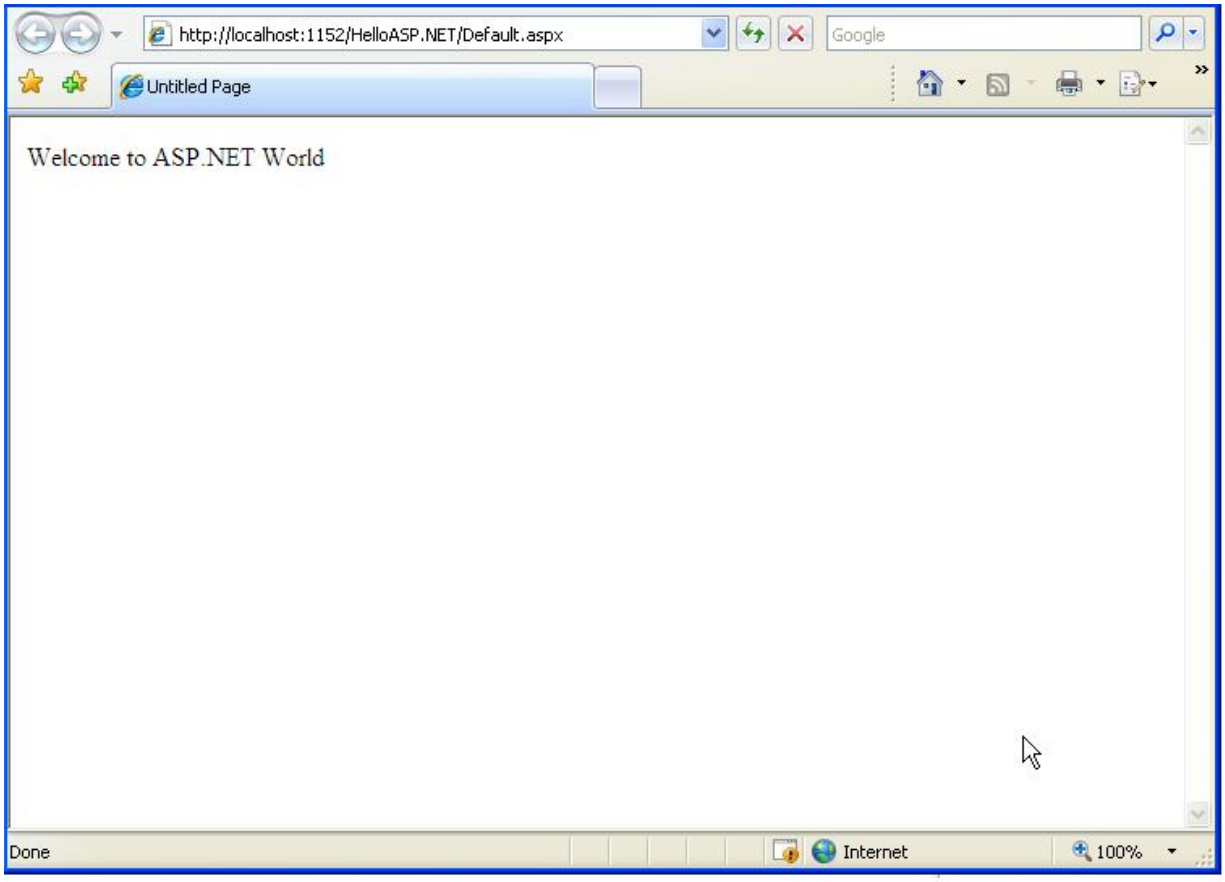
Double click vào màn hình thiết kế, hoặc tại cửa sổ Solution Explorer, chọn File Default.aspx, click chuột phải và chọn View Code, cửa sổ soạn thảo mã lệnh cho trang web xuất hiện.



Mặc định hàm Page_Load() xuất hiện, trong thân hàm này, ta viết dòng mã lệnh sau.

```
Response.Write("Welcome to ASP.NET World");
```

Nhấn **F5** để chạy chương trình. Kết quả như sau:



Ứng dụng đầu tiên của chúng ta đã hoàn tất.

Qua ứng dụng đầu tiên chúng ta có thể rút ra được một điều đó là, việc xây dựng các ứng dụng website ASP.NET giống như việc xây dựng những ứng dụng Window được xây dựng bằng Visual Studio .NET. và bạn hãy hình dung việc xây dựng website cũng giống như việc xây dựng một ứng dụng Windows mà chúng ta đã học ở phần trước.

Sự kiện Page_Load()

Sự kiện Page_Load() cũng giống như hàm Main() trong chương trình Windows Form. Hàm Page_load được gọi đầu tiên khi một trang được tải về trình duyệt của người sử dụng. Sự kiện Page_Load như sau

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
```

```
        // Mã lệnh thực thi đầu tiên khi trang được tải.
    }
}
```

Page_Load được tự động sinh ra khi double click vào cửa sổ thiết kế.

Các đối tượng ASP.NET

ASP.NET có một số đối tượng đặc biệt cần quan tâm được thể hiện như sau

Đối tượng Request

Ví dụ: khi bạn truy cập website itgatevn.com.vn, bạn sẽ thấy trên thanh địa chỉ web như sau
http://www.itgatevn.com.vn
Khi bạn vào một bài tin nào đó, thanh địa chỉ có dạng như sau:
http://www.itgatevn.com.vn/index.aspx?u=nd&scid=14&nid=4442

Việc chuyển thành dạng thứ hai có ý nghĩa rằng người sử dụng đang xem chi tiết một tin với mã số nid=4442, và chương trình sẽ lấy đúng bản tin với mã số trên để cung cấp cho người sử dụng, và để lấy mã số này từ thanh địa chỉ trên trình duyệt, ASP.NET cung cấp đối tượng Request. Đối tượng Request gồm có các thuộc tính

Thuộc tính	Mô tả
ApplicationPath	Đây là thuộc tính chỉ đọc, cung cấp đường dẫn của ứng dụng ASP.NET trên máy chủ triển khai.
Url	Thuộc tính chỉ đọc, trả lại đối tượng URI chứa đựng địa chỉ hoàn chỉnh của một yêu cầu.
UserHostAddress	Trả lại địa chỉ IP của máy người sử dụng
Browser	Trả lại đối tượng HttpBrowserCapabilities chứa những thông tin về khả năng của browser trên máy người dùng.
Cookies	Trả về đối tượng HttpCookiecollection cung cấp những biến cookie của client.
QueryString	Cung cấp khả năng truy cập đến những tham số trên thanh địa chỉ.
Form	Cung cấp khả năng lấy dữ liệu được gửi từ Form dữ liệu

Ví dụ về cách sử dụng các thuộc tính của đối tượng Request

```
protected void Page_Load(object sender, EventArgs e)
    {
        Response.Write("Application Path: " + Request.ApplicationPath);
        Response.Write("<br>Url: " + Request.Url.ToString());
        Response.Write("<br>User Host Address: " + Request.UserHostAddress);
        Response.Write("<br>Browser Name: " + Request.Browser.Browser);
        Response.Write("<br>Browser Version: " + Request.Browser.Version);
    }
```

Ví dụ về cách sử dụng thuộc tính Form của đối tượng Request để nhận dữ liệu từ Form.

```
<input type="checkbox" id="chk" name="chk" value="1"/>
```

Mã lệnh lấy giá trị của điều khiển checkbo "chk" từ Form.

```
protected void Button_Click(object sender, EventArgs e)
    {
        // Nhận giá trị của checkbox trong form.
        string checkvalue = Request.Form["chk"];
    }
```

Đối tượng Response

Đối tượng Response có nhiệm vụ gởi thông tin đến trình duyệt người sử dụng (Client). Những thuộc tính và hàm thường sử dụng của đối tượng Response được mô tả như bảng sau

Tên	Mô tả
ApplicationPath	Đây là thuộc tính chỉ đọc, cung cấp đường dẫn của ứng dụng ASP.NET trên máy chủ triển khai.
Cache	Cung cấp những thuộc tính về khả năng lưu trữ như thời gian hết hạn, cài đặt riêng của website. Đây là thuộc tính chỉ đọc.
ContentType	Cho phép gán hoặc nhận kiểu dữ liệu được truyền đến trình duyệt người sử dụng (Client) thông qua đối tượng Response.
Cookies	Cho phép gán một cookie vào trình duyệt người sử dụng
IsClientConnected	Trả lại giá trị đúng/sai xác định khi nào người dùng kết nối hoặc không kết nối đến ứng dụng.
public void Redirect(string)	Hàm cho phép chuyển đến một địa chỉ được xác định trong tham số truyền vào đến trình duyệt người sử dụng
public void Write(string)	Được sử dụng để gởi một chuỗi ký tự xuất hiện ra màn hình.

Đối tượng Session

Đối tượng Session cho phép ứng dụng ASP.NET lưu trữ trạng thái của người sử dụng. Session bắt đầu và kết thúc khi người sử dụng kết nối và tắt trình duyệt đến một trang web, session cũng được tự động tắt khi người dùng không sử dụng trang web trong một thời gian xác định, thời gian mặc định là 20 phút.

Tên	Mô tả
Count	Trả về số lượng session trong Session-state
SessionID	Trả về một session xác định trong Session-state
Timeout	Cho phép gán hoặc nhận thời gian tự động hủy bỏ session tính bằng phút.

Tên	Mô tả
public void Add(string,object)	Hàm cho phép thê một item vào Session-state
public void Clear()	Được sử dụng để xóa toàn bộ giá trị được lưu trữ trong Session-State.
public void Remove(string)	Được sử dụng để loại bỏ một item trong Session-state

Ví dụ gán giá trị cho biến Session

```
// Gán giá trị kiểu số cho Session
Session[“intvalue”] = 1;
// Gán giá trị kiểu chuỗi cho Session
Session[“stringvalue”] = “Welcome to ASP.NET Session”;
// gán giá trị kiểu DataTable cho Session
DataTable dt = new DataTable();
dt.Columns.Add("col1");
dt.Columns.Add("col2");
dt.Columns.Add("col3");
Session["datatablevalue"] = dt;
```

Ví dụ lấy giá trị từ biến Session

```
// Lấy giá trị kiểu integer
int ivalue = Convert.ToInt32(Session["intvalue"]);
// Lấy giá trị từ Session
string svalue = Session["stringvalue"].ToString();
// Lấy giá trị kiểu DataTable từ Session
DataTable dtvalue = new DataTable();
dtvalue = (DataTable)Session["datatablevalue"];
```

Xây dựng ứng dụng Web sử dụng Server Controls

Server Controls

Tôi sẽ không giải thích với các bạn cặn kẽ thế nào là Server Controls, các bạn có thể tham khảo khái niệm này tại những tài liệu về ASP.NET khác, các bạn hãy hình dung Server Control cũng giống như các Control trong ứng dụng Windows Form. Tôi giới thiệu với các bạn cách thức sử dụng một số loại Server Control sau đây.

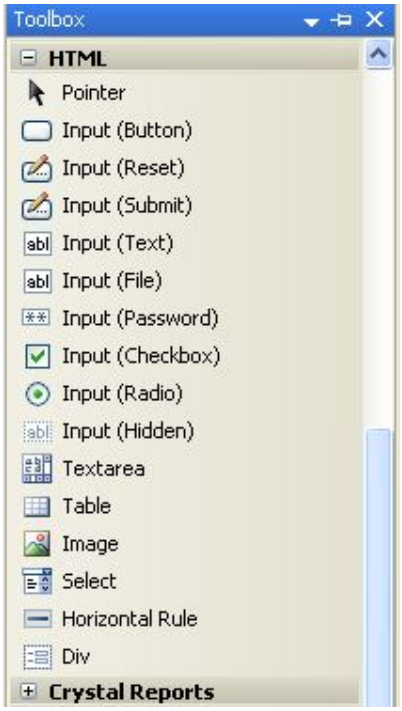
HTML Server Controls

Html Controls được gắn kèm vào ASP.NET để đáp ứng khả năng chuyển đổi từ ASP thông thường sang nền tảng ASP.NET. HtmlControls là những đối tượng thuộc về namespace System.Web.UI.HtmlControls. Sau đây là một số HtmlControls thường sử dụng.

Html Server Control	Mã lệnh
HtmlForm	<form runat=server></form>
HtmlInputText	<input type=text runat=server /> và <input type=password runat=server />
HtmlInputCheckBox	<input type=checkbox runat=server />
HtmlInputRadioButton	<input type=radio runat=server />

Html Server Control	Mã lệnh
HtmlInputImage	<input type=image runat=server />
HtmlAnchor	
HtmlButton	<input type=button runat=server />
HtmlTable	<table id="table1" runat=server></table>
HtmlTableRow	<tr></tr>
HtmlTableCell	<td></td>

Bạn có thể tìm thấy những control này trong cửa sổ ToolBox tại HTML Tab.



HTML Tab trong ToolBox

Chúng ta sẽ bắt đầu với một số HTML Control quan trọng

HtmlAnchor

Điều khiển HtmlAnchor làm việc như tag <a> của HTML nhưng chạy tại máy chủ, HtmlAnchor được sử dụng để chuyển người dùng từ một trang đến một trang khác.

Ví dụ:
Trong đó,

- id=myanchor1 - là tên duy nhất của một control trên một trang
- href="http://www.itgatevn.com.vn" - là địa chỉ mà anchor sẽ chuyển người dùng đến, ở đây là trang itgatevn có địa chỉ http://www.itgatevn.com.vn

HtmlInputText

HtmlInputText làm việc như thẻ HTML <input type=Text>, nhưng chạy tại máy chủ. HtmlInputText được sử dụng để nhận dữ liệu từ người dùng.

Ví dụ: `<input id=txtName1 size=12 type=text runat=server />`
Trong đó,

- size=12 - giá trị số ký tự lớn nhất có thể nhập vào textbox.

HtmlInputCheckBox

HtmlInputCheckBox làm việc như thẻ HTML <input type=checkbox>, nhưng chạy tại máy chủ. HtmlInputCheckBox được sử dụng để thực hiện câu hỏi với trả lời là Có hoặc Không.

Ví dụ: `<input id=chkCoffeeorNot type=checkbox checked runat=server />`

HtmlInputRadioButton

HtmlInputRadioButton làm việc như thẻ HTML <input type=radio>, nhưng chạy tại máy chủ. HtmlInputRadioButton được sử dụng để thực hiện một sự lựa chọn khi có nhiều lựa chọn được đưa ra cho người dùng chọn lựa.

Ví dụ: `<input id=radioChooseOne type=radio checked runat=server />`

HtmlSelect Control

HtmlSelect làm việc như thẻ HTML <select>, nhưng chạy tại máy chủ. HtmlSelect được sử dụng để tạo ra một danh sách cho phép người sử dụng lựa chọn một trong danh sách đó.

Ví dụ:

```
<select runat=server>
   <option>MU</option>
   <option>Asenal</option>
   <option>Bonton</option>
   <option>Chelsea</option>
</select>
```

Web Server Controls

WebServer Controls là bộ công cụ được cung cấp trong namespace System.Web.UI.WebControls. Một số WebControls được mô tả theo bảng sau:

WebControls	Mô tả
WebControls cơ bản	Tương đồng như những HtmlControls bình thuwofng như: TextBox, Label, Button, HyperLink, RadioButton, và CheckBox.
ListControls	được sử dụng để xây dựng một danh sách, và cho phép nhận dữ liệu từ cơ sở dữ liệu, những control thuộc dạng này bao gồm: ListBox, DropDownList, CheckBoxList và RadioButtonList.
RichWebControls	Có các điều khiển như Calendar, AdRotator, TreeView, mỗi control có một tính năng riêng.
Data Controls	Được sử dụng để hiển thị dữ liệu từ một bảng trong cơ sở dữ liệu, bao gồm những Control: DataGrid, DataList, và Repeater.

Bạn có thể tìm thấy những Control này tại ToolBox.

Với những WebForm Control, bạn hãy hình dung chúng nhưng những Control trên môi trường Window Form. Với những Web Control, ASP.NET cho phép bạn xây dựng những

ứng dụng mạnh mẽ trên môi trường internet tương tự như trên môi trường ứng dụng để bàn.

TextBox Control

Được sử dụng để nhận dữ liệu từ người sử dụng, như ký tự, số, ngày tháng … Thuộc tính quan trọng nhất của TextBox được liệt kê như sau:

Thuộc tính	Mô tả
Text	Cho phép gán giá trị hiển thị trong textbox hoặc nhận giá trị do người dùng nhập vào.
TextMode	Có ba giá trị tương ứng với 3 dạng khác nhau của TextBox - Single Line - dữ liệu hiển thị trên 1 dòng. - Password - Sử dụng cho ô mật khẩu. - Multiline - dữ liệu hiển thị trên nhiều dòng.

Literal Control

Literal được sử dụng để hiển thị dữ liệu mà không quan tâm đến định dạng của dữ liệu ấy. Điều khiển này thích hợp để hiển thị dữ liệu dưới dạng nội dung, do đó, thuộc tính chính của Literal đó là thuộc tính Text. Bạn có thể tìm thấy điều khiển Literal dưới biểu tượng  trên thanh Toolbox.

Ví dụ sử dụng Literal

```
Literal1.Text = "This is literal contron";
```

FileUpload Control

FileUpload là thuộc tính được sử dụng để cho phép tải một tệp tin bất kỳ lên máy chủ web. Đây là điều khiển được tích hợp vào các phiên bản ASP.NET 2.0 trở về sau. Bạn có thể tìm thấy điều khiển này tại thanh công cụ với biểu tượng 

Ví dụ sử dụng File Upload

```
// kiểm tra có file upload
        if (FileUpload1.FileName.Length > 0)
        {
            // kiểm tra nội dung của file
            if (FileUpload1.PostedFile.ContentLength > 0)
            {
                // lưu file vào thư mục uploadfile
                FileUpload1.PostedFile.SaveAs("uploadfile/" +
System.IO.Path.GetFileName(FileUpload1.FileName));

            }
        }
```

Panel Control

Panel là điều khiển được sử dụng để chứa những nhóm điều khiển cho một chức năng riêng, Panel cho phép điều khiển việc hiển thị những điều khiển được chứa trong nó.

Các bạn xem hướng dẫn để sử dụng Panel trong file kèm theo tài liệu này.

View & MultiView Control

Multiview là điều khiển chứa những điều khiển con là View, View cho phép chứa tất cả các điều khiển khác. Chúng ta sử dụng View và MultiView khi chúng ta trong một số trường hợp khi cần thực hiện một công việc theo nhiều giai đoạn của tác vụ (các bước theo tuần tự), hoặc một tác vụ với nhiều giao diện khác nhau. Hai điều khiển này luôn đi chung với nhau. Trong một thời điểm, bạn chỉ xác định được duy nhất một View được hiển thị thông qua thuộc tính ActiveViewIndex của điều khiển MultiView.

Các bạn xem hướng dẫn để sử dụng View và Multiview trong file video kèm theo tài liệu này.

Calendar Control

Calendar được sử dụng để hiển thị ngày tháng. Người sử dụng có thể xem ngày tháng theo tuần hoặc tháng. Những thuộc tính thường sử dụng bao gồm

Thuộc tính	Mô tả
DayNameFormat	Thuộc tính này được sử dụng để xác định định dạng tên của những ngày trong tuần.
VisibleDate	Được sử dụng để xác định tháng được xuất hiện hay không trong Calendar. Thuộc tính này được cập nhật sau khi sự kiện VisibleMonthChanged được thực hiện.
FirstdayOfWeek	Thuộc tính này được sử dụng để xác định ngày của tuần được xuất hiện tại cột đầu tiên của điều khiển Calendar.
SelectedDate	Thuộc tính này được dùng để biểu diễn cho ngày được chọn trong điều khiển Calendar.
SelectionMode	Thuộc tính này được sử dụng để xác định khi nào một người dùng có thể chojnmoojt ngày, một tuần, hoặc một tháng. Mặc định, giá trị của thuộc tính này là Day.

DropDownList Control

Như Select Control trong Html, DropDownList Control cho phép người sử dụng chọn một mục trong danh mục được địnhnghĩa trước. Mỗi mục là một đối tượng riêng biệt có các thuộc tính riêng như Text, Value và Selected. Bạn có thể thêm mục vào DropDownList thông qua thuộc tính Items lúc thiết kế hoặc ngay khi chương trình chạy.

Mã lệnh thêm mục cho DropDownList khi chương trình đang chạy

```
protected void Page_Load(object sender, EventArgs e)
```

```
    {
        this.DropDownList1.Items.Add(new ListItem("Item1", "Value1"));
        this.DropDownList1.Items.Add(new ListItem("Item2", "Value2"));
    }
```

Ngoài ra, DropDownList còn có khả năng nhận dữ liệu tự động từ cơ sở dữ liệu thông qua thuộc tính DataSource và phương thức DataBind().

Mã lệnh gán DataSource cho DropDownList khi chương trình đang chạy

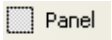
```
protected void Page_Load(object sender, EventArgs e)
    {
        this.DropDownList1.DataSource = ds; // ds là một DataSet chứa dữ liệu
được lấy từ Cơ sở dữ liệu.
        this.DropDownList1.DataBind();
    }
```

Các công cụ khác không được giới thiệu ở đây nhưng cách thức sử dụng công cụ được biểu diễn trong file hướng dẫn: PTA_NET_2_0_WEB_CONTROLS.WMV và ví dụ kèm theo tài liệu.

Thêm điều khiển khi chương trình chạy.

Đôi khi, bạn cần thêm những điều khiển vào trang web khi chương trình đang chạy, ASP.NET cung cấp khả năng này cho nhà phát triển. Trước tiên bạn cần xác định chỗ để đặt điều khiển, có hai công cụ bạn có thể sử dụng để đặt chỗ chứa đựng các control cần tạo.

- PlaceHolder 

- Panel 

Ví dụ sau sẽ giúp bạn cách thêm một Control lúc chương trình đang chạy.

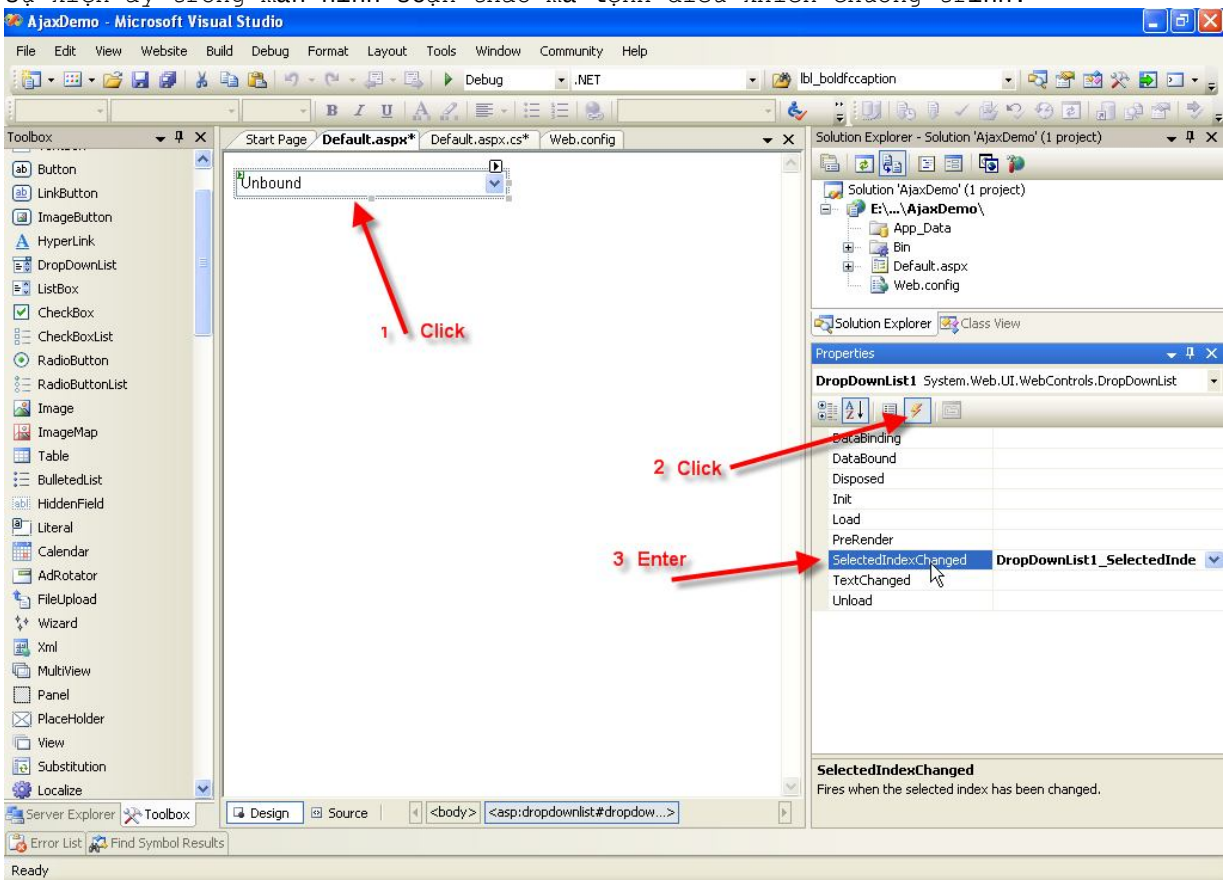
```
protected void Page_Load(object sender, EventArgs e)
    {
        Calendar cl = new Calendar();
        PlaceHolder1.Controls.Add(cl);  //  một  calendar  đã  được  thêm  vào
Placeholder trên trang.
    }
```

Chúng ta sẽ sử dụng kỹ hơn PlaceHolder Control ở trong phần thực hiện kiến trúc WebPortal trong phần sau.

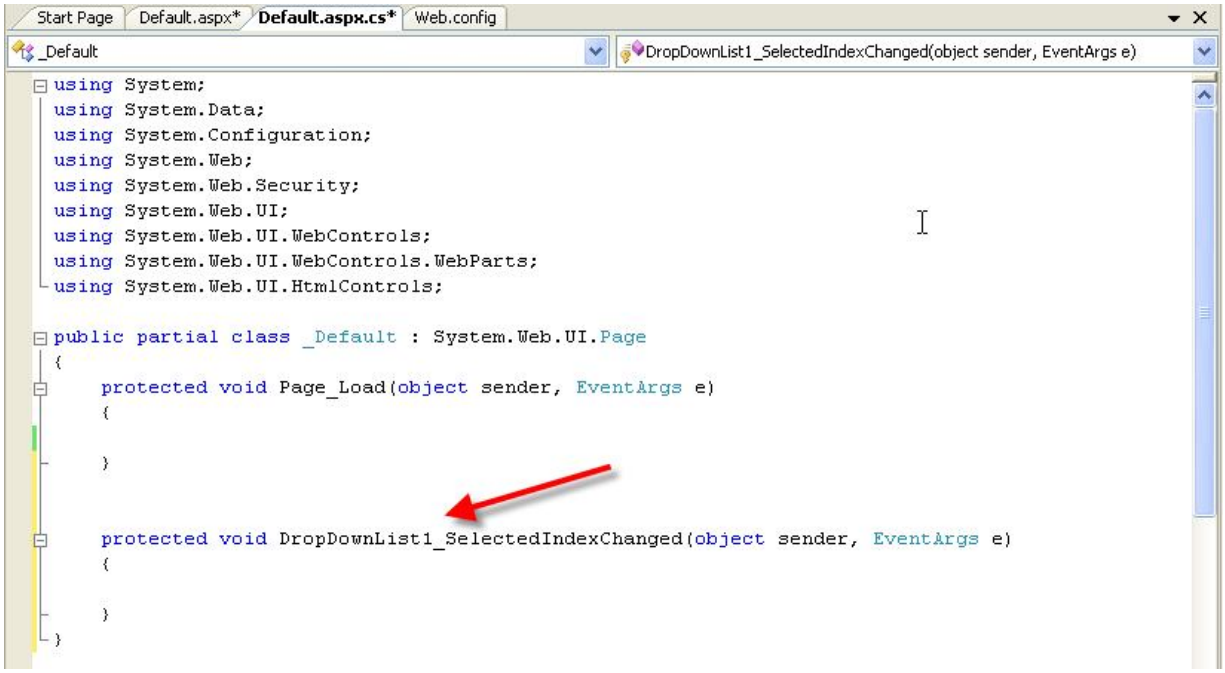
Điều khiển các Server Controls

ASP.NET cung cấp cho nhà phát triển những sự kiện đi kèm với mỗi điều khiển, ví dụ sau sẽ hướng dẫn bạn cách thức xem sự kiện, và sử dụng những sự kiện mà ASP.NET cung cấp.

Tạo mới một ứng dụng Website, tại màn hình thiết kế, kéo thả vào cửa sổ thiết kết một DropDownList. Chọn DropDownList ấy và mở cửa sổ Properties. Trên cửa sổ Properties của DropDownList, nhấn vào biểu tượng  , các sự kiện của điều khiển này xuất hiện, chọn một sự kiện và nhấn Enter, VisualStudio tự động phát sinh ra sự kiện ấy trong màn hình soạn thảo mã lệnh điều khiển chương trình.



Màn hình sự kiện xuất hiện như sau:



Từ lúc này, khi người dùng chọn thay đổi một mục tại DropDownList, chương trình sẽ kiểm tra thực hiện thao tác này thông qua sự kiện DropDownList1_SelectedIndexChange().

Đây cũng là cách thức chung để xem mỗi Control có bao nhiêu sự kiện và kiểm soát số sự kiện đã khai báo cho mỗi Control.
Xem ví dụ thực hiện điều khiển Control PTA_NET_2_0_Control_CONTROLS.WMV

Bài tập 1

Tạo một ứng dụng Web cho phép người sử dụng nhập Email, sau đó hiển thị Email vừa nhập lên màn hình.

Hướng dẫn

Bạn hãy thực hiện các bước sau:

- Tạo một ứng dụng Web
- Hiển thị cửa sổ thiết kết.
- Kéo thả một TextBox Control vào cửa sổ thiết kế.
- Kéo thạt một Button Control vào cửa sổ thiết kế.
- Chọn ButtonControl, mở cửa sổ Properties, chọn  , chọn sự kiện Click, nhấn Enter, sự kiện Click được xuất hiện trong cửa sổ soạn thảo lệnh điều khiển chương trình, soạn thảo đoạn mã lệnh sau

```
Response.Write(this.TextBox1.Text);
```

- Nhấn F5 để thực thi chương trình.

Bài tập 2

Thực hiện một ứng dụng website cho phép người sử dụng khai báo thông tin cá nhân bao gồm: họ và tên, giới tính, ngày sinh, địa chỉ, điện thoại, thư điện tử (email), nghề nghiệp, cơ quan (công ty hoặc trường học), và thông tin giới thiệu về bản thân. Sau đó lưu trữ vào cơ sở dữ liệu.

Kết nối cơ sở dữ liệu trong ASP.NET

Dữ liệu là một phần không thể thiếu trong việc xây dựng một hệ thống ứng dụng Web. Như một website bán hàng, họ phải cập nhật thông tin sản phẩm mới sản xuất để khách hàng lựa chọn, tiếp nhận đơn đặt hàng ngày giờ, và xử lý các đơn đặt hàng ấy. Nói chung, bất kỳ ứng dụng web nào cũng cần phải có dữ liệu, thông tin được cập nhật thường xuyên. Ở phần này, chúng ta sẽ làm việc với cơ sở dữ liệu (CSDL) trên môi trường ứng dụng Web ASP.NET.

DataBinding trong ASP.NET

Data Binding là một kỹ thuật kết nối dữ liệu với những đối tượng. Sử dụng data binding, bạn có thể nối dữ liệu trong một nguồn dữ liệu đến một đối tượng người dùng. Mọi việc thay đổi trên các đối tượng giao diện người dùng có thể trực tiếp được cập nhật vào nguồn dữ liệu. Có hai kiểu binding dữ liệu đó là **Simple Data Binding** và **Complex Data Binding** . Chúng ta quan tâm đến kiểu thứ hai - Complex Data Binding - với nguồn dữ liệu được truy xuất từ cơ sở dữ liệu.

Để Binding dữ liệu giữa CSDL và một Control, các bạn hãy quan tâm tới những bước sau

- Lấy dữ liệu từ cơ sở dữ liệu và đổ vào một đối tượng lưu trữ (thường là DataSet, DataTable).
- Khai báo nguồn dữ liệu (DataSource) cho đối tượng giao diện người dùng (Web Control).
- Thực hiện lệnh DataBinding() để hiện thực quá trình binding dữ liệu.

Kết nối cơ sở dữ liệu chúng ta đã học ở các phần trước, do đó hai bước: khai báo nguồn dữ liệu và thực hiện lệnh DataBinding() sẽ được tìm hiểu sâu và kỹ để bạn có thể nắm rõ.

Binding dữ liệu vào một DropDownList Control

Tạo mới một ứng dụng Web, bằng cách vào menu **File → New → Website**.

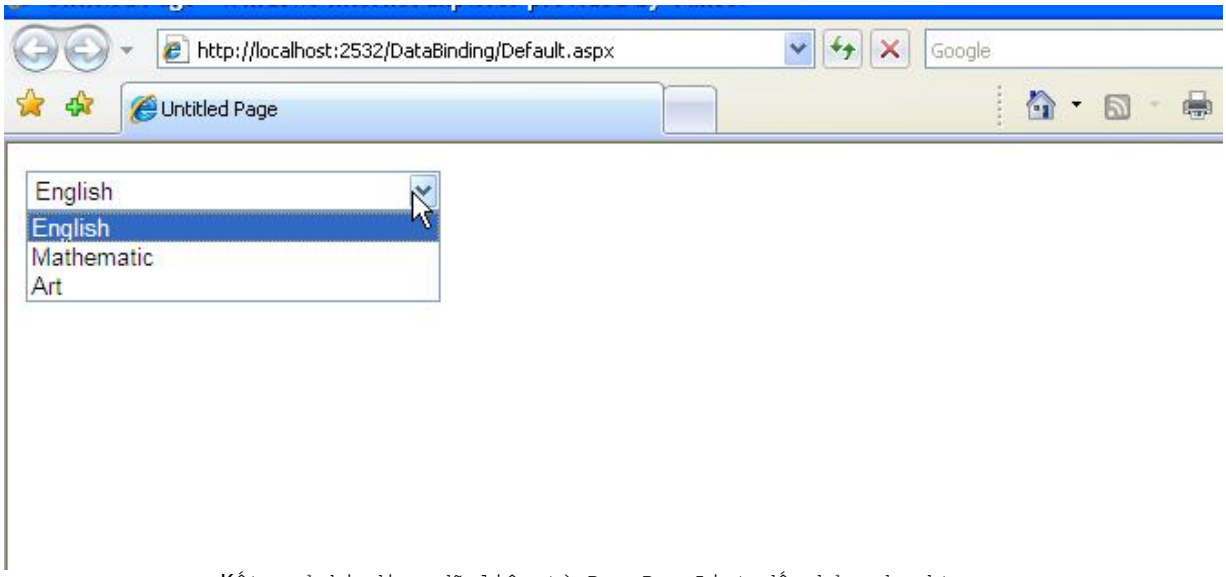
Cửa sổ thiết kết website xuất hiện, kéo thả một **DropDownList Control** từ cửa sổ **ToolBox** vào màn hình thiết kế, đặt tên cho Dropdownlist là ddltypes

Double click vào màn hình thiết kế, Form_Load() xuất hiện, bạn thực hiện những đoạn mã lệnh sau

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.OleDb;

public partial class _Default : System.Web.UI.Page
{
    string str_connection =
"Provider=SQLOLEDB;server=(local);database=_NET;uid=_net;pwd=";

    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
```

```
        {
            OleDbConnection conobj = new OleDbConnection(str_connection);
            conobj.Open();
            OleDbCommand cmd = new OleDbCommand("select * from booktypes",
conobj);
            OleDbDataAdapter da = new OleDbDataAdapter(cmd);
            DataSet ds = new DataSet();
            da.Fill(ds, "booktypes");
            // xác định DataSource để bind dữ liệu
            this.ddltypes.DataSource = ds.Tables["booktypes"];
            this.ddltypes.DataTextField = "vbooktypename";
            this.ddltypes.DataValueField = "itypeid";
            // Gọi hàm DataBind() thực hiện bind dữ liệu giữa CSDL và
DropDownList ddltypes
            this.ddltypes.DataBind();
            conobj.Close();
        }
    }
}
```

Chạy ứng dụng, kết quả sẽ như sau:



Kết quả binding dữ liệu từ DropDownList đến bảng booktypes

Dữ liệu từ bảng BookTypes trong cơ sở dữ liệu được đổ vào trong điều khiển DropDownList **ddltypes**.

Thuộc tính IsPostBack

Thuộc tính IsPostBack cho phép kiểm tra khi nào trang được load lại. Thuộc tính này được sử dụng trong việc tải những dữ liệu cố định trên một trang hoặc một điều khiển.

Web Server Control Template

Template là một sự kết nối nững thành phần HTML, Controls và những Server Controls được gắn kèm, cho phép dùng để thay đổi và thao tác trên giao diện của điều khiển.
ASP.NET cung cấp ba Controls để hiển thị dữ liệu cho phép thay đổi giao diện hiển thị: DataGrid, Repeater, và DataList. Giao diện của những điều khiển này được thay đổi bằng cách sử dụng template.

Ví dụ với một danh sách học sinh, như sau:

STT	Họ và Tên	Giới tính	Ngày sinh	Địa chỉ
1	Trần Tuấn Anh	Nam	12/12/1985	Hà Nội
2	Nguyễn Hoài Nam	Nữ	1/3/1986	Tp.Hồ Chí Minh
…				
Tổng số 20 học sinh				

Cần phải có phần đầu, phần thân, và phần cuối của một danh sách, trong đó phần đầu mô tả những nội dung được thể hiện trong phần thân, và phần cuối là tổng kết cho danh sách ở phần thân.

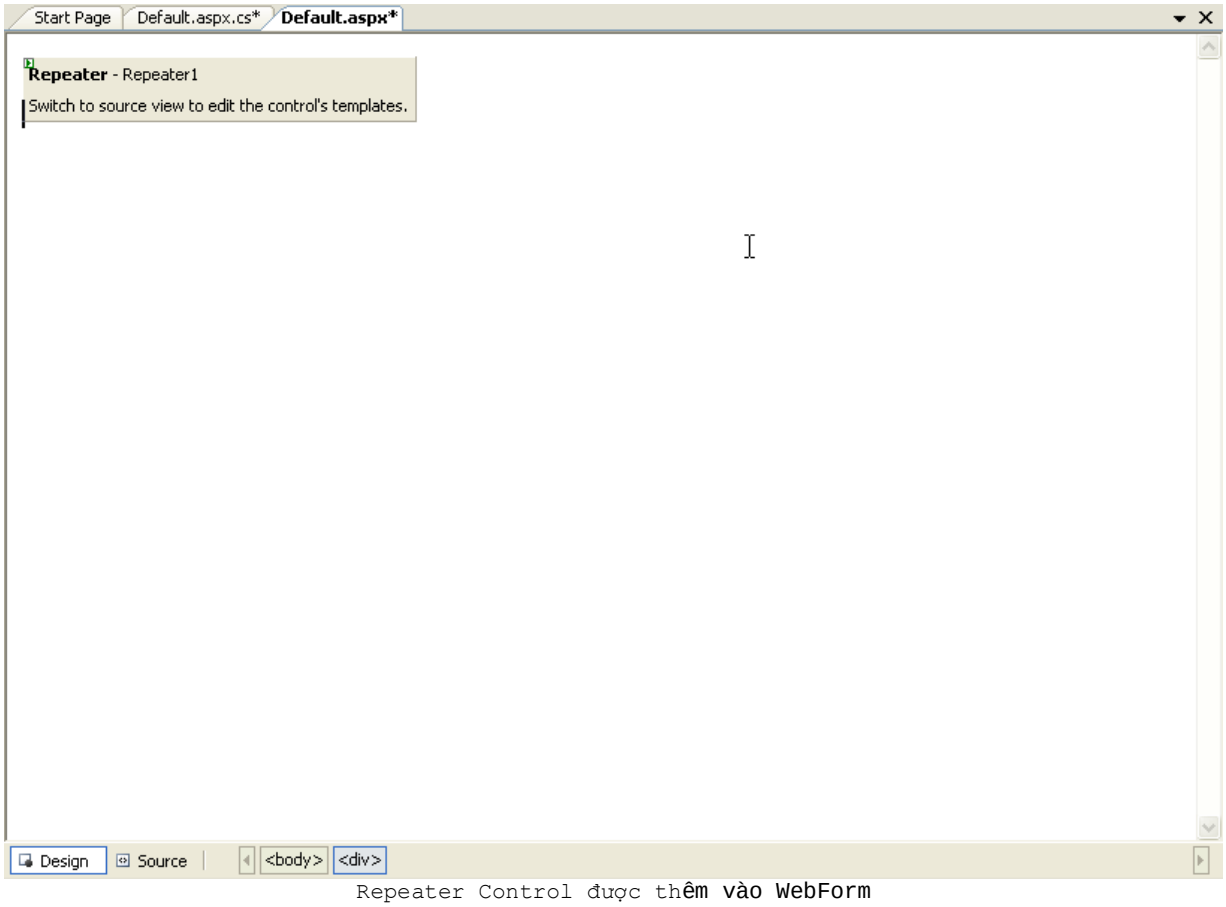
Template trong ASP.NET cũng như vậy, có ba phần chính:

- HeaderTemplate: định dạng cho phần đầu của danh sách
- ItemTemplte: định dạng cho phần thân của danh sách
- FooterTemplate: định dạng cho phần cuối của danh sách.

Ngoài ra còn có các Template khác như AlternatingItemTemplate, SelectedItemTemplate, SeparatorTemplate… chúng ta sẽ đề cập những khuôn mẫu này khi thực hiện một ví dụ cụ thể nào đó.

Repeater Control

Một Repeater control hiển thị dữ liệu từ những nguồn dữ liệu khác nhau bằng cách sử dụng những giao diện được thiết kế. Repeater không có giao diện, và được hiển thị như hình sau:



Repeater Control được thêm vào WebForm

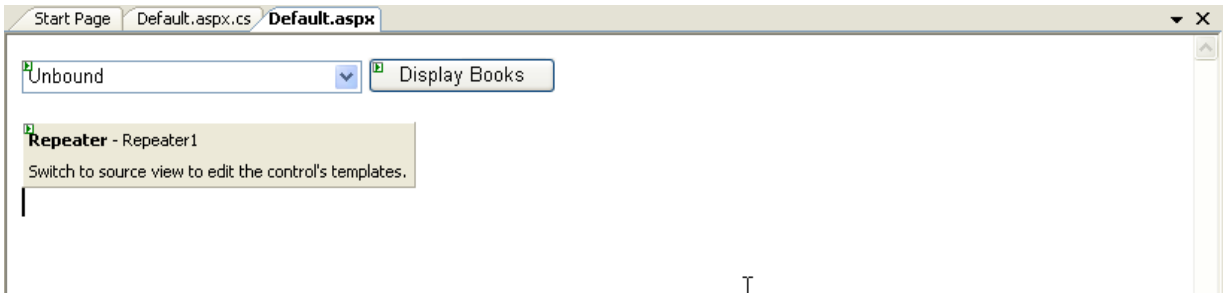
Bạn cần phải tạo template để xây dựng giao diện cho Repeater.

Ví dụ:

Quản lý nhà trường muốn xem danh sách học sinh theo lớp, bạn cần thực hiện ứng dụng để đáp ứng yêu cầu này.

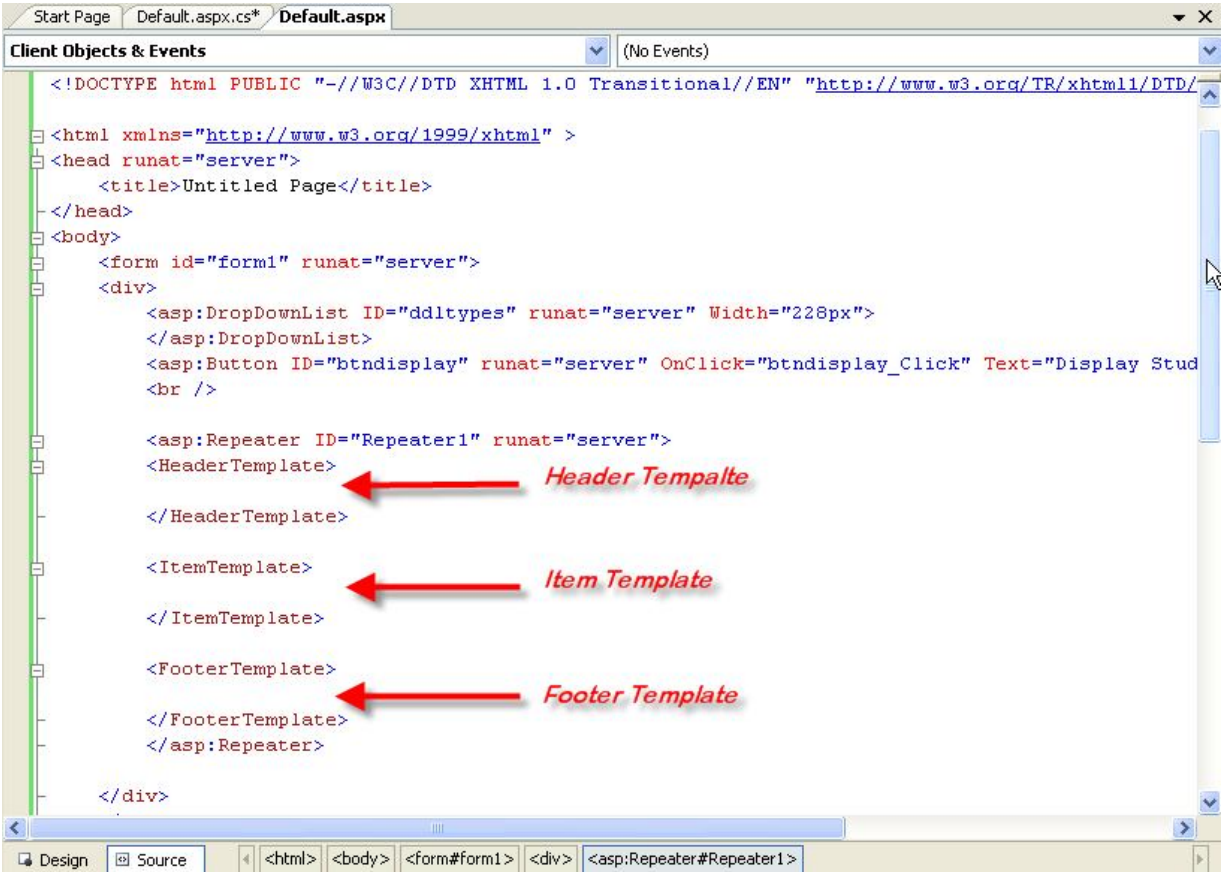
Bạn thực hiện ứng dụng như sau:

- Tạo một ứng dụng web
- Kéo thả một **DropDownList** để hiển thị danh sách lớp cho phép chọn lựa, đặt tên là **ddlclasses**
- Kéo thả một **Button** dùng để hiển thị danh sách những học sinh trong lớp được chọn tại dropdownlist, đặt tên là **btndisplay**.
- Một **Repeater** cho phép hiển thị danh sách, đặt tên là **rpstudents**



Bây giờ, bạn cần xây dựng template cho danh sách học sinh. Bạn quan tâm tới ba phần: **HeaderTempalte**, **ItemTempalte**, và **FooterTempalte**. Thực hiện như sau:

- Mở chế độ hiển thị soạn thảo HTML
- Trong phần thân của điều khiển repeater, bạn soạn thảo cấu trúc như mô tả sau đây:



Đó là cấu trúc chính của một Repeater, và công việc của bạn là phải xây dựng cấu trúc cho mỗi phần Header, Item, Footer để tạo thành một Template cho Repeater.

Như bảng danh sách sinh viên ở phía trên, trong ngôn ngữ HTML, nó có dạng:

```
<table border="0" cellpadding="0" style="border-collapse: collapse" width="100%"
id="table1">
	<tr>
		<td  width=50><b>STT</b></td>
		<td width=200><b>H&#7885; và Tên</b></td>
		<td width=75><b>Gi&#7899;i tính</b></td>
		<td width=125><b>Ngày sinh</b></td>
		<td><b>&#272;&#7883;a ch&#7881;</b></td>
	</tr>


	<tr>
		<td>1</td>
		<td>Trần Tuấn Anh</td>
		<td>Nam</td>
		<td>12/12/1985</td>
		<td>Hà Nội</td>
```

```
      </tr>
      <tr>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
      </tr>
</table>
```

Với phần đầu (header) là:

```
<table border="0" cellpadding="0" style="border-collapse: collapse" width="100%"
id="table1">
      <tr>
            <td  width=50><b>STT</b></td>
            <td width=200><b>H&#7885; và Tên</b></td>
            <td width=75><b>Gi&#7899;i tính</b></td>
            <td width=125><b>Ngày sinh</b></td>
            <td><b>&#272;&#7883;a ch&#7881;</b></td>
      </tr>
```

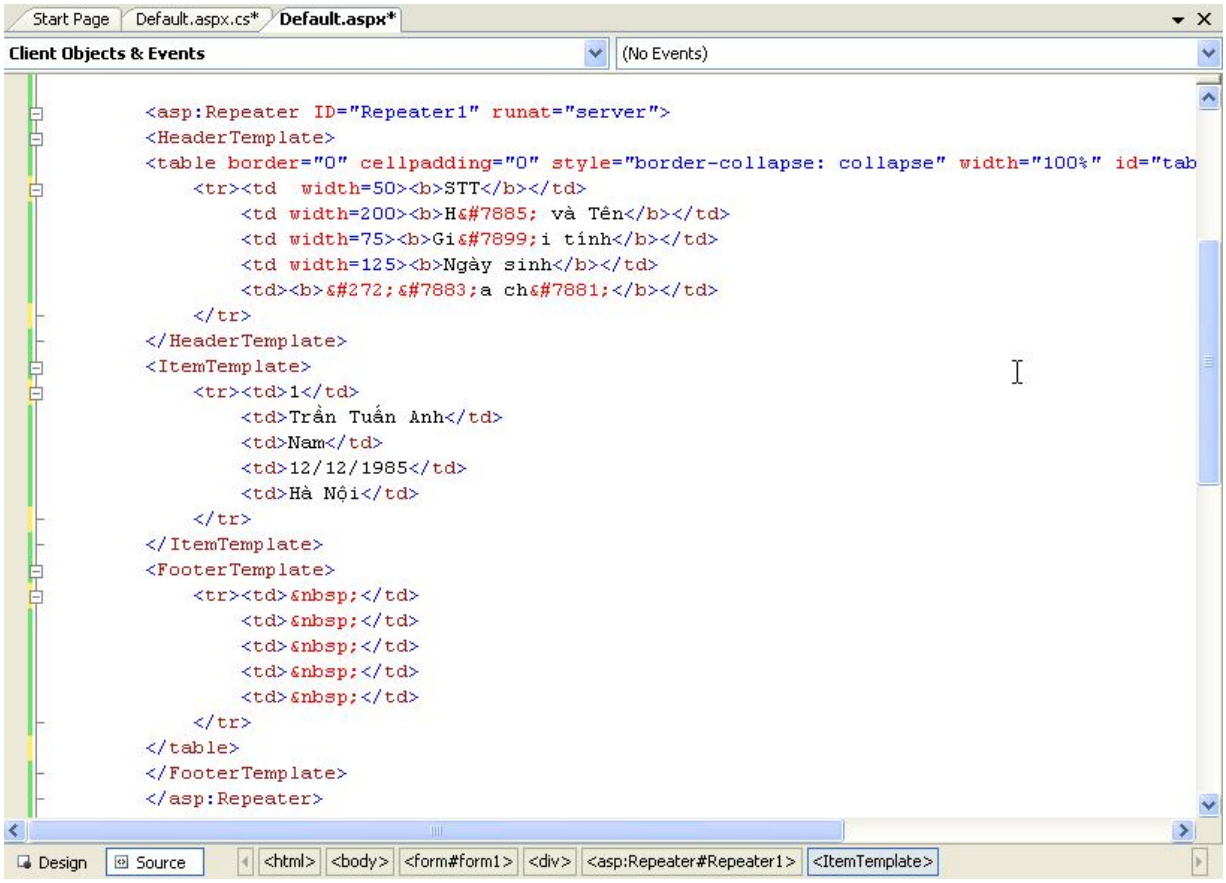
Phần thân (Item) là:

```
      <tr>
            <td>1</td>
            <td>Trần Tuấn Anh</td>
            <td>Nam</td>
            <td>12/12/1985</td>
            <td>Hà Nội</td>
      </tr>
```

Và phần cuối (Footer)

```
      <tr>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
      </tr>
</table>
```

Bây giờ, bạn chỉ cần copy mỗi phần ở trên vào trong mỗi phần tương ứng của Repeater **rpstudents** ở trên,



Ba thành phần chính của Repeater Template đã được lên cấu trúc

Tới đây bạn đã sắp hoàn tất quá trình tạo template cho repeater. Như đã đề cập, Repeater thực hiện việc Binding từ Repeater đến dữ liệu trong cơ sở dữ liệu, bạn sẽ thực hiện việc này tại ItemTemplate.

ItemTemplate là nơi hiển thị danh sách học sinh, chúng ta chỉ cần tạo một Template để hiển thị cho tất cả học sinh trong danh sách. Ví dụ danh sách học sinh gồm 20 bạn, bạn chỉ cần tạo template cho một bạn đầu tiên, tất cả thông tin của 19 bạn còn lại sẽ được hiển thị theo template đã thực hiện cho bạn đầu tiên ấy.

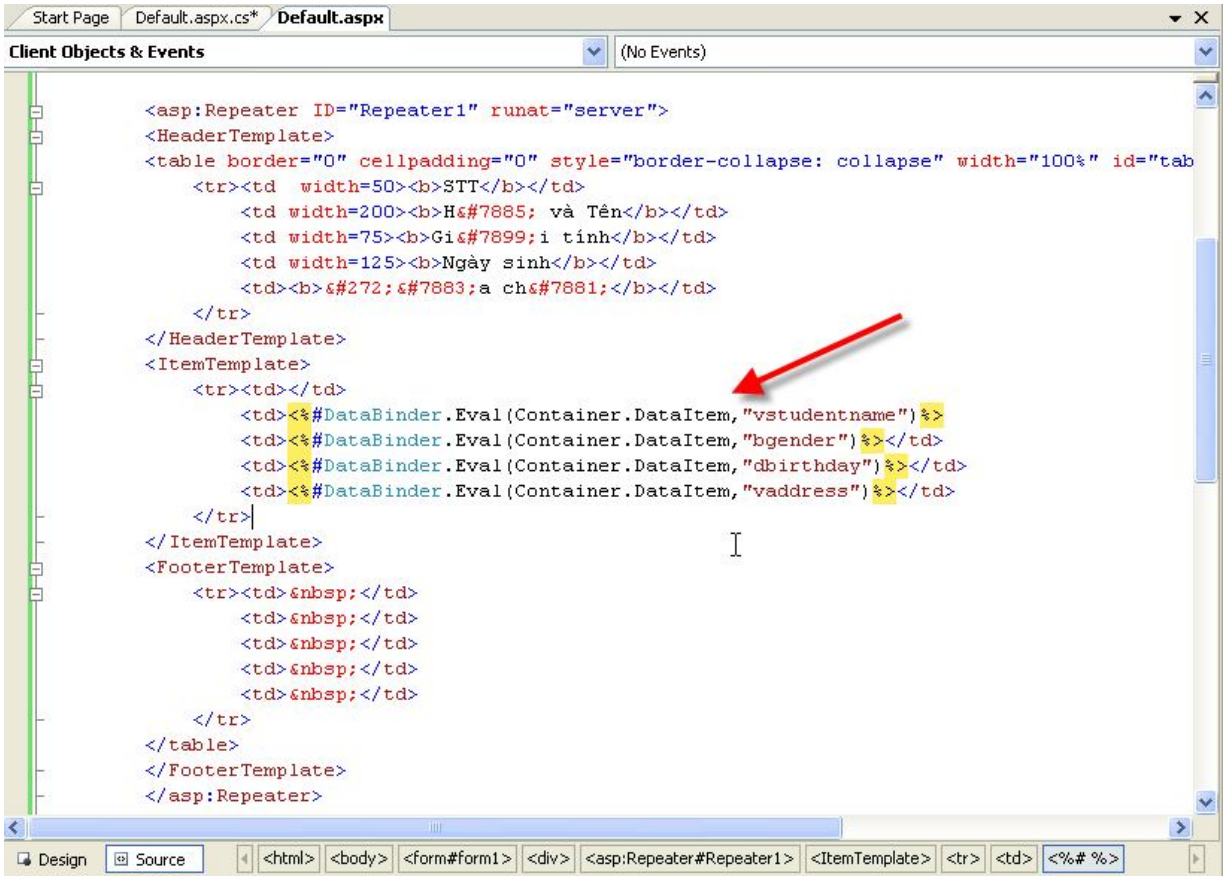
Để binding dữ liệu đến datasource, ASP.Net cung cấp phương thức DataBinder.Eval() để thực hiện việc gắn dữ liệu từ DataSource vào ItemTemplate của Repeater. Công thức của DataBinder.Eval() như sau:

<% # DataBinder.Eval(ContainerName, DataFieldName, {FormatString}) %>.

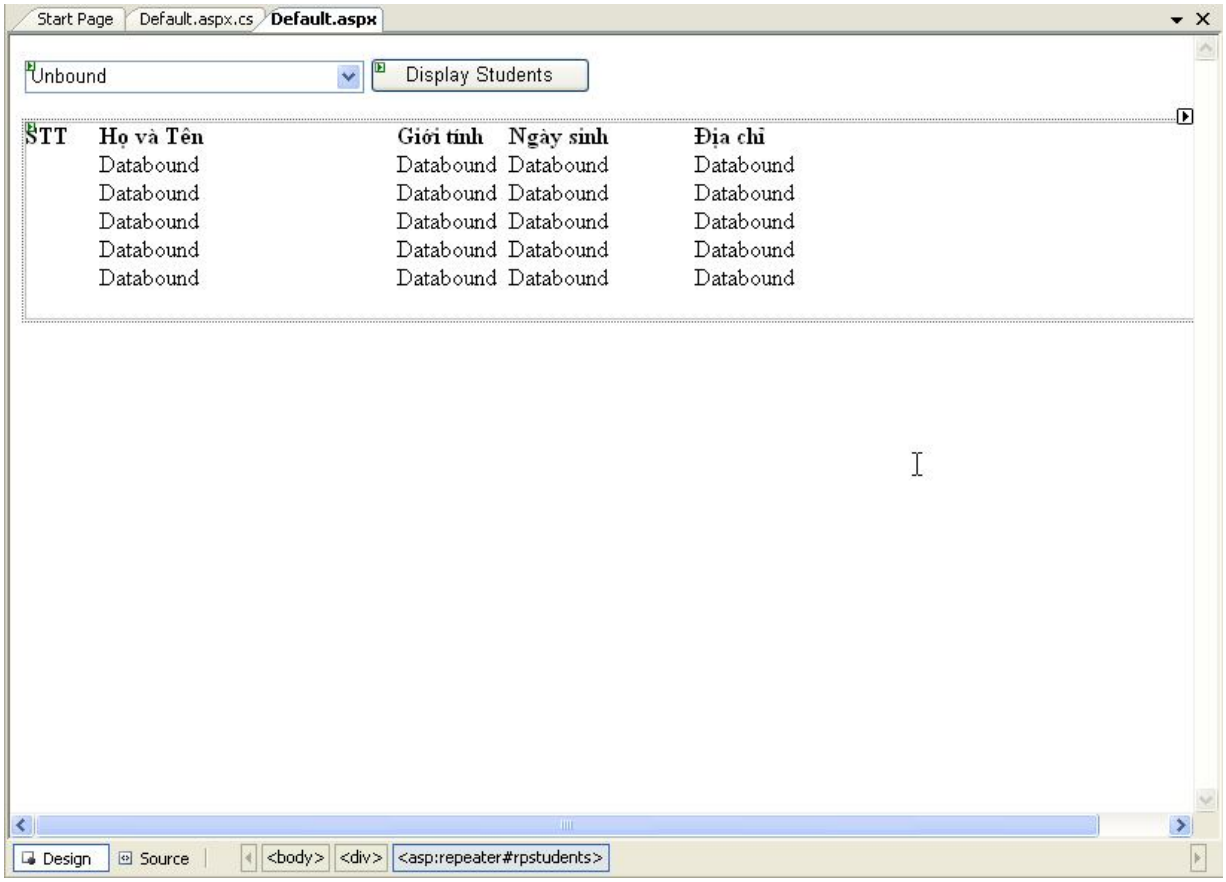
Áp dụng công thức trên cho trường tên, bạn sẽ được như sau:

```
<%#DataBinder.Eval(Container.DataItem,"vstudentname")%>
```

Áp dụng tương tự cho các trường khác, chúng ta sẽ được kết quả là cấu trúc của ItemTemplate được hiển thị như mô tả sau:



Như thế bạn đã hoàn tất việc tạo Template cho Repeater, và kết quả như mô tả sau đây



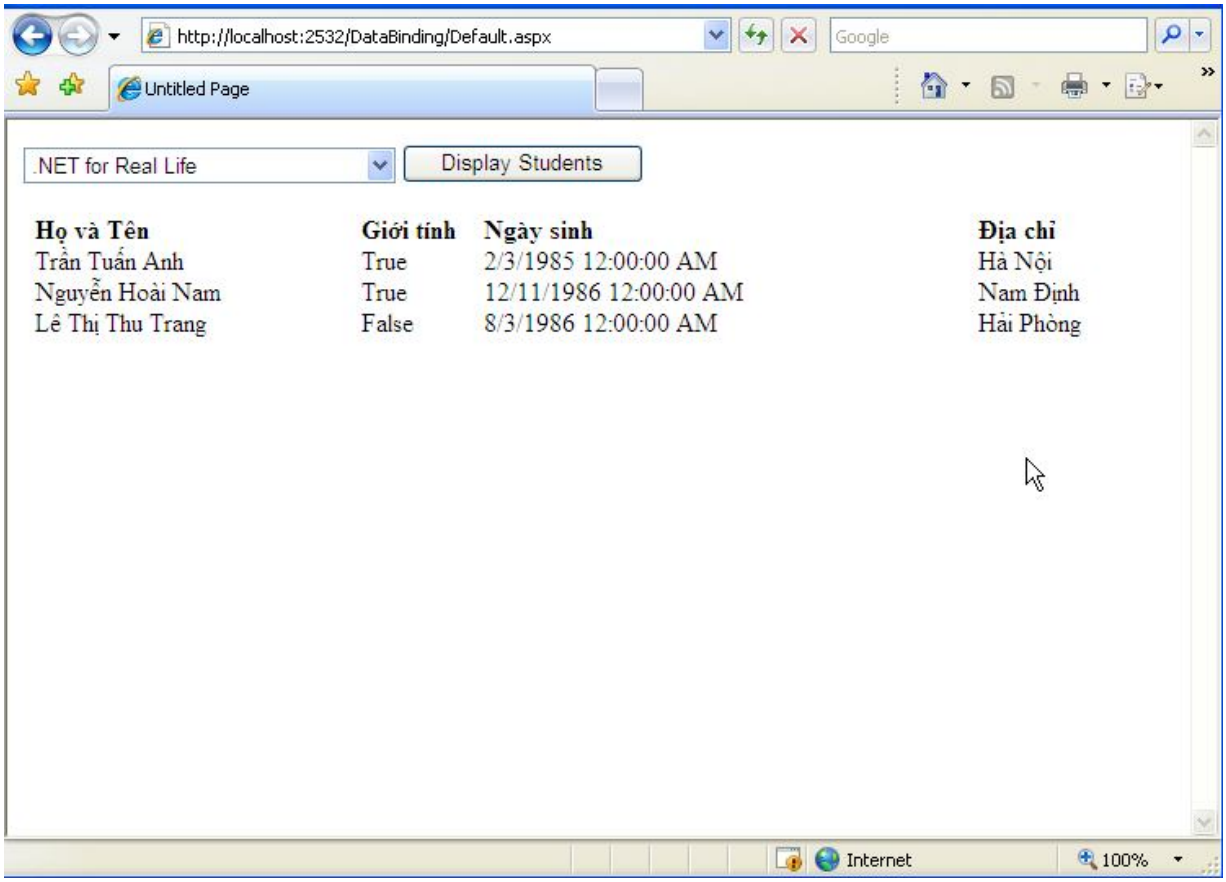
Kết quả sau khi xây dựng hoàn tất template cho repeater

Bây giờ bạn cần xây dựng DataSource và thực hiện lệnh DataBind() cho repeater rpstudents.

Khi người quản lý muốn xem danh sách sinh viên, anh ấy cần nhấn vào nút "Display Students" để hiển thị danh sách sinh viên tại repeater. Do đó việc Bind dữ liệu từ Repeater đến DataSource cần thực hiện tại đây. Hàm btndisplay_Click() như sau:

```
protected void btndisplay_Click(object sender, EventArgs e)
    {
        OleDbConnection conobj = new OleDbConnection(str_connection);
        conobj.Open();
        OleDbCommand cmd = new OleDbCommand("select * from students where
iclassid=?", conobj);
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.Add("@classid", this.ddlclasses.SelectedValue);
        OleDbDataAdapter da = new OleDbDataAdapter(cmd);
        DataSet ds = new DataSet();
        da.Fill(ds, "students");
        // xác định DataSource để bind dữ liệu
        this.rpstudents.DataSource = ds.Tables["students"];
        // Gọi hàm DataBind() thực hiện bind dữ liệu giữa CSDL và Repeater
        this.rpstudents.DataBind();
        conobj.Close();
    }
```

Nhấn F5 để chạy chương trình, kết quả như sau:

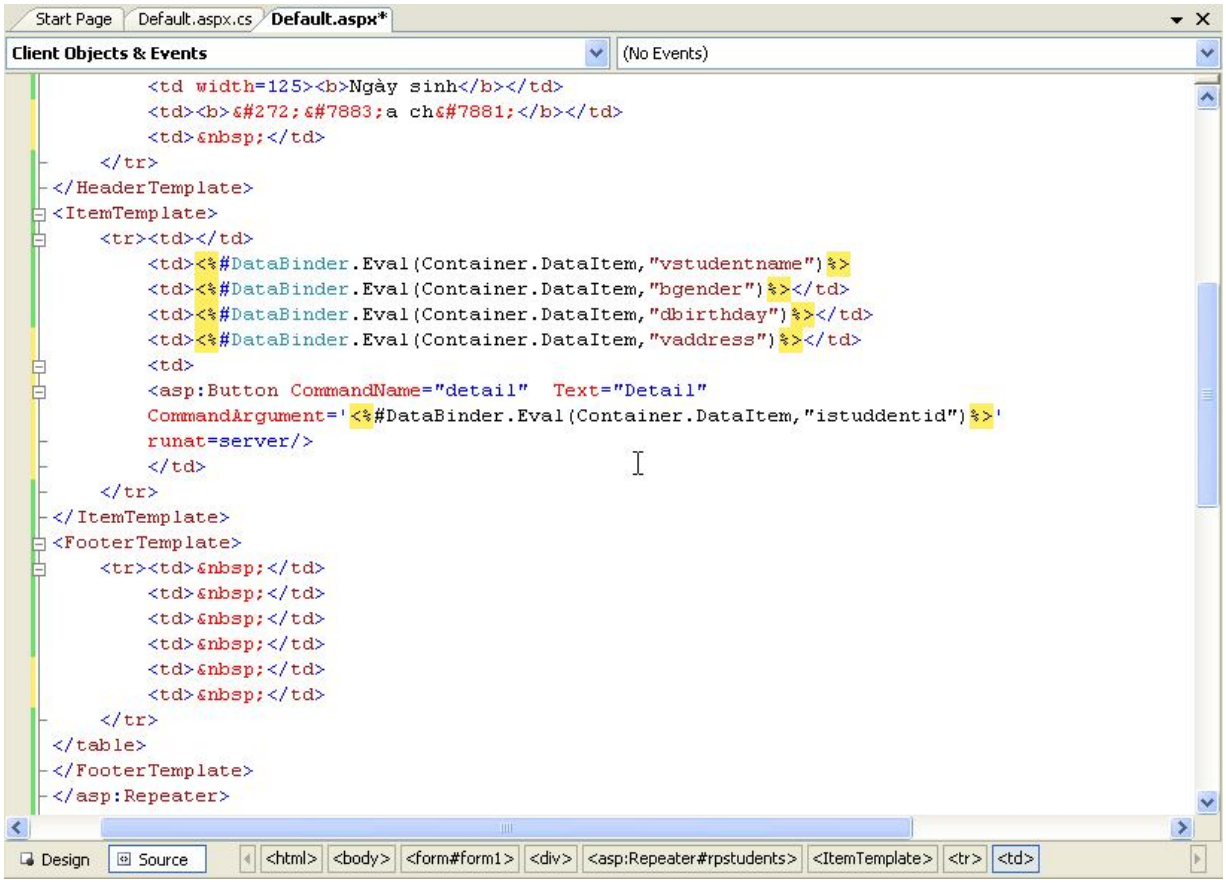


Như thế bạn đã hoàn tất việc Binding từ Repeater đến DataSource chứa dữ liệu được lấy từ CSDL.

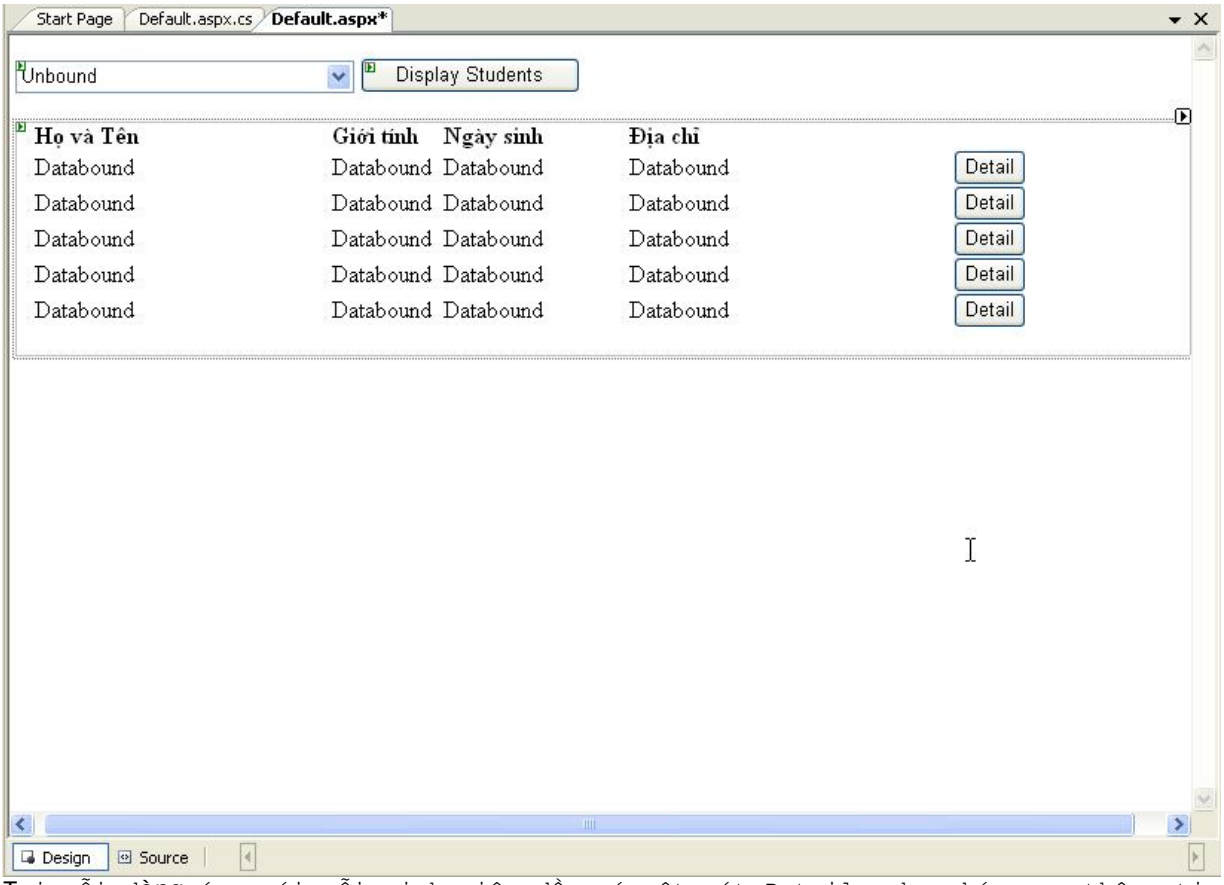
Gắn điều khiển vào Repeater

Sau khi bạn đã hiển thị được danh sách sinh viên trong từng lớp, có một nhu cầu đó là xem chi tiết điểm số của từng sinh viên, như thế tại mỗi dòng ứng với mỗi sinh viên, bạn phải thêm một button cho phép xem chi tiết điểm số từng học viên. Bạn thực hiện điều này theo những bước sau:

Tại phần ItemTemplate, bạn thêm vào Button Control, như sau:



Ở cửa sổ thiết kế, bạn có được kết quả



Tại mỗi dòng ứng với mỗi sinh viên đều có một nút Detail, cho phép xem thông tin chi tiết của mỗi sinh viên đó.

Để kiểm soát hàng nào tương ứng với một sinh viên được chọn, chúng ta truyền vào tham số CommandArgument='<%#DataBinder.Eval(Container.DataItem,"istuddentid")%>' với mã sinh viên ứng với mỗi sinh viên trên một dòng. Và sự kiện ItemCommand của Repeater cho phép hứng lấy giá trị truyền vào này. Sự kiện ItemCommand của Repeater như sau:

```
protected void rpstudents_ItemCommand(object source, RepeaterCommandEventArgs
e)
{

}
```

Tham số RepeaterCommandEventArgs e chứa đựng các giá trị của CommandName và CommandArgument của Button được gắn tại ItemTemplate.

Để kiểm tra giá trị truyền vào như thế nào, bạn hãy thêm dòng mã lệnh sau vào thân của sự kiện ItemCommand(),
Response.Write(e.CommandName + " " + e.CommandArgument.ToString());
Khi ấy, khi bạn chọn nhấn một nút Detail tương ứng với một sinh viên trên một dòng, thức thì xuất hiện mã số tương ứng với sinh viên ấy được lấy từ cơ sở dữ liệu.

Với các điều khiển DataList và DataGrid, việc Binding data là tương tự, chỉ khác nhau ở việc tạo và sử dụng những điều khiển được gắn vào Template.

Xem thêm cách tạo Template tại hướng dẫn: PTA_NET_2_0_ASP_NET_Create_Template.WMV

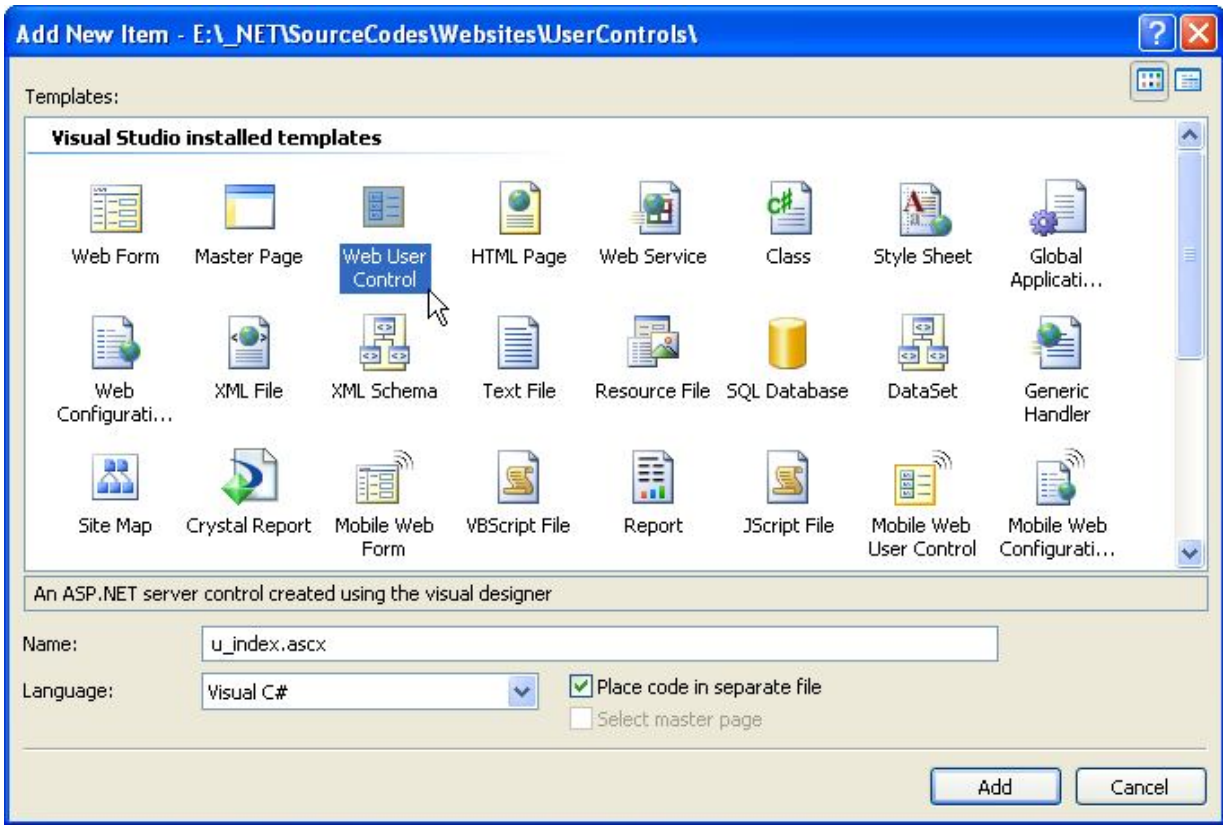
UserControl và ứng dụng trong xây dựng WebPortal

UserControl có thể được hiểu đó là những điều khiển do người sử dụng tạo nên, bạn có thể tạo ra những điều khiển cho riêng bạn trong nhiều trường hợp và sử dụng UserControl vào ứng dụng của bạn. Đặc biệt trong kiến trúc xây dựng WebPortal mà tôi giới thiệu trong phần này mô tả rõ nhất về việc sử dụng UserControl trong xây dựng WebPortal với ASP.NET.

Tạo và Sử dụng UserControl

Để tạo UserControl bạn thực hiện các bước sau:

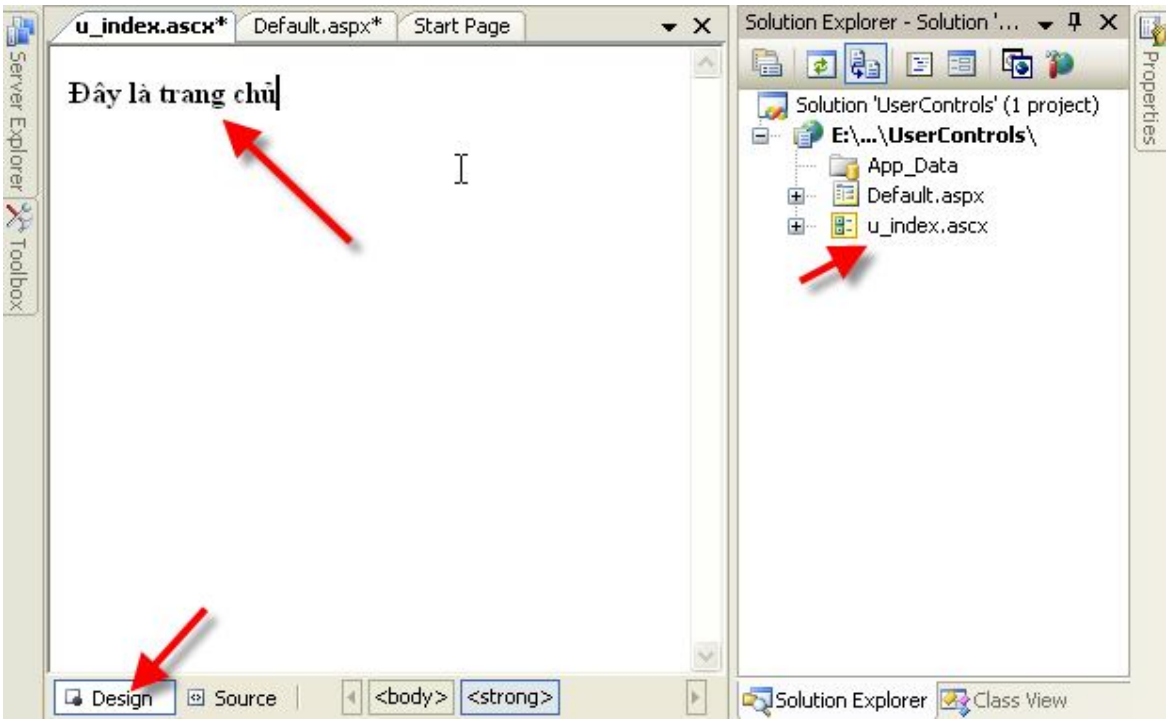
- Từ menu Visual Studio 2005, bạn chọn **Website → Add New Item**, Cửa sổ **Add New Item** xuất hiện như mô tả sau đây



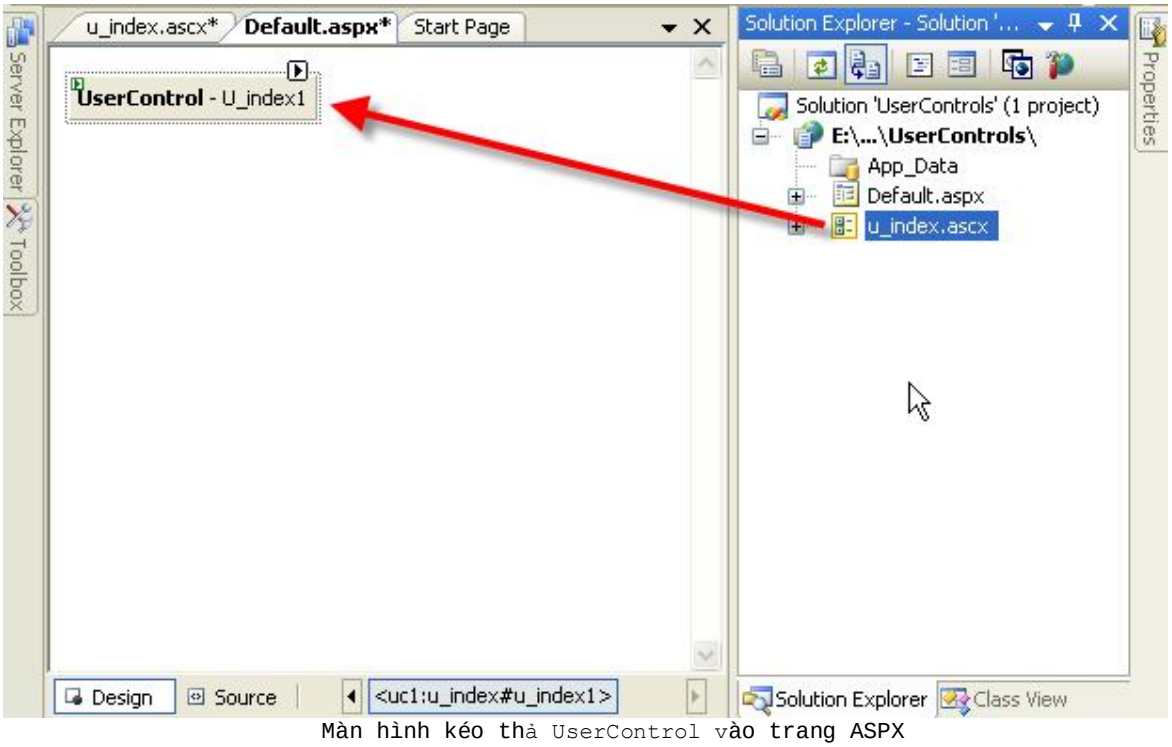
- Chọn **Web User Control**.
- Tại trường Name: bạn gõ tên của control là **u_index.ascx**
- Tại mục Language: bạn chọn ngôn ngữ **C#**
- Nhấn **Add** để thực hiện thêm vào.

Một file có tên u_index.ascx được tạo ra, và bạn có thể tìm thấy file này tại Cửa sổ Solution Explorer.

Bạn mở file này bằng cách **double click** vào file tại cửa sổ **Solution Explorer**, chuyển sang chế độ thiết kế **Design**, bạn gõ vào dòng chữ: **"Đây là trang chủ"**.

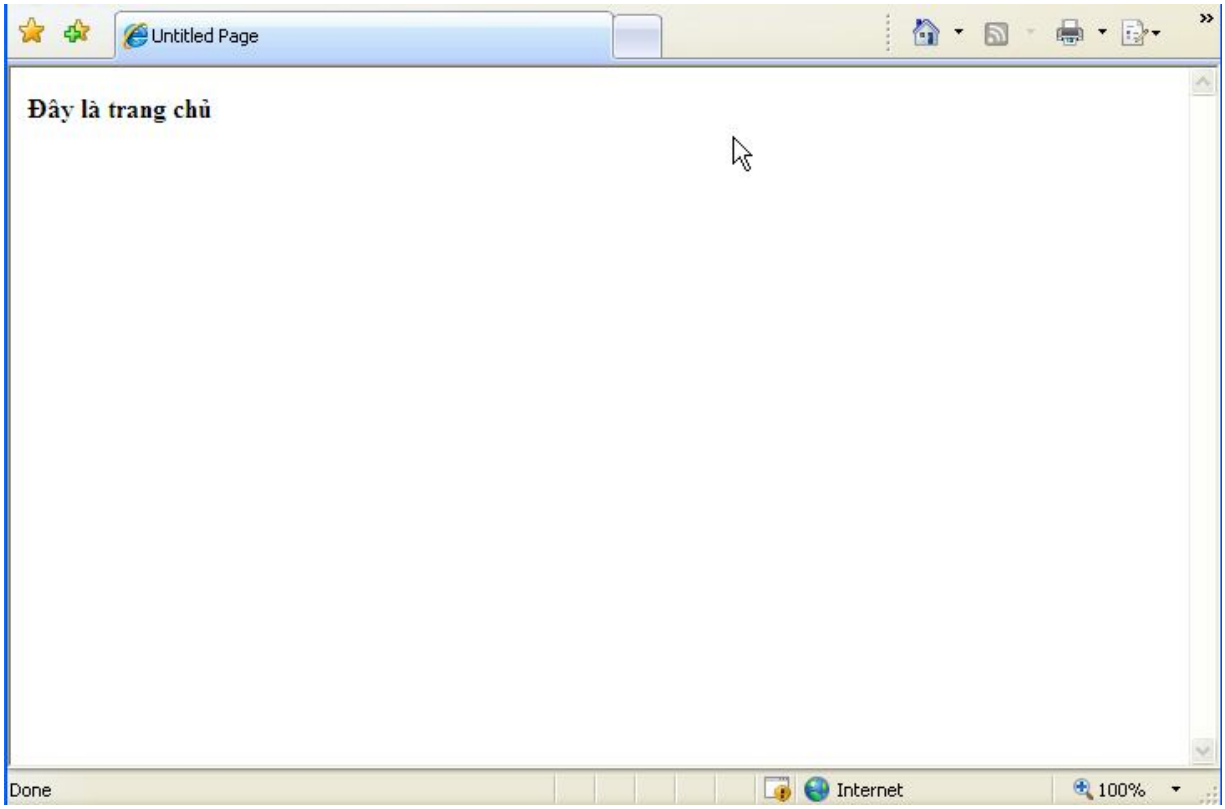


Sau đó bạn mở trang Default.aspx, và kéo thả file u_index.ascx vào màn hình thiết kế trang Default.aspx, và bạn nhấn F5 để chạy chương trình, bạn sẽ thấy rằng nội dung tại u_index.ascx xuất hiện tại trang Default.aspx



Màn hình kéo thả UserControl vào trang ASPX

Kết quả trực hiện chương trình như sau:



Bây giờ bạn có thể mở file soạn thảo mã lệnh điều khiển UserControl u_index.ascx và thực hiện dòng lệnh sau:

```
protected void Page_Load(object sender, EventArgs e)
    {
        Response.Write("Từ UserControl");
    }
```

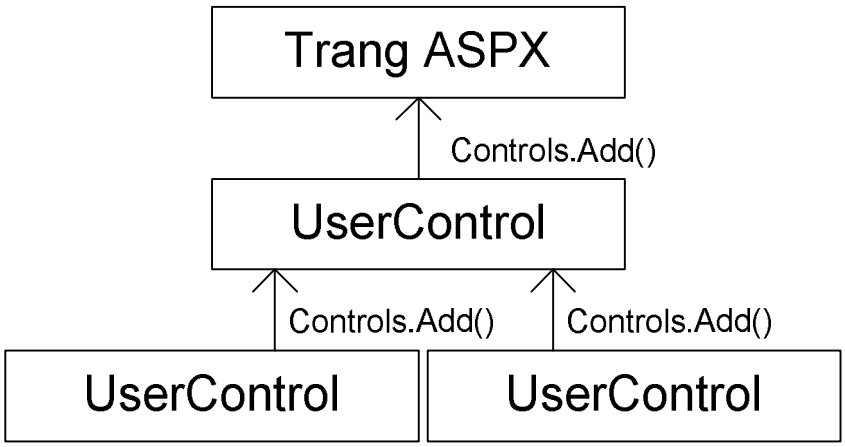
Bây giờ thực hiện chương trình, và kết quả sẽ là dòng "Từ UserControl" cũng xuất hiện tại trang ASPX.

Như thế UserControl khi được gắn vào một trang ASPX, tất cả những thực thi trong UserControl đều được thực thi tại trang ASPX, vì bây giờ UserControl trở thành thành phần của trang ASPX.

Có rất nhiều kỹ thuật xây dựng WebPortal, kỹ thuật xây dựng Webportal sử dụng UserControl là một kỹ thuật tôi tự xây dựng và kiến trúc dựa trên kiến trúc tải (load) động của UserControl.

Kiến trúc tải UserControl động

Kiến trúc tải UserControl động được mô hình hóa như sau



Mô hình kiến trúc tải UserControl động.

Việc tải UserControl là việc thực hiện bằng mã lệnh chương trình lại hành động kéo thả UserControl vào trang ASPX, ASCX. Ở mô hình trên, một trang ASPX có thể load nhiều UserControl, và mỗi UserControl có thể tải các UserControl khác và cứ tiếp tụ như thế. Với mô hình này cho phép bạn tạo ra những ứng dụng có độ phức tạp rất cao nhưng đem lại sự dễ dàng cho người sử dụng ứng dụng web của bạn.

Công thức của hàm LoadControl như sau:

```
Controls.Add(LoadControl("[ControlPath][ControlName]"));
```

Trong đó,

- *ControlPath*: là đường dẫn đến UserControl.
- *ControlName*: là tên UserControl

Ví dụ:

```
Controls.Add(LoadControl("u_module1.ascx"));
```

Hiện thực kiến trúc WebPortal

Sau đây chúng ta thực hiện kiến trúc WebPortal dựa trên kỹ thuật tải UserControl động.

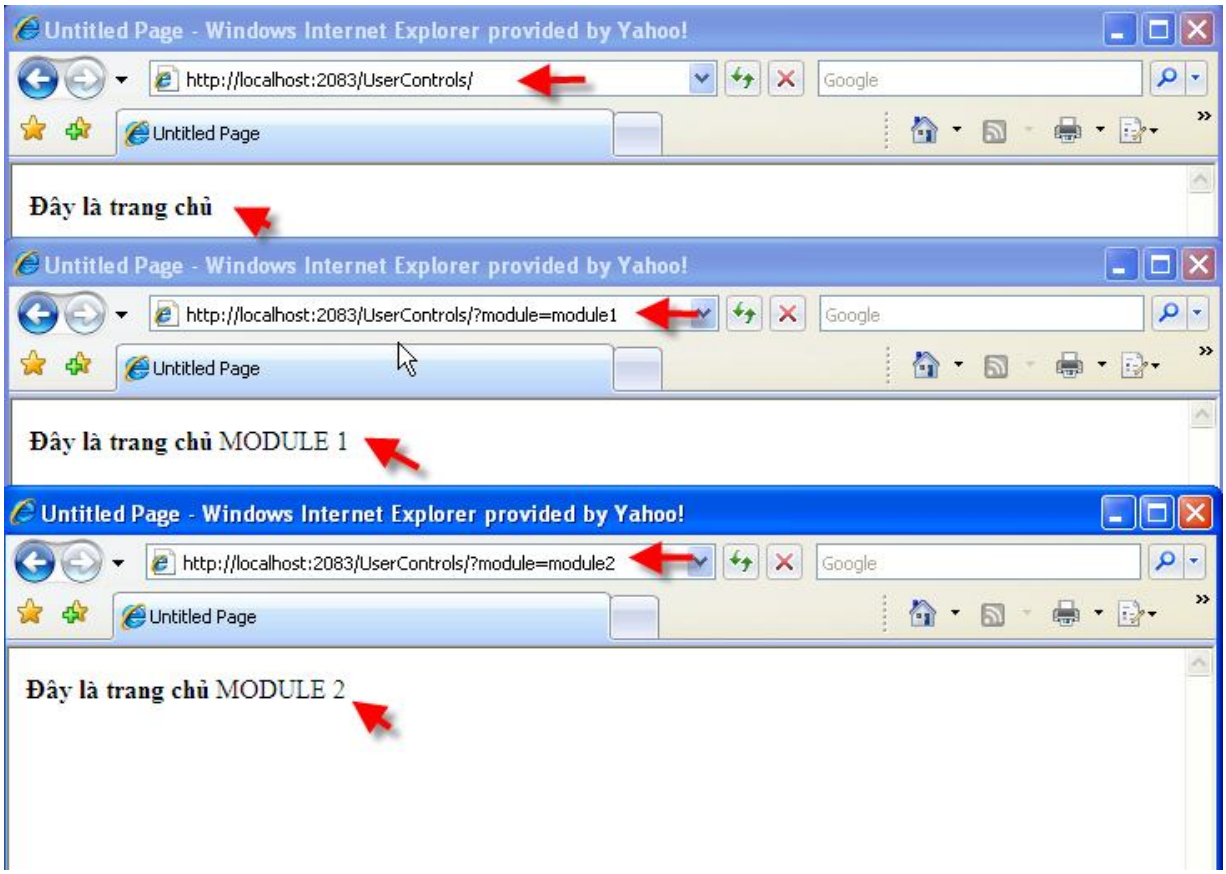
Tiếp tục với ví dụ ở trên về UserControl, bây giờ bạn thêm vào hai UserControl khác tên u_module1.ascx và u_module2.ascx, tại mỗi UserControl vừa thêm, bạn thêm tương ứng hai dòng giá trị sau: "UserControll: Module 1", "UserControl: Module2"

Tại file soạn thảo mã lệnh điều khiển chương trình cho UserControl u_index.ascx.cs, bạn thực hiện như sau

```
string module = "";
        if(Request.QueryString["module"] != null)
            module = Request.QueryString["module"].ToString().Trim();
```

```
        switch (module)
        {
            case "module1":
                Controls.Add(LoadControl("u_module1.ascx"));
                break;
            case "module2":
                Controls.Add(LoadControl("u_module2.ascx"));
                break;
        }
```

Nhấn F5, chạy chương trình, kết quả như sau:



Khi không có tham số module, chỉ có giá trị tại u_index.ascx được tải lên trang ASPX, khi cung cấp tham số module, và gán lần lượt bằng hai giá trị module1 và module2, bạn thu được kết quả như hình mô tả ở trên.

Ví dụ trên là mô tả cơ bản nhất cho việc sử dụng UserControl trong xây dựng WebPortal, kiến trúc này cho phép bạn tạo ra những ứng dụng cực kỳ phức tạp về sau.

Kiến trúc tải UserControl động sử dụng PlaceHolder

Trong phần này chúng ta cùng thực hiện hiện thực hóa mô hình WebPortal sử dụng PlaceHolder.

Cũng như ví trụ trên, nhưng chúng ta sẽ thay thế UserControl liên kết (UserControl được sử dụng để load các UserControl khác) bằng một Place Holder bằng cách kéo thả một PlaceHolder vào trong trang ASPX từ cửa sổ ToolBox.

Sau đó, thay thế đoạn mã lệnh điều khiển bằng đoạn mã lệnh như sau:

```
string module = "";
        if(Request.QueryString["module"] != null)
            module = Request.QueryString["module"].ToString().Trim();
        switch (module)
        {
            case "module1":
                phcontrol.Controls.Add(LoadControl("u_module1.ascx"));
                break;
            case "module2":
                phcontrol.Controls.Add(LoadControl("u_module2.ascx"));
                break;
        }
```

Khi đó bạn sẽ có một kết quả tương tự như với cách sử dụng một UserControl để load các Control khác.

Bài tập tự ôn luyện.

1. Tạo ứng dụng quản lý danh sách lớp, danh sách sinh viên trong từng lớp. Ứng dụng có các tính năng: thêm, sửa, xóa lớp và sinh viên trong từng lớp.
2. Xây dựng ứng dụng quản lý kho hàng, cho phép người quản lý quản lý danh mục hàng hóa, số lượng hàng hóa trong kho, quản lý nhập xuất hàng.
3. Xây dựng hệ thống website (Webportal) với yêu cầu sau:
   - Module tin tức: cung cấp thông tin thời sự
   - Module âm nhạc: cho phép người sử dụng lựa chọn và nghe nhạc trực tuyến
   - Module thư viện hình: cho phép ngươi sử dụng xem hình ảnh được sắp xếp theo phân loại bởi người quản trị website.

Cấu hình cho ứng dụng Web ASP.NET

ASP.NET cung cấp file để cấu hình cho ứng dụng Website, đó là web.config. Chúng ta sẽ tìm hiểu cách thức để sử dụng file web.config cho công việc cấu hình cho ứng dụng Web.

Mục <appSettings>

appSettings trong file web.config được sử dụng để định nghĩa những cài đặt do người dùng xây dựng cho ứng dụng website. Chẳng hạn bạn có thể sử dụng thẻ appSettings để lưu trữ thông tin về chuỗi kết nối đến cơ sở dữ liệu.

Công thức của thẻ <appSettings> như sau:

```
<appSettings>
<add key=”(key)”  value=”(value”/>
</appSettings>
```

Trong đó,

- Key: xác định giá trị khóa trong hastable trong appSettings.
- Value: là nội dung tương ứng với khoa key.

Ví dụ về việc sử dụng appSettings để lưu trữ thông tin chuỗi kết nối

```
<appSettings>
<add                                                    key=”ConnectingString”
value=”Provider=SQLOLEDB;database=(local);uid=sa;pwd=” />
</appSettings>
```

Đọc giá trị từ thẻ appSettings

Để đọc các giá trị từ thẻ appSettings, bạn thực hiện như sau

Import thư viện System.Configuration; bằng cách

```
using System.Configuration;
```

đọc giá trị từ một khóa (key).

```
string connectionstring =
System.Configuration.ConfigurationSettings.AppSettings["ConnectingString"];
```

Ở ví dụ trên, ConnectingString là khóa được đặt trong thẻ appSettings tại file web.config

Thẻ <customErrors>

Sử dụng thẻ <customErrors>, bạn có thể định nghĩ được những thông điệp lỗi xuất hiện cho một ứng dụng ASP.NET. Công thức của thẻ customErrors như sau:

```
<customErrosrs defaultRedirect=(URL) mode=”(on/off/remoteonly)”>
<error statusCode=”(statusCode)” redirect=”(URL)”/>
</customErrors>
```

Trong đó,

- defaultRedirect - xác định địa chỉ sẽ được chuyển đến khi có lỗi xuất hiện.

- mode - xác định mở, đóng, hoặc hiển thị trạng thái lỗi đến trình duyệt người sử dụng.
- <error> định nghĩa những lỗi xuất hiện.

Ví dụ:

```
<customErrosrs defaultRedirect="customerrors.aspx" mode="On">
<error statusCode="500" redirect="CustomErrorMessage.aspx"/>
</customErrors>
```

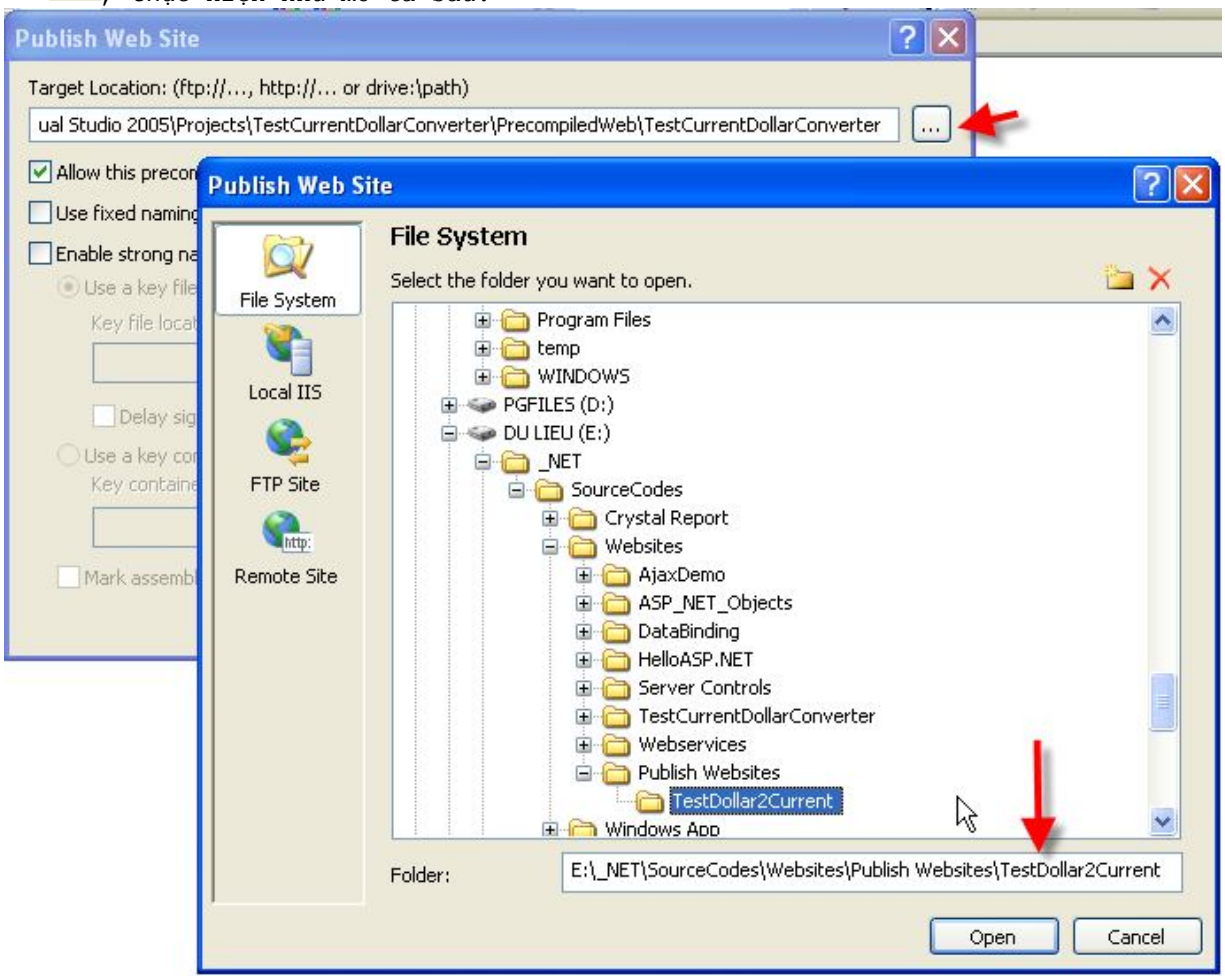
Khi lỗi xuất hiện, nếu mã lỗi là 500, trang sẽ được chuyển đến trang CustomErrorMessage.aspx, mặc định trang được chuyển đến customerrors.aspx.

Những hàm ở mức Ứng dụng (Application)

Xuất bản một ứng dụng Web ASP.NET

Với ASP.NET 2.0, sau khi xây dựng, kiểm tra ứng dụng web, bạn phải tiến hành xuất bản ứng dụng web để triển khai thực tế. Các bước triển khai như sau:

1. Mở ứng dụng Web đã xây dựng **TestCurrentDollar2Current**
2. Tại menu **Build**, chọn **Publish Website**.
3. Màn hình **Publish Website** xuất hiện, tại mục Target Location, nhấn vào nút ..., thực hiện như mô tả sau:



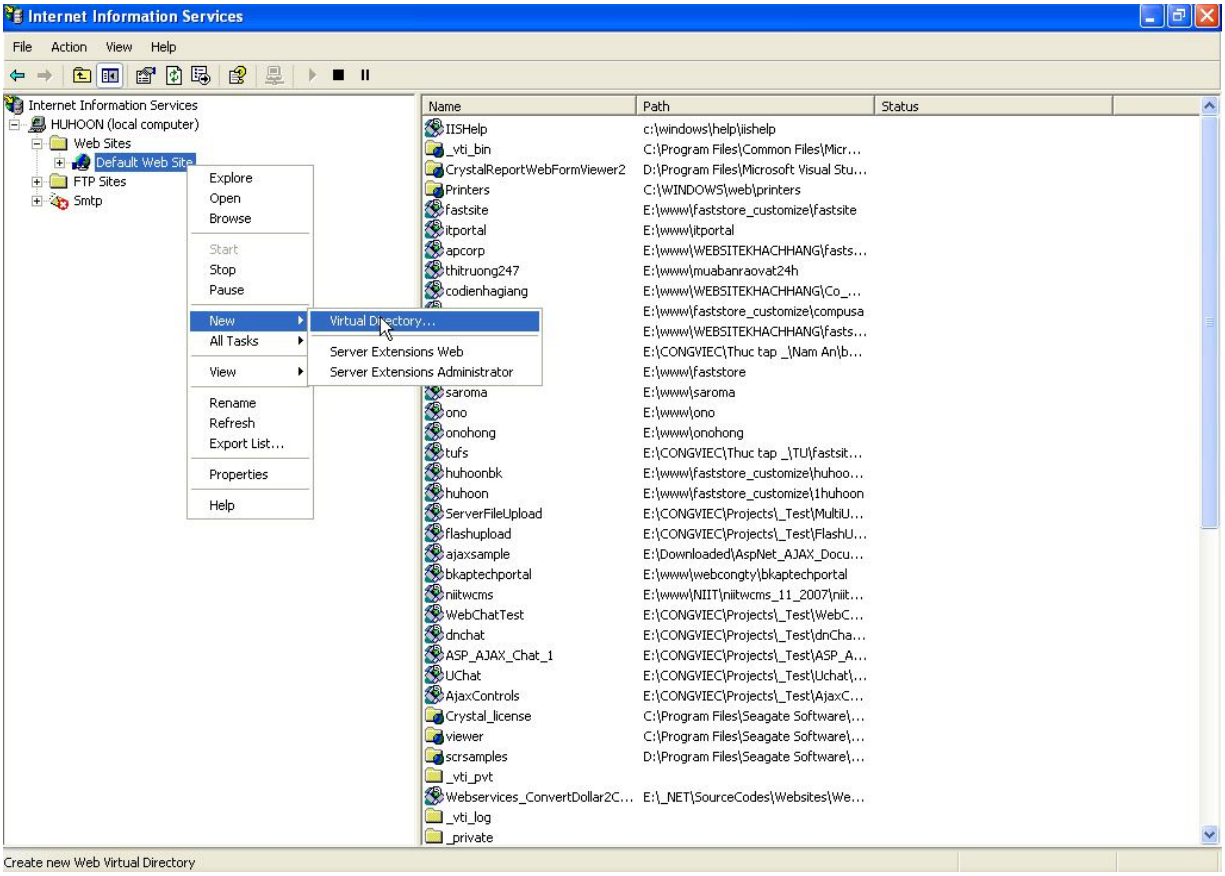
4. Nhấn **Open**, sau đó nhấn **OK** để thực hiện Publish website.

Bây giờ, website đã được xuất bản, và bạn có thể sử dụng để triển khai thực tế.

Triển khai một ứng dụng Website ASP.NET trên IIS

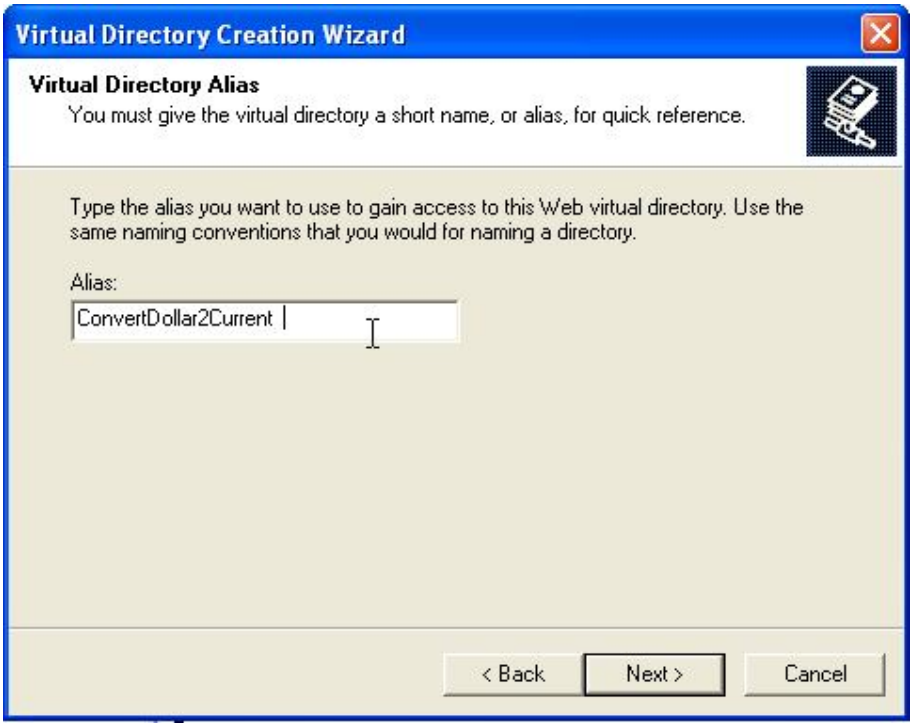
Sau khi website đã được xuất bản, bạn đã có thể triển khai Website phục vụ người dùng của bạn, các bước triển khai một website được thực hiện như sau:

1. Mở ứng dụng **Internet Information Services** tại **Control Panel → Administrative Tools**.
2. Màn hình ứng dụng Internet Information Services xuất hiện.
3. Tại mục **Websites → Default Web Site**, nhấn chuột phải, chọn **New → Virtual Directory…** như mô tả sau:

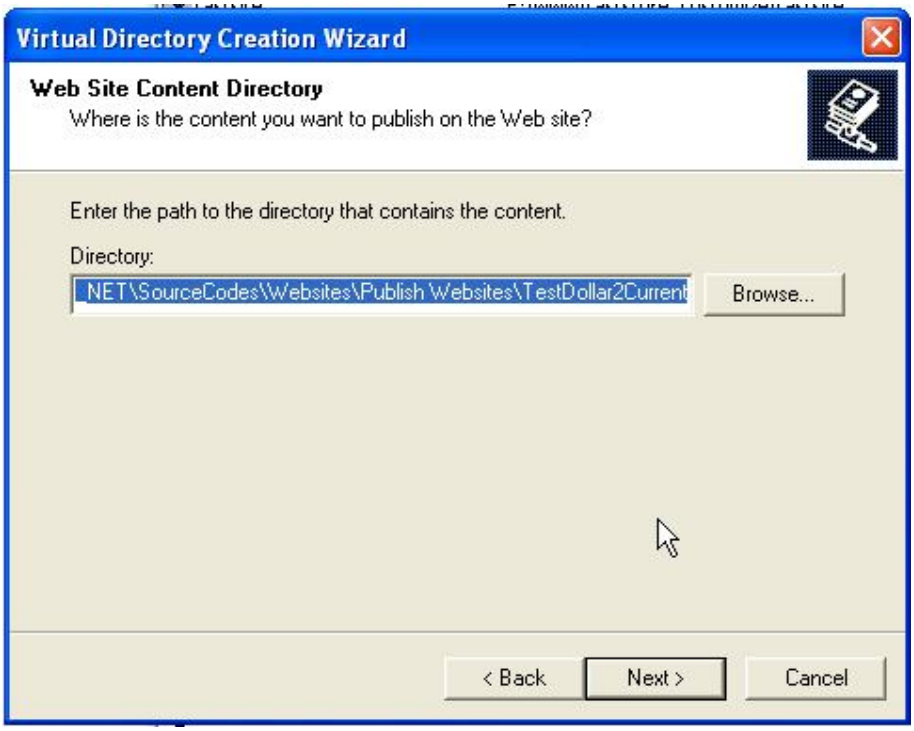


4. Màn hình tạo mới Virtual **Directory Creation Wizard** xuất hiện, chọn **Next**.

5. Cửa sổ **Virtual Directory Alias** xuất hiện, tại ô **Alias**, bạn nhập vào: **ConvertDollar2Current**

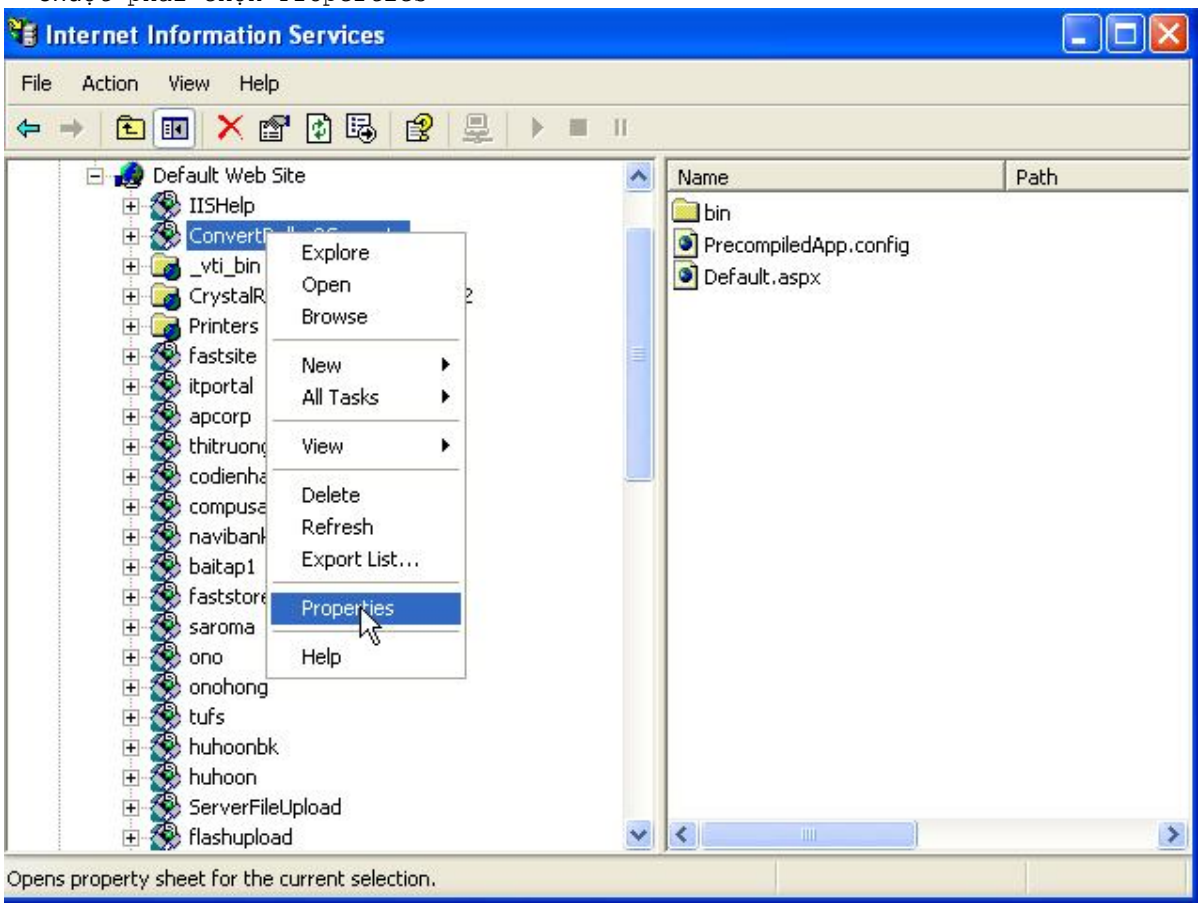


6. Cử sổ **Web Site Content Directory**, bạn đưa đến thư mục mà website của bạn đã được xuất bản, nhấn **Next**, như mô tả sau:

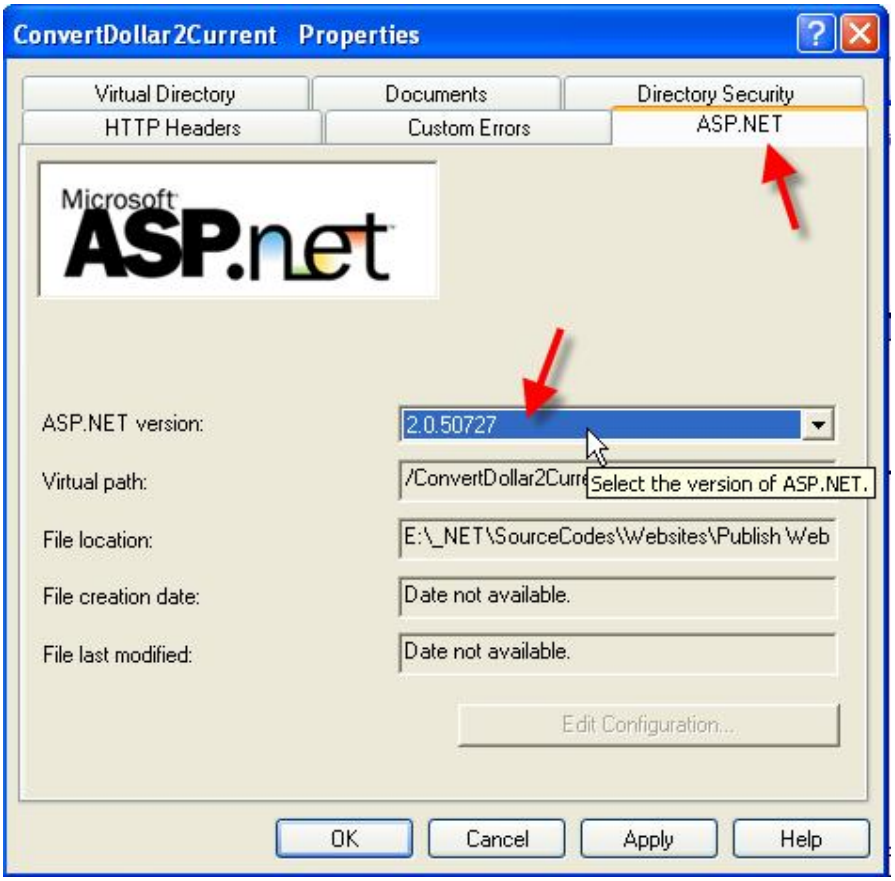


7. Tại cửa sổ **Access Permision**, nhấn **Next**.
8. Cửa sổ thông báo đã tạo thành công thư mục ảo cho ứng dụng web xuất hiện, nhấn **Finish**.

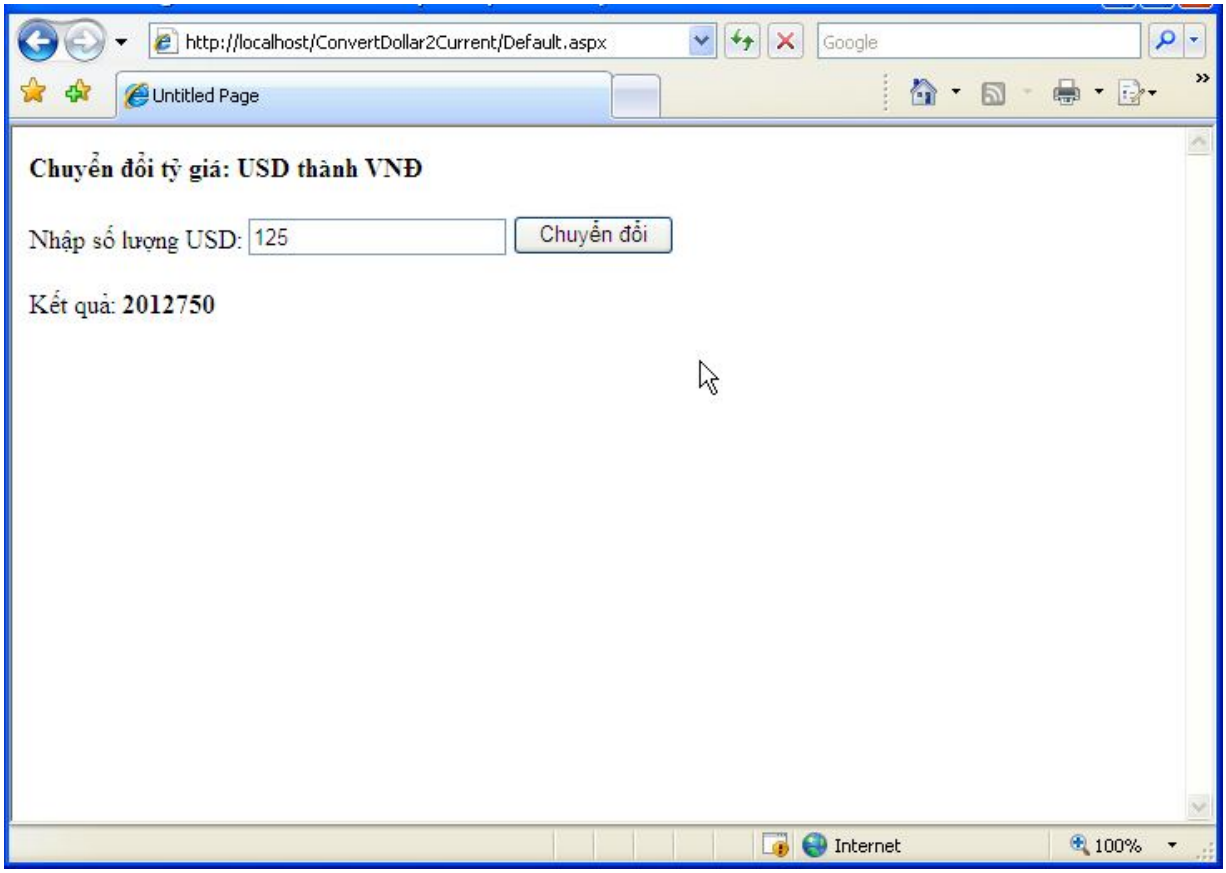
9. Bạn đã triển khai thành công ứng dụng web của bạn.
10. Vì ứng dụng của bạn được viết trên nền tảng .NET 2.0, do đó bạn cần cho IIS hiểu rằng ứng dụng của bạn được xây dựng để chạy trên nền tảng .NET 2.0.
11. Chọn mục **ConvertDollar2Current** tại cây thư mục **Default Web Site**. Nhấn chuột phải chọn Properties



12. Tại cửa sổ **ConvertDollar2Current Properties** xuất hiện, bạn chọn tab **ASP.NET**.
13. Tại lựa chọn **ASP.NET Version**, chọn **.NET 2.0**, như mô tả sau đây



14. Nhấn OK để kết thúc. Ứng dụng của bạn đã hoàn toàn được triển khai trên IIS.
15. Thực thi ứng dụng, bạn mở Browser và gõ như sau: http://localhost/ConvertDollar2Current/Default.aspx , trang web của bạn đã có thể được sử dụng.



Ứng dụng ASP.NET được triển khai thành công và thực thi

Web Services.

Trong những năm gần đây, sự hát triển của những ứng dụng Web đã dần thay thế cho những ứng dụng Desktop, ở một số doanh nghiệp đã chuyển toàn bộ hệ thống ứng dụng quản lý của mình từ hệ thống chạy trên Desktop sang nền tảng ứng dụng Web. Những hệ thống này có nhiều thành phần phức tạp được phát triển trên nhiều ngôn ngữ khác nhau và được triển khai tại nhiều địa điểm khác nhau. Do đó, việc phát triển những ứng dụng phân tán đặt ra một yêu cầu là đả bảo cho những thành phần này hoạt động xuyên suốt với nhau. Webservice được ra đời để đáp ứng yêu cầu này.

Một webservices cung cấp một số những phương thức cho phép sử dụng từ một hoặc nhiều ứng dụng khác nhau, bất chấp việc những ứng dụng đó được xây dựng trên ngôn ngữ nào. Những phương thức như thế được gọi là Web methods. Những hàm được sử dụng bởi một Web Service có thể được các ứng dụng khác truy cập đến bằng cách sử dụng chuẩn như Simple Oject Access Protocol (SOAP). SOAP là một giao thức sử dụng ngôn ngữ XML để mô tả dữ liệu và sử dụng HTTP để truyền dữ liệu. Một ứng dụng sử dụng một Web service được gọi là một Web service client.

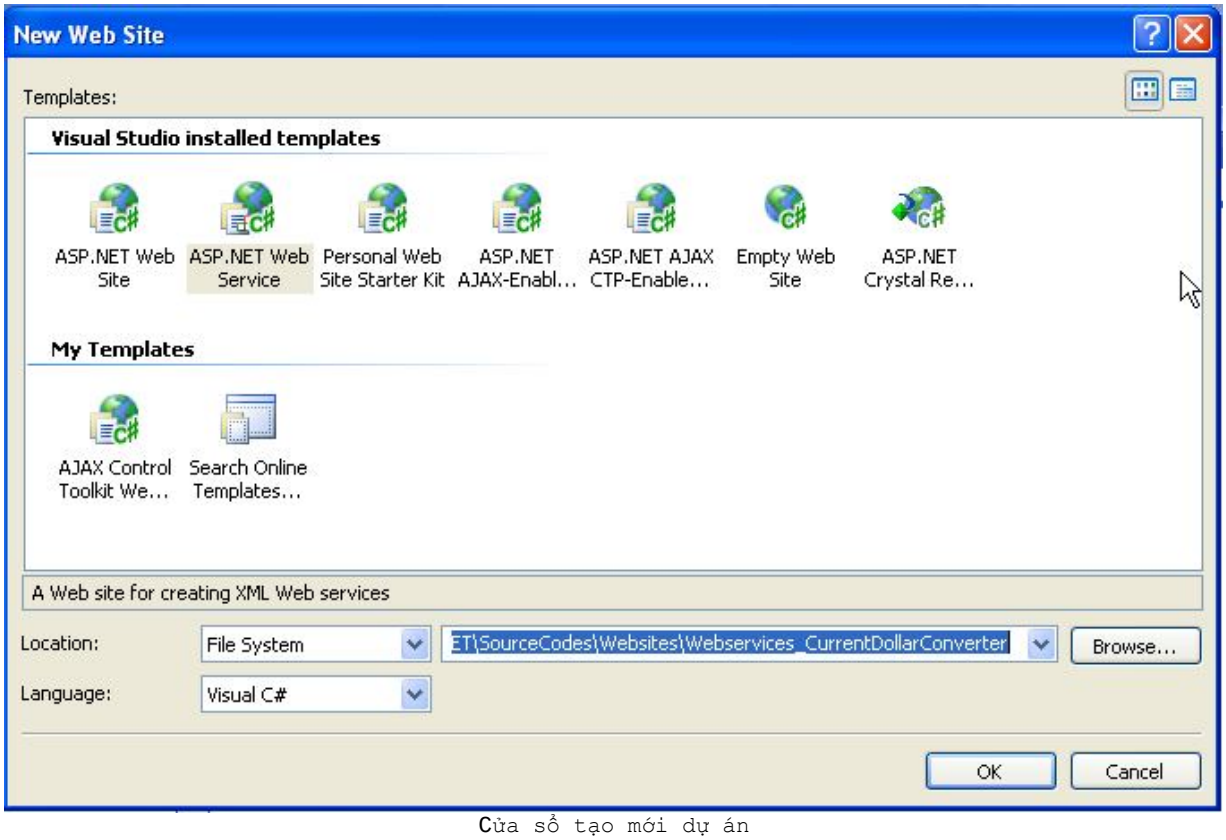
Web serivces cung cấp những lợi ích sau:

- Dễ dàng truy xuất: bất kỳ một máy tính nào truy cập vào Internet đều có thể dễ dàng truy xuất đến một Web Service.
- Cấu trúc linh hoạt: tùy theo yêu cầu của ứng dụng mà những loại khác nhau của web service có thể được tạo và sử dụng trong một ứng dụng.
- Dễ dàng tích hợp: sử dụng web service, bạn có thể dễ dàng tích hợp vào các ứng dụng được phát triển trên những nền tảng phần cứng và phần mềm khác nhau, giúp giảm thiểu chi phí. Hơn thế, bạn có thể tích hợp ứng dụng Web với ứng dụng chạy trên Desktop.

Khởi tạo và gọi một Web Services

Visual Studio .NET IDE cho phép dễ dàng khởi tạo một Web Service, để tạo một web services, bạn thực hiện theo các bước sau:

1. Mở Visual Studio .Net IDE
2. Chọn kiểu dự án là **Visual C# Project** từ cửa sổ **Project Types**, và chọn **Tempalte** là **ASP.NET Web Services**.
3. Trong ô textbox Location, gõ **http://localhost/Webservices_CurrentDollarConverter.asmx** và click vào nút **OK**.

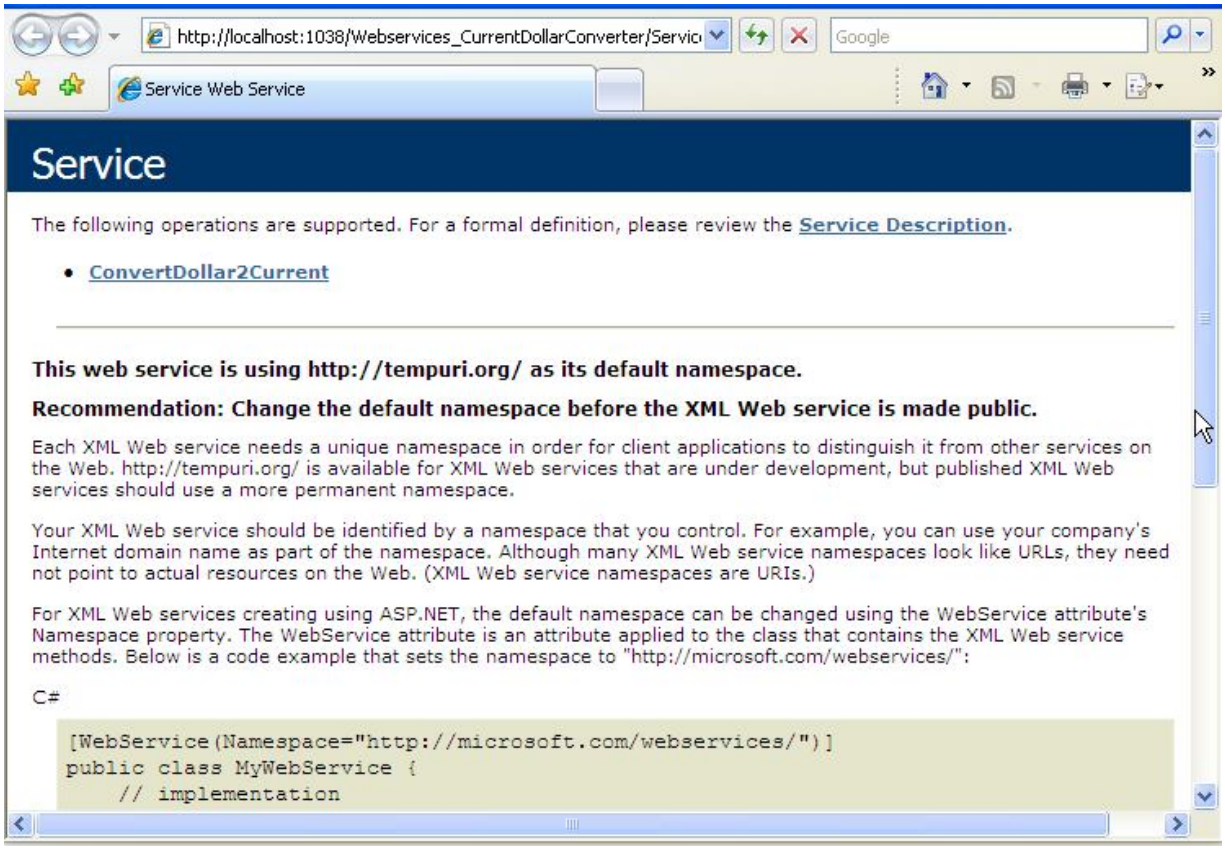


Cửa sổ tạo mới dự án

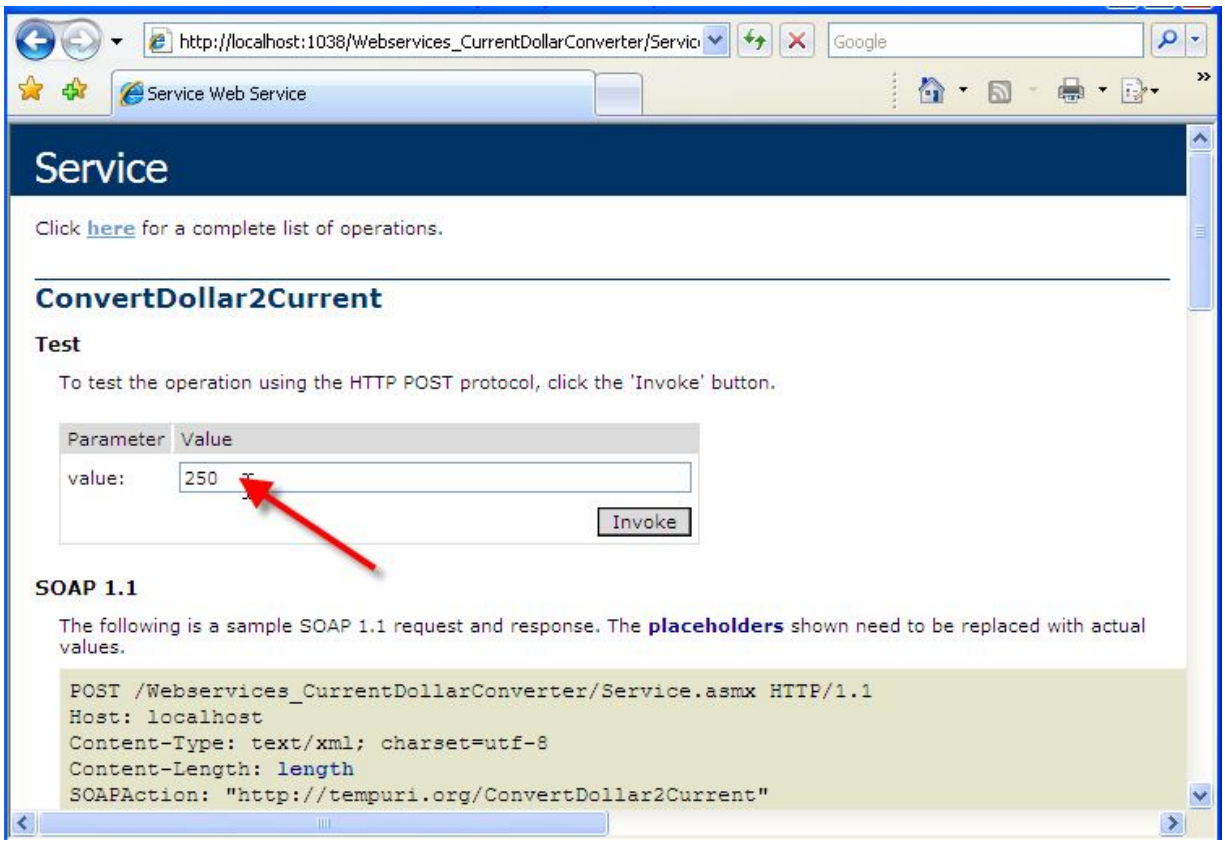
4. Click chuột phải file **Webservices_CurrentDollarConverter.asmx** tại cửa sổ Solution Explorer và chọn **View Code** để mở.
5. Thêm dòng mã lệnh sau vào cửa sổ soạn thảo mã lệnh điều khiển chương trình, trong lớp Service1

```
[WebMethod]
    public double ConvertDollar2Current(double value)
    {
        double rate = 16102;// tỷ giá tiền Việt và đồng $
        return value * rate;
    }
```

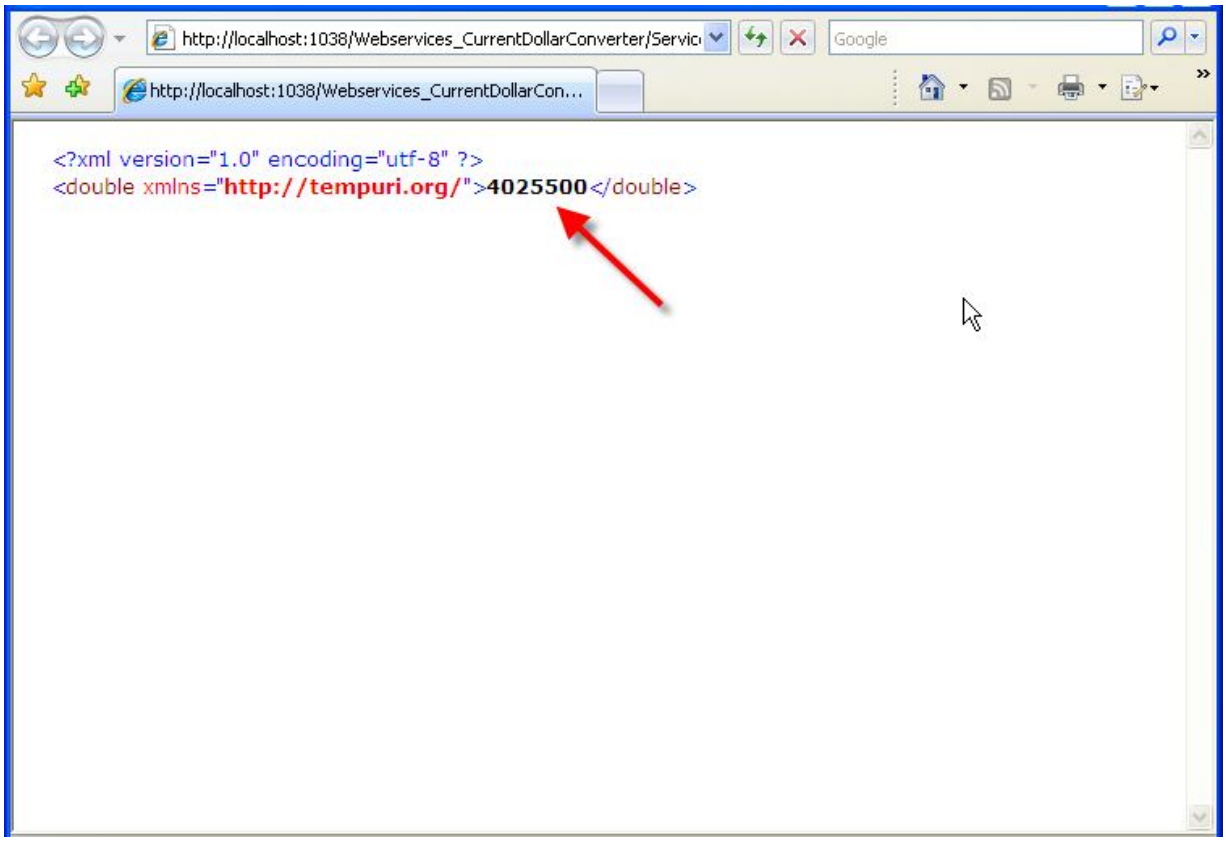
6. Nhấn **F5** để chạy chương trình.
7. Webservice xuất hiện, nhấn vào liên kết ConvertDollar2Current để thực hiện gọi webservice này.



8. Sau khi nhấn vào liên kết **ConvertDollar2Current** trên trang Web Service, màn hình sau xuất hiện cho phép nhập vào tham số Web Service như sau:



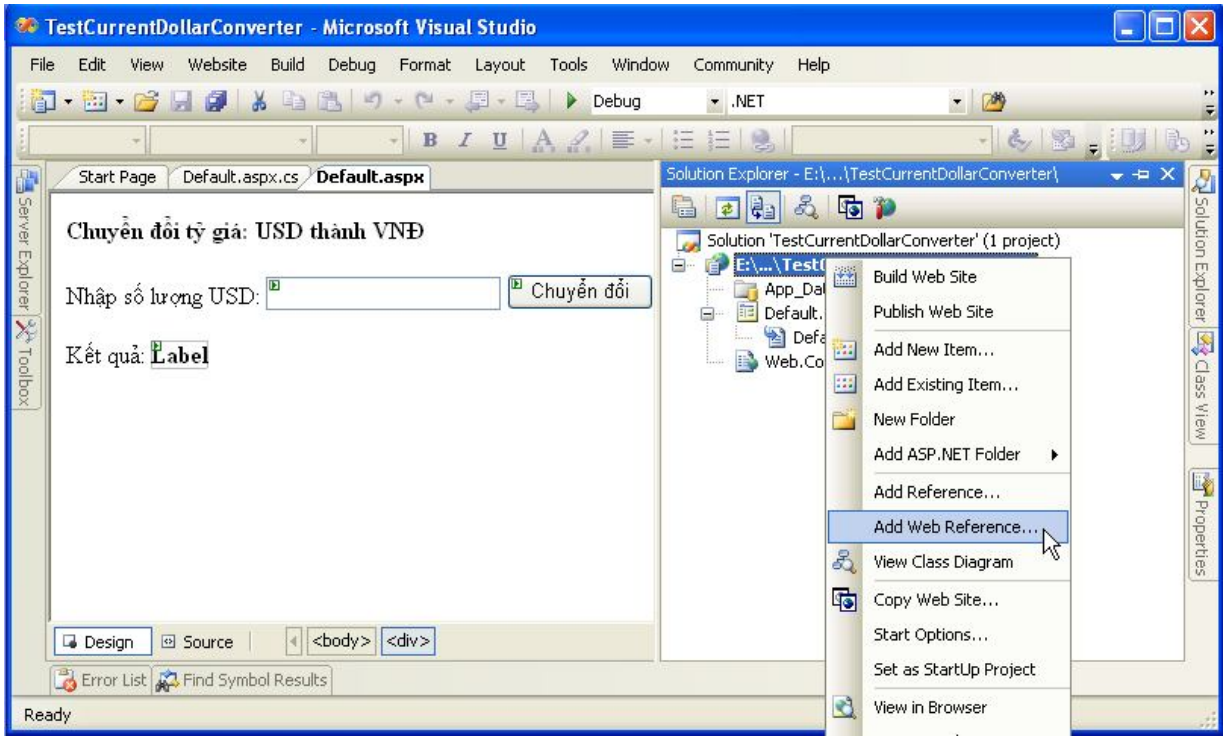
9. Tại ô **giá trị (value:)**, bạn nhập vào một số tương ứng với số dollar để chuyển đổi, và nhấn **Invoke**
10. Kết quả xuất hiện như sau:



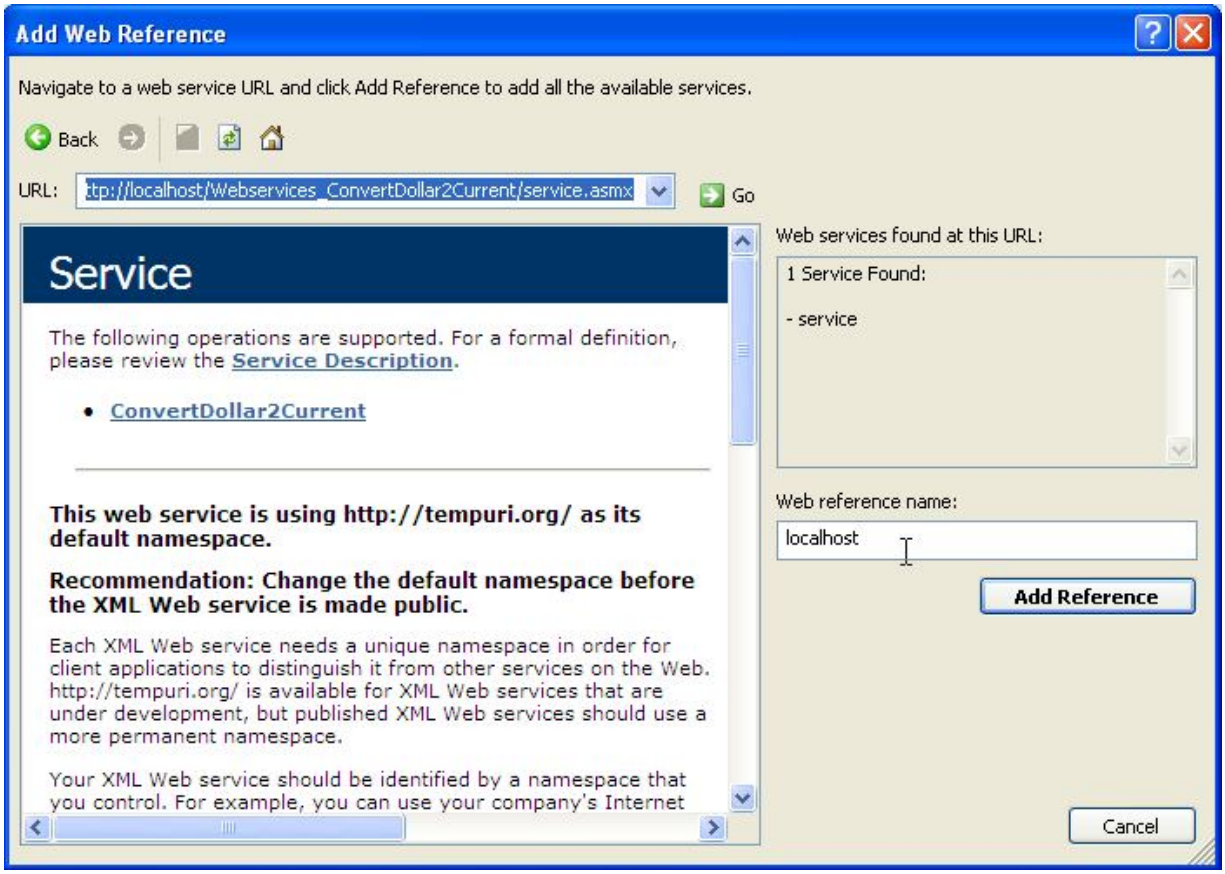
Sau khi kiểm tra ứng dụng, tiến hành build lớp Web Service để tạo thư viện dựa trên liênkết động (DLL), bằng cách chọn menu **Build**, **Build Solution**. Bây giờ Web Service đã có thể sử dụng tại bất kỳ ứng dụng nào.

Bạn sẽ tiến hành thử nghiệm gọi và thực thi Web Services này từ một ứng dụng Web khác, nhưng trước khi tiến hành gọi, bạn phải triển khai Web Services vừa tạo, việc triển khai Web Service tương tự như việc ***triển khai một ứng dụng web ASP.NET*** mà bạn đã được học trong phần trước. Sau khi triển khai Webservice, bạn thực hiện các bước sau.

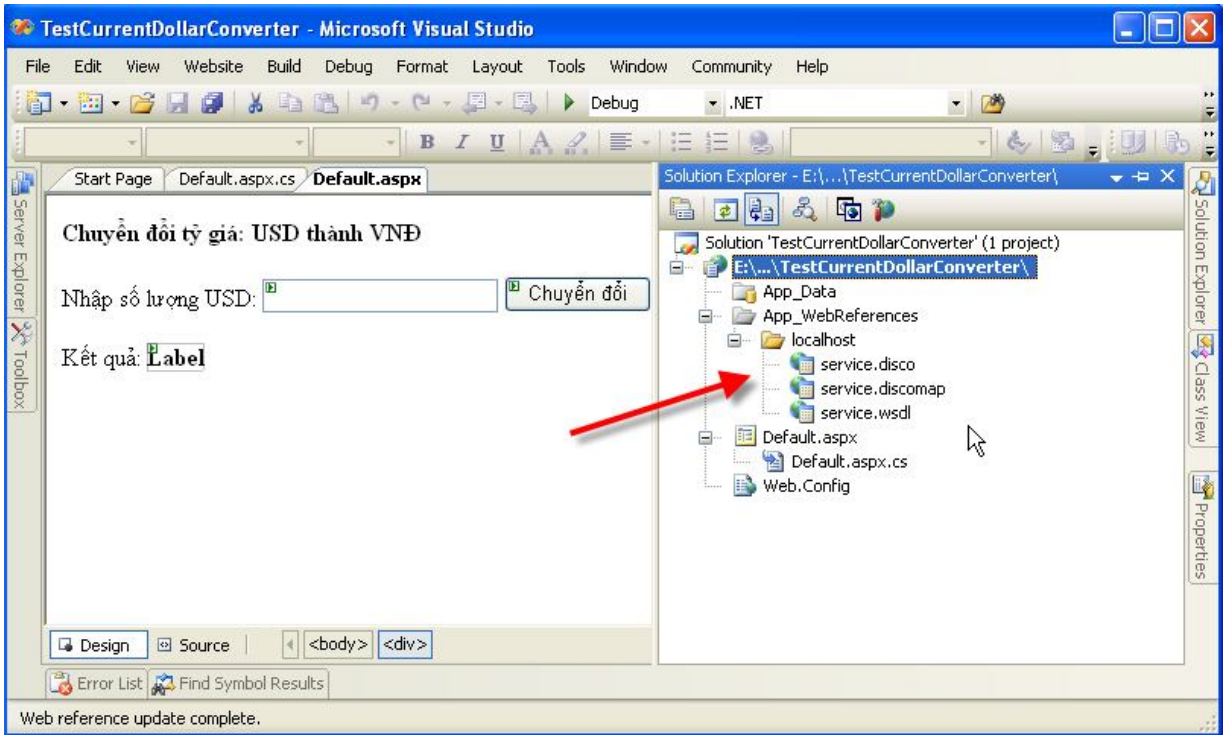
- Mở ứng dụng ConvertDollar2Current đã thực hiện ở phần trước.
- Tại cửa sổ **Solution Explorer**, chọn project và nhấn chuột phải, popup menu xuất hiện, chọn **Add Web References…** như mô tả sau:



- Cửa sổ **Add Web Reference** xuất hiện, tại mục **URL**, bạn gõ http://localhost/Webservices_ConvertDollar2Current/service.asmx là đường dẫn của Web Service chúng ta vừa triển khai, nhấn nút **Go**, kết quả như sau:



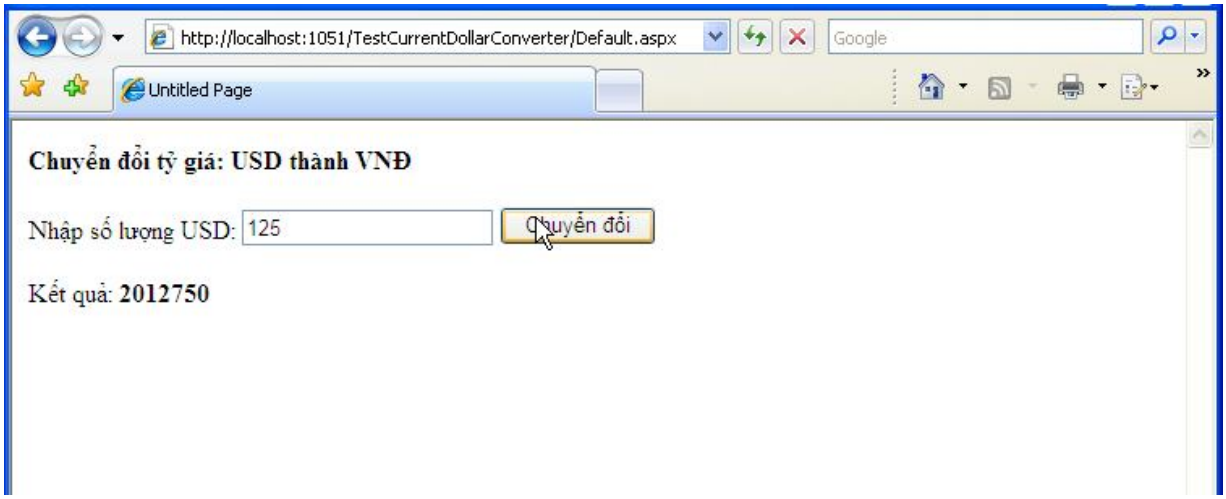
- Danh sách những hàm trong Webservice bạn xây dựng được liệt kê tại pane bên dưới. tại mục Web reference name, bạn gõ localhost
- Nhấn **Add Reference**



- Web Service đã được add vào ứng dụng và sẵn sàng sử dụng.
- Tại sự kiện của nút "Chuyển đổi", bạn thực hiện như sau:

```
protected void btnconvert_Click(object sender, EventArgs e)
    {
        //khong su dung webservice
        //this.lblresult.Text = (Convert.ToDouble(this.txtdollar.Text.Trim()) *
16102).ToString();
        // su dung web service
        // Khai báo đối tượng của class Service trong WebService
        localhost.Service ws_dollar2current = new localhost.Service();
        // Gọi hàm từ đối tượng Web Service
        this.lblresult.Text =

ws_dollar2current.ConvertDollar2Current(Convert.ToDouble(this.txtdollar.Text)).
ToString();
    }
```

- Để đổi dollar ra tiền Việt, trước đây chúng ta thực hiện phép tính, nhưng khi ứng dụng webservice, bạn chỉ cần khai báo đối tượng Web Service và gọi hàm từ đối tượng này.
- Nhấn **F5** để thực thi.



Bạn đã sử dụng một Web Service thành công.

DỰ ÁN

Project 1.

Dự án: Website thông tin và bán hàng trực tuyến

Mô tả:

SunToys là một công ty xuất khẩu đồ chơi trên toàn thế giới. Công việc kinh doanh của họ liên tục phát triển trên thị trường châu Âu, năm tới, họ muốn sản phẩm của mình có mặt tại thị trường Mỹ. SunToys yêu cầu xây dựng một hệ thống phần mềm cho phép khách hàng tại thị trường Mỹ có thể xem thông những sản phẩm của họ. Hơn thế nữa, SunToys đã ký kết với hãng chuyển phát sản phẩm tại Mỹ cho phép sản phẩm của SunToys đến tay từng người tiêu dùng cá nhân. Sản phẩm sau khi được chọn mua, sẽ được phân phối sau ba ngày. Giám đốc công ty quyết định sẽ xây dựng một hệ thống cho phép quản lý số lượng hàng hóa sản xuất và tiếp nhận đơn đặt hàng từ khách hàng, và theo dõi quá trình phân phối sản phẩm.

Là một công ty phần mềm chuyên nghiệp. bạn hãy thực hiện yêu cầu trên.

Project 2.

Dự án: Hệ thống quản trị kho hàng

Mô tả:

INGA là tập đoàn phân phối hệ thống máy chủ hàng đầu tại Việt Nam, công ty có hai trụ sở chính tại Sài Gòn và Hà Nội. Mỗi ngày, công ty phải giải quyết rất nhiều đơn đặt hàng của khách hàng. INGA có một kho hàng chính tại Sài Gòn, công ty muốn xây dựng một hệ thống quản trị kho hàng và quản lý đơn đặt hàng, bán hàng với những yêu cầu sau. Hệ thống được cài đặt tại máy chủ tại Sài Gòn, và ở mọi nơi nhân viên bán hàng đều có thể thực hiện việc tra cứu thông tin hàng hóa và thực hiện giải quyết đơn hàng mà không cần phải cài đặt trên máy tính thứ 2. Hãy thực hiện yêu cầu của INGA.

ĐỌC THÊM

ASP.NET & AJAX Framework

ASP.NET AJAX là công nghệ miễn phí cho phép người lập trình tạo ra những ứng dụng web động. Cho phép bạn tạo ra những website cá nhân đến những hệ thống website lớn có độ tương tác cao. Bạn có thể tải ứng dụng này về từ website ASP.NET có địa chỉ http://www.asp.net

Xem hướng dẫn tạo ứng dụng với ASP.NET AJAX theo sự chỉ dẫn của giảng viên.

Hệ cơ sở dữ liệu MySQL Server 5.0 & lập trình thao tác dữ liệu với MySQL Server.

MySQL là hệ cơ sở dữ liệu mã nguồn mở nổi tiếng nhất thế giới, đây là hệ cơ sở dữ liệu với tốc độ truy cập cao, mạnh mẽ, dễ sử dụng. Được sử dụng bởi nhiều hệ thống lớn như Yahoo, Google, Nokia, YouTube … Việc sử dụng một hệ cơ sở dữ liệu mã mở, miễn phí giúp tiết kiệm chi phí cho doanh nghiệp của bạn. Bạn có thể tải và sử dụng miễn phí hệ cơ sở dữ liệu này thông qua địa chỉ www.mysql.com

MySQL cung cấp khả năng giao tiếp với C# một cách mạnh mẽ và dễ dàng thông qua namespace MySQL.Data.MySqlClient. Cách thức thực hiện giao tiếp dữ liệu với MySQL giống như cách thức mà bạn làm việc với ADO.NET đã trình bày ở phần trước.

Để sử dụng namespace MySQLClient, đòi hỏi bạn phải tham chiếu đến thư viện này bằng chọn **References** di chuyển đến tệp tin **MySQL.Data.dll**. Nhấn **OK** để thực hiện

Sau khi đã tham chiếu file MySQL.Data.dll vào project, để sử dụng các hàm trong thư viện này, bạn phải khai báo sử dụng thư viện.

```
using MySql.Data;
using MySql.Data.MySqlClient;
```

Từ đây bạn có thể sử dụng những hàm, thủ tục trong thư viện MySQL.

Kết nối đến cơ sở dữ liệu MySQL

Đoạn mã lệnh sau thực hiện việc kết nối đến cơ sở dữ liệu MySQL bằng lập trình C#.

```
MySqlConnection con = new
MySqlConnection("server=localhost;database=_net;uid=root;pwd=root");
con.Open();
Response.Write("Connected into MySQL Database");
con.Close();
```

Sau khi đã kết nối đến cơ sở dữ liệu MySQL, bạn có thể thực hiện một câu truy vấn đến một bảng trong cơ sở dữ liệu này, đoạn mã lệnh sau thực hiện truy vấn đến một bảng và hiển thị số dòng dữ liệu có trong bảng này.

```
MySqlConnection con = new
MySqlConnection("server=localhost;database=_net;uid=root;pwd=root");
        con.Open();
        Response.Write("Connected into MySQL Database");
        MySqlCommand cmd = new MySqlCommand("select * from list",con);
        cmd.CommandType = CommandType.Text;
        MySqlDataAdapter da = new MySqlDataAdapter(cmd);
        DataSet ds = new DataSet();
```

```
        da.Fill(ds, "list");
        Response.Write(ds.Tables["list"].Rows.Count.ToString());
        con.Close();
```

Hai ví dụ trên cho thấy, lập trình kết nối cơ sở dữ liệu MySQL sử dụng Namespace MySQL.Data.MySqlClient cũng giống như làm việc với hệ cơ sở dữ liệu MSSQL thông qua ADO.NET. Ví dụ sau sẽ mô tả rõ hơn về vấn đề này.

```
MySqlConnection con = new
MySqlConnection("server=localhost;database=_net;uid=root;pwd=root");
        con.Open();
        Response.Write("Connected into MySQL Database");
        // thực hiện thêm mới dữ liệu vào CSDL
        cmd = new MySqlCommand("insert into list values(0,'New record
1','Description Record 1')");
        cmd.Connection = con;
        cmd.ExecuteNonQuery();
        Response.Write("<br>Inserted into MySql DB");

        // thực hiện cập nhật dữ liệu
        cmd = new MySqlCommand("update list set name='Updated Name' where
id=?id");
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.Add("?id", "1");
        cmd.Connection = con;
        cmd.ExecuteNonQuery();
        con.Close();
```

Kết nối với CSDL sử dụng SQLClient

Regular Expressions

Regular Expression là một chuỗi những ký tự dùng để biểu diễn (thay thế) cho một chuỗi nào đó. Regular Expression là một kỹ thuật khó sử dụng phương pháp so sánh. Bảng sau thể hiện những biểu thức thường sử dụng nhất.

Mô tả kiểm tra	Công thức	Mô tả	Ví dụ
Một tập hợp ký tự	[ ]	Được sử dụng để so sánh bất kỳ một ký tự nằm trong [ ]. Có thể được sử dụng để xác định một chuỗi ký tự bắt đầu và kết thúc bằng việc sử dụng dấu gạch ngang, như [a-z]	"P[0-5]" Kiểm tra mã sản phẩm bắt đầu với ký tự P và theo sau bởi những số trong khoảng 0 đến 5.
	\w	Được sử dụng để so sánh từ, số, ký tự gạch dưới	"\w{8,20}" kiểm tra mật khẩu với độ dài nhỏ nhất là 8 ký tự và lớn nhất là 20 ký tự.

Mô tả kiểm tra	Công thức	Mô tả	Ví dụ
Khoảng	{n}	Được sử dụng để so sánh một biểu thức lập lại n lần	“P[0-9] {4}” kiểm tra một mã sản phẩm bắt đầu bằng ký tự P và theo sau bởi 4 số nằm trong khoảng 0 đến 9
Ký tự bất ký	.	Được sử dụng để kiểm tra một ký tự bất kỳ ngoại trừ ký tự xuống dòng.	“2.12.2008” kiểm tra định dạng dữ liệu có dạng: 1/12/2008, 2-12-2008, 2 12 2008 và cả 2.12.2008.
Một ký tự không phải ký tự khoảng trắng	\S	Được dùng để kiểm tra một ký tự bất kỳ ngoại trừ khoảng trắng, tabs, và ký tự xuống dòng.	“http://\S\S\S\S\S\S.\S\S\S.\S\S” kiểm tra địa chỉ web có được nhập đúng như, http://itgatevn.com.vn
Một hoặc nhiều ký tự	+	Sử dụng để kiểm tra có ít nhất một sự xuất hiện trong biểu thức trước nó	“\S+@\S+\. S+” Kiểm tra một địa chỉ mail như pta30000@gmail.com
Bất ký một ký tự đơn	?	Được sử dụng để kiểm tra bất kỳ một ký tự đơn nào trước nó	“programs? “ xác định ký tự s trong chữ programs có thể có hoặc không có.
Một ký tự trắng bất kỳ	\s	được sử dụng để kiểm tra một ký tự bất kỳ bao gồm: khoảng trắng, tabs, và ký tự xuống dòng.	“\smore” kiểm tra khoảng trắng trước một từ.
Một số bất kỳ	\d	Được sử dụng để kiểm tra một số trong khoảng 0 đến 9	“\d{5} Kiểm tra một số có năm chữ số như 11101
Ký tự Escape	\	Được sử dụng để kiểm tra một ký tự theo sau dấu gạch chéo ngược	“\~” kiểm tra người dùng đã nhập ký tự ~ vào chưa.
Không có hoặc có nhiều chữ	*	Được sử dụng để kiểm tra không có hoặc có nhiều ký tự trong biểu thức	“P*\d{3}” kiể tra đầu vào của một mã sản phẩm bắt đầu bằng ký tự P như P333,P777

Regular Expression giúp kiểm tra dữ liệu do người dùng nhập vào từ ứng dụng, để tránh những lỗi xuất phát từ việc nhập dữ liệu của chương trình.

Một số ứng dụng của Regular Expressions

Hàm kiểm tra tính hợp lệ của email.

```
public static bool IsValidEmail(string email)
{
        return System.Text.RegularExpressions.Regex.IsMatch(email,
@"^.+\@(\[?)[a-zA-Z0-9\-\.]+\.([a-zA-Z]{2,3}|[0-9]{1,3})(\]?)$");
}
```

Hàm kiểm tra xem đường dẫn địa chỉ web có hợp lệ hay không

```
public static bool IsValidURL(string url)
{
         return System.Text.RegularExpressions.Regex.IsMatch(url,
@"^(http|https|ftp)\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(:[a-zA-Z0-9]*)?/?([a-zA-
Z0-9\-\._\?\,\'/\\\+&%\$#\=~])*[^\.\,\)\(\s]$");
}
```

Hàm sau dùng để kiểm tra một chuỗi có dạng là số interger hay không.

```
public static bool IsValidInt(string val)
{
        return System.Text.RegularExpressions.Regex.IsMatch(val, @"^[1-
9]\d*\.?[0]*$");
}
```

Một số bài đọc khác

Định dạng hiển thị cho kiểu dữ liệu Double, Float trong C#

Những ví dụ sau chỉ cho bạn cách thức định dạng những số kiểu fload sang kiểu ký tự trong CS bằng cách sử dụng hàm static của lớp String là String.Format.

Số sau dấu chấm

Ví dụ định dạng này chuyển đổi một số double sang string với kết thú là 2 chữ số thập phân đằng sau. Định dạng sử dụng là: „0.00".

```
String.Format("{0:0.00}", 123.4567); // "123.46"
String.Format("{0:0.00}", 123.4); // "123.40"
String.Format("{0:0.00}", 123.0); // "123.00"
```

Ví dụ tiếp theo cho phép bạn rút gọn những số thập phân thừa đằng sau dấu chấm, sử dụng định dạng „0.##".

```
String.Format("{0:0.##}", 123.4567); // "123.46"
String.Format("{0:0.##}", 123.4); // "123.4"
String.Format("{0:0.##}", 123.0); // "123"
```

Số trước dấu chấm thập phân

Sử dụng định dạng: „00.0, chúng ta sẽ được kết quả là số trước dấu chấm lúc nào cũng lớn hơn hoặc băng 2, và sau nó là một chữ số..

```
String.Format("{0:00.0}", 123.4567); // "123.5"
String.Format("{0:00.0}", 23.4567); // "23.5"
String.Format("{0:00.0}", 3.4567); // "03.5"
String.Format("{0:00.0}", -3.4567); // "-03.5"
```

Cách phần ngàn

Sử dụng định dạng mẫu: „0,0.0"

```
String.Format("{0:0,0.0}", 12345.67); // "12,345.7"
String.Format("{0:0,0}", 12345.67); // "12,346"
```

Số 0

```
String.Format("{0:0.0}", 0.0); // "0.0"
String.Format("{0:0.#}", 0.0); // "0"
String.Format("{0:#.0}", 0.0); // ".0"
String.Format("{0:#.#}", 0.0); // ""
```

Và sau đây là một số ví dụ vui.

```
String.Format("{0:my number is 0.0}", 12.3); // "my number is 12.3"
String.Format("{0:0aaa.bbb0}", 12.3); // "12aaa.bbb3"
```

Gởi Email từ một trang ASP.NET

Gởi email là một yêu cầu cần phải thực hiện đối với một ứng dụng web, chẳng hạn như người sử dụng đăng ký thành viên của trang web của bạn, bạn sẽ gởi một email để xác nhận việc đăng ký này. ASP.NET cung cấp cho bạn thư viện để làm việc với email thông qua namespace System.Web.Mail.

Đoạn mã lệnh sau đây cho phép gởi một HTML Email

```
MailMessage msg = new MailMessage();
        StringWriter strwriter = new StringWriter();
        HtmlTextWriter htmltxtwriter = new HtmlTextWriter(strwriter);
        htmltxtwriter.RenderBeginTag("html");
        htmltxtwriter.RenderBeginTag("head");
        htmltxtwriter.RenderBeginTag("title");
        htmltxtwriter.Write("Thank You!");
        htmltxtwriter.RenderEndTag();
        htmltxtwriter.RenderEndTag();
        htmltxtwriter.RenderBeginTag("body");
        htmltxtwriter.Write("Thank you for registering with Itgatevn.com.vn");
        htmltxtwriter.RenderEndTag();
        htmltxtwriter.RenderEndTag();

        msg.From = "ITGATEVN.COM.VN";
        msg.To = this.TextBox1.Text.Trim();
        msg.Bcc = "pta30000@gmail.com";
        msg.Subject = "Thanks for registering";
        msg.Body = strwriter.ToString();
        msg.BodyFormat = MailFormat.Html;
        SmtpMail.Send(msg);
```

Upload file hình ảnh vào cơ sở dữ liệu SQL

ASP.NET cung cấp một cách thức đơn giản để cập nhật hình ảnh vào cơ sở dữ liệu SQL Server. Hình ảnh có thể được lưu trữ trong một cơ sở dữ liệu với thuộc tính của bảng là kiểu image.

Bạn có thể sử dụng câu lệnh SQL sau để tạo một bảng cho phép lưu trữ hình ảnh:

```
CREATE  TABLE [dbo].[_image](
[Img _id]         [int]  identity(1,1)  CONSTRAINT  [pk_image]  PRIMARY  KEY  NOT
NULL,
[Img _name]       [varchar] (30)    not NULL,
[Img _file]            [IMAGE] NULL,
[Img _type]       [varchar] (30) NULL
)
ON [PRIMARY] Textimage_on [PRIMARY]
```

Trong đó,

- Img_id: lưu trữ id của hình và là khóa chính của bảng
- Img_name: lưu trữ tên của hình ảnh.
- Img_file: lưu trữ hình ảnh
- Img_type: lưu trữ kiểu nội dung của hình ảnh

Bây giờ chúng ta tiến hành thực hiện các bước để upload hình vào cơ sở dữ liệu

1. Tạo một Webform để upload file hình
2. Upload hình vào cơ sở dữ liệu.

Tạo Webform để tải file hình ảnh.

Bạn tạo form upload hình như sau:

```
<form enctype="multipart/form-data" id="form1" runat="server">
    <div>
        <table id="table1" border="0" cellpadding="0" style="border-collapse:
collapse"
            width="100%">
            <tr>
                <td width="125">
                     Image Name</td>
                <td width="10">
                     </td>
                <td>
                    <asp:TextBox ID="txtname"
runat="server"></asp:TextBox></td>
            </tr>
            <tr>
                <td>
                     File to Upload</td>
                <td>
                     </td>
                <td>
                    <input id="fileimage" runat="server" type="file" /></td>
            </tr>
            <tr>
                <td>
                     </td>
                <td>
```

```
                     </td>
                <td>
                    <asp:Button ID="btupload" runat="server" Text="Upload"
OnClick="btupload_Click" /></td>
            </tr>
            <tr>
                <td>
                     </td>
                <td>
                     </td>
                <td>
                     </td>
            </tr>
        </table>

    </div>
</form>
```

Tải file hình vào cơ sở dữ liệu.

Khi người dùng nhấn vào nút Upload, sự kiện btupload_Click được gọi. Bạn có thể copy đoạn mã lệnh sau vào sự kiện btupload_Click

```
protected void btupload_Click(object sender, EventArgs e)
    {
        // tạo đối tượng Stream
        System.IO.Stream img_strm = fileimage.PostedFile.InputStream;
        // nhận kích thước của hình
        int img_len = fileimage.PostedFile.ContentLength;
        // nhận kiểu hình
        string strType = fileimage.PostedFile.ContentType;
        string strName = txtname.Text;
        // tạo mảng kiểu byte của file đã được upload
        byte[] imgData = new byte[img_len];
        int n = img_strm.Read(imgData, 0, img_len);
        int result = SaveToDB(strName, imgData, strType);

    }
// Hàm SaveToDB()

int SaveToDB(string imgName, byte[] imgbin, string imgcontenttype)
    {
        OleDbConnection con = new
OleDbConnection("Provider=SQLOLEDB;server=(local);database=_NET;uid=_net;pwd=");

        OleDbCommand cmd = new OleDbCommand("INSERT INTO _image values(?,?,?)",
con);
        OleDbParameter param0 = new
OleDbParameter("@imgName",OleDbType.VarWChar, 30);
        param0.Value = imgName;
        cmd.Parameters.Add(param0);

        OleDbParameter param1 = new OleDbParameter("@imgName",
OleDbType.Binary);
        param1.Value = imgbin;
        cmd.Parameters.Add(param1);

        OleDbParameter param2 = new OleDbParameter("@imgType",
OleDbType.VarWChar, 30);
```

```
        param2.Value = imgcontenttype;
        cmd.Parameters.Add(param2);
        con.Open();
        int numofwsAffected = cmd.ExecuteNonQuery();

        con.Close();
        return numofwsAffected;
    }
```

Và bây giờ, file hình đã được lưu trữ trong cơ sở dữ liệu của bạn.

THAM KHẢO

Các bạn có thể tham khảo thêm kiến thức tại một số địa chỉ sau đây

- Thư viện bài viết công nghệ ITGATEVN - http://www.itgatevn.com.vn/articles/
- Thư viện MSDN: http://msdn.microsoft.com