## Cơ bản về lớp trong C#

### 5.1 Khai báo Field và thuộc tính

Ví dụ về Field

```
public class HelloWorld
{
    public string _Message;
    public string SayMessage()
    {
        return _Message;
    }
}
```

Trong đoạn mã trên bạn thấy Field _Message được khai báo kiểu string và bổ ngữ truy cập là public, và _Message được trả về giá trị bởi phương thức SayMessage().

Ví dụ về thuộc tính

```
public class HelloWorld
{
   public string _Message;
   public string Message
   {
      get { return _Message; }
      set { _Message = value; }
   }
}
```

Một thuộc tính Message được khai báo ở trên gồm 2 phương thức get trả về giá trị cho Message và phương thức set thiết lập giá trị cho Message. Thuộc tính Message ở trên là phương thức vừa đọc vừa ghi. nếu bạn xây dựng thuộc tính chỉ đọc thì bạn chỉ cung cấp phương thức get hay thuộc tính chỉ ghi bạn cung cấp cho thuộc tính đó phương thức set.

### 5.2 Phương thức khởi dựng của lớp

Phương thức khởi dựng là phương thức đặc biệt của lớp, nó được gọi tự động khi khởi tạo mới lớp đó.bạn sử dụng phương thức khởi dụng để khởi tạo các private fields chứa đựng trong lớp. Phương thức khởi dựng của lớp phải trùng với tên của lớp, 1 phương thức của lớp có thể có đối số hoặc không có đối số, và có thể có nhiều phương thức khởi dựng cho lớp nhưng các đối số trong các phương thức phải khác nhau.

Ví dụ:

Xây dựng lớp: Construction.cs

```
using System;
public class Construction
{
    int _giatri1;
    int _giatri2;
      public Construction()
      {
```

```
        _giatri1 = 0;
        _giatri2 = 0;
      }

    public Construction(int _giatri1, int _giatri2)
    {
        this._giatri1 = _giatri1;
        this._giatri2 = _giatri2;
    }

    public int Sum()
    {
        return _giatri1 + _giatri2;
    }
}
```

Trong lớp này chúng ta xây dựng hai phương thức khởi dựng một phương thức không có đối số và một phương thức có đối số, và một hàm tính tổng của 2 giá trị nó được sử dụng trang trang asp.net như sau:

Trang UseConstruction.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="UseConstruction.aspx.cs" Inherits="UseConstruction" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Sử dụng phương thức khởi dựng của lớp</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Label ID="lblhello" runat="server" Text="Label"></asp:Label>
    </div>
    </form>
</body>
</html>
```

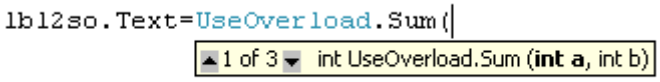
Trang UseConstruction.aspx.cs

```
using System;

public partial class UseConstruction : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
      Construction construc = new Construction(5, 6);
      lblhello.Text = "Giá trị là: " + construc.Sum().ToString();
   }
}
```

**5.3 Overloading phương thức**

Khi một phương thức được overloaded có nghĩa là hai phương thức có tên trùng nhau nhưng các đối số của nó phải khác nhau. Khi trong lớp của bạn có các phương thức overload thì bạn gọi hàm VS sẽ xuất hiện như sau để bạn có thể dễ dàng chọn được phương thức mình cần gọi.




Ví dụ:

Bạn tạo một lớp
Lớp UseOverload.cs

```
using System;

public class UseOverload
{
    public static int Sum(int a, int b)
    {
        return a + b;
    }

    public static int Sum(int a, int b, int c)
    {
        return a + b + c;
    }

    public static int Sum(int a, int b, int c, int d)
    {
        return a + b + c + d;
    }
}
```

Trong lớp này gồm 3 hàm tính tổng lần lượt được đưa vào 2,3,4 đối số

Sử dụng lớp này trong trang ASP.NET

Trang Overloading.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Overloading.aspx.cs"
Inherits="Overloading" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <h1>Chồng hoá phương thức tính tổng</h1>
     Tổng 2 số:<asp:Label ID="lbl2so" runat="server"
Text="Label"></asp:Label><br /><br />
     Tổng 3 số:<asp:Label ID="lbl3so" runat="server"
Text="Label"></asp:Label><br /><br />
```

```
     Tổng 4 số:<asp:Label ID="lbl4so" runat="server"
Text="Label"></asp:Label><br /><br />
    </div>
    </form>
</body>
</html>
```

Trang Overloading.aspx.cs

```
using System;

public partial class Overloading : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        lbl2so.Text = UseOverload.Sum(5, 5).ToString();
        lbl3so.Text = UseOverload.Sum(5, 5, 5).ToString();
        lbl4so.Text = UseOverload.Sum(5, 5, 5, 5).ToString();
    }
}
```

Trong lớp này bạn gọi lần lượt các phương thức tính tổng với 2,3,4 đối số để truyền giá trị vào các Label tương ứng cùng tên.

Kết xuất của chương trình:


### 5.4 Khai báo không gian tên (Namespaces)

Nếu bạn từng lập trình java chắc hẳn bạn đã quen với khái niệm packed mà bạn để đóng gói các lớp mà bạn xây dựng có đặc tính chung(miêu tả hay xử lý vấn đề gì đó). Trong .Net cũng vậy từ khoá Namespaces cũng có nhiệm vụ như packed trong java.

.Net cung cấp cho chúng ta các Namespaces như:

```
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;
```

Và để sử dụng các Namespaces trong C# bạn cần sử dụng từ khoá using. Một Namespaces có thể chứa các Namespaces con, và trong Namespace con nhất chứa các lớp thành viên

Ví dụ

Bạn tạo ra hai lớp phép cộng và phép trừ để thực hiện các phép toán tương ứng như sau:

Lớp Phepcong.cs

```
using System;

namespace Vidu.Tinhtoan
{
    public class Phepcong
    {
        public static int Sum(int a, int b)
        {
            return a + b;
        }
    }

}
```

Và lớp Pheptru.cs

```
using System;

namespace Vidu.Tinhtoan
{
    public class Pheptru
    {
        public static int Minus(int a, int b)
        {
            return a - b;
        }
    }

}
```

Như bạn thấy hai lớp này nằm trong Namespaces Vidu.Tinhtoan, thì Vidu là Namespaces lớn nhất, còn Namespaces Tinhtoan là con của Vidu và trong tính toán chứa các lớp Phepcong và Pheptru.

Sử dụng Namespaces này trong trang asp.net

Trang Namespaces.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Namespaces.aspx.cs"
Inherits="Namespaces" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
      <h1>Khai báo và sử dụng Namespaces</h1>
      Lớp phép cộng:
        <asp:Label ID="lblcong" runat="server" Text="Label"></asp:Label><br
/><br />
```

```
      Lớp phép trừ:
        <asp:Label ID="lbltru" runat="server" Text="Label"></asp:Label>
    </div>
    </form>
</body>
</html>
```

Lớp Namespaces.aspx.cs

```
using System;
using Vidu.Tinhtoan;
public partial class Namespaces : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        lblcong.Text = Phepcong.Sum(5, 5).ToString();
        lbltru.Text = Pheptru.Minus(5, 5).ToString();
    }
}
```

Như ví dụ trên bạn thấy chúng ta sử dụng namespace Vidu.Tinhtoan giống với các Namespace khác mà Microsoft cung cấp cho chúng ta.

Kết xuất của chương trình

# Khai báo và sử dụng Namespaces

Lớp phép cộng: 10

Lớp phép trừ: 0

### 5.5 Lớp Partial

.Net cho phép chúng ta tạo ra một lớp trong nhiều file khác nhau mỗi File cung cấp hay xử lý một công việc gì đó trên lớp đó.

Ví dụ sau đây chúng ta sẽ tạo một lớp Calculator với 2 phương thức cộng và trừ nằm trên hai File khác nhau.

File Calminus.cs

```
using System;

namespace Vidu.Tinhtoan
{
    public partial class Calculator
    {
        public static int Minus(int a, int b)
        {
            return a - b;
        }
    }
}
```

File Calsum.cs

```
using System;
```

```
namespace Vidu.Tinhtoan
{
    public partial class Calculator
    {
        public static int Sum(int a, int b)
        {
            return a + b;
        }
    }
}
```

Như các bạn thấy hai file Calsum và Calminus chứa đựng cùng một tên lớp Calculator và trong mỗi File chứa đựng một phương thức riêng là thành phần của lớp đó.

Sử dụng lớp này hoàn toàn giống với việc sử dụng một lớp khác.

File UsePartial.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="UsePartial.aspx.cs"
Inherits="UsePartial" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Sử dụng lớp Partial</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
     <h1>Lớp Partial</h1>
     Kết quả cộng:
        <asp:Label ID="lblcong" runat="server" Text="Label"></asp:Label><br
/><br />
     Kết quả trừ:
        <asp:Label ID="lbltru" runat="server" Text="Label"></asp:Label>
    </div>
    </form>
</body>
</html>
```

Và File UsePartial.aspx.cs

```
using System;
using Vidu.Tinhtoan;

public partial class UsePartial : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        lblcong.Text = Calculator.Sum(5, 5).ToString();
        lbltru.Text = Calculator.Minus(5, 5).ToString();
    }
}
```

### 5.6 Kế thừa và trừu tượng hoá một lớp

Khi một lớp được kế thừa từ một lớp thứ 2 thì nó được thừa hưởng tất cả các thuộc tính và phương thức không private từ lớp thứ nhất.

Kế thừa được sử dụng thông suốt trong .NetFrameWork, ví dụ trong tất cả các trang ASP.NET đều được kế thừa từ Lớp System.Web.UI.Page và tất cả các lớp trong .Net đều được dẫn xuất từ lớp cơ sở System.Object.

Ví dụ sau chúng ta sẽ đưa ra 2 lớp TelevisionProduct và ComputerProduct được kế thừa từ lớp BaseProduct.

Ví dụ:

```
using System;
public class BaseProduct
{
    decimal _price;
    public decimal Price
    {
        get { return _price; }
        set { _price = value; }
    }
}

public class ComputerProduct : BaseProduct
{
    string _processor;
    public string Processor
    {
        get { return _processor; }
        set { _processor = value; }
    }
}

public class TelevisionProduct : BaseProduct
{
    bool _isDHTV;
    public bool isDHTV
    {
        get { return _isDHTV; }
        set { _isDHTV = value; }
    }
}
```

Trong ví dụ trên bạn thấy hai lớp ComputerProduct và TelevisionProduct được kế thừa từ lớp BaseProduct, trong lớp BaseProduct có thuộc tính Price lên hai lớp kế thừa sẽ được kế thừa thuộc tính này.

Khi kế thừa từ một lớp khác, bạn có thể overload các thuộc tính và phương thức trong lớp này. Overloading một thuộc tính hay phương thức là một tiện ích khi bạn muốn thay đổi các ứng xử của phương thức hay thuộc tính đó trong lớp này.

Để Overload một phương thức hay thuộc tính từ lớp cơ sở, thì thuộc tính hay phương thức này phải được đánh dấu với từ khoá vitual hay abstract của C# hay trong VB.NET là Overridable hoặc MustOverride.

Ví dụ, chúng ta sẽ đưa ra hai lớp ProductBase và OnSaleProduct được kế thừa từ lớp ProductBase nhưng nó sẽ overload một thuộc tính từ lớp ProductBase.

Ví dụ:

```
using System;

public class ProductBase
{
    decimal _price;
    public virtual decimal Price
    {
        get { return _price; }
        set { _price = value; }
    }
}

public class OnSaleProduct : ProductBase
{
    public override decimal Price
    {
        get{ return base.Price / 2;}
        set{base.Price = value;}
    }
}
```

Trong ví dụ trên Lớp OnSaleProduct được kế thừa từ lớp ProductBase và Override thuộc tính Price, ở lớp này muốn chỉ ra giá bằng một nửa ở lớp cơ sở. chú ý với VB.NET thì từ khoá base là MyBase.

Bạn có thể dùng từ khoá abstract khi khai báo một lớp để như đánh dấu lớp này yêu cầu kế thừa. Bạn không thể tạo đối tượng một lớp trừu tượng, để sử dụng một lớp trừu tượng bạn phải dẫn xuất một lớp mới từ lớp trừu tượng và tạo đối tượng trong lớp dẫn xuất.

Ví dụ:

```
public abstract class BaseEmployee
{
  public abstract decimal Salary
  {
    get;
  }

  public string Company
  {
    get { return "Hoa Sen"; }
  }
}

public class SaleEmployee:BaseEmployee
{
  public override decimal Salary
  {
    get { return 66.666; }
  }
}
```

Trong ví dụ trên bạn thấy Lớp SaleEmployee được kế thừa từ lớp BaseEmployee và Overload thuộc tính Salary.

### 5.7 Khai báo Interface:

Một giao diện mà một danh sách các thuộc tính hay phương thức mà lớp kế thừa phải cài đặt. nếu một lớp cài đặt một giao diện, thì lớp này sẽ chứa tất cả các thuộc tính cũng như phương thức của giao diện này.

Ví dụ:

```
using System;

public interface IProduct
{
    decimal Price
    {
        get;
    }

    void SaveProduct();
}

public class MusicProduct : IProduct
{
    public decimal Price
    {
        get { return 20.99m; }
    }

    public void SaveProduct()
    {
        //Save Music Product
    }
}

public class BookProduct : IProduct
{
    public decimal Price
    {
        get { return 23.99m; }
    }

    public void SaveProduct()
    {
        //Save Book Product
    }
}
```