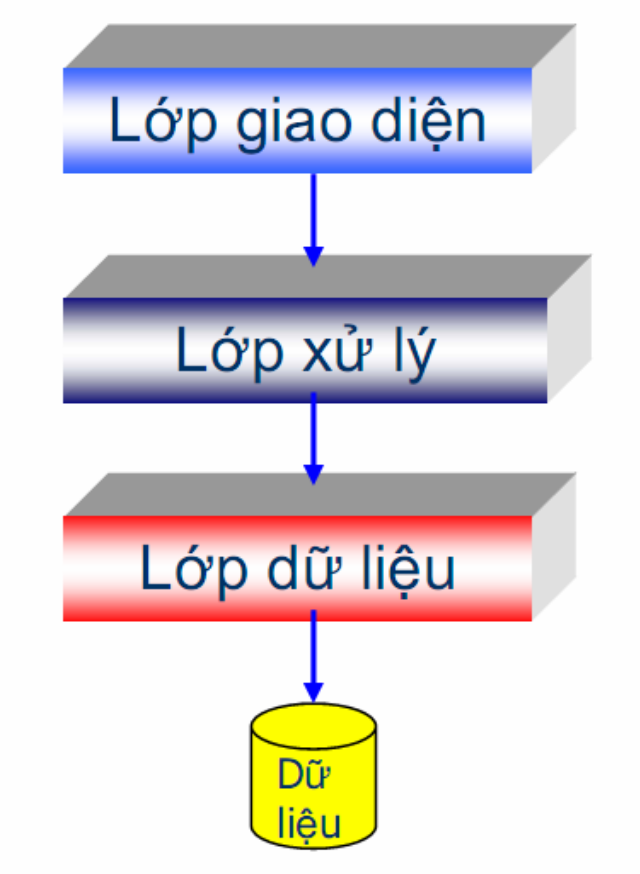
CHƯƠNG 4: XÂY DỰNG ỨNG DỤNG NHIỀU LỚP

4.1. Mô hình 2 lớp (two tier), 3 lớp (three tier) và n lớp.

Trước đây, đối với các phần mềm có sử dụng liên quan đến dữ liệu, thường khi làm người lập trình thường tích hợp việc giao tiếp với người sử dụng , xử lý rồi ghi xuống dữ liệu trên cùng một form (đây là mô hình một lớp). Nhưng trong kiến trúc 3 lớp (mô hình 3 lớp), phải có việc phân biệt rạch ròi giữa các lớp này. Mô hình 3 lớp có thể được mô tả như sau:



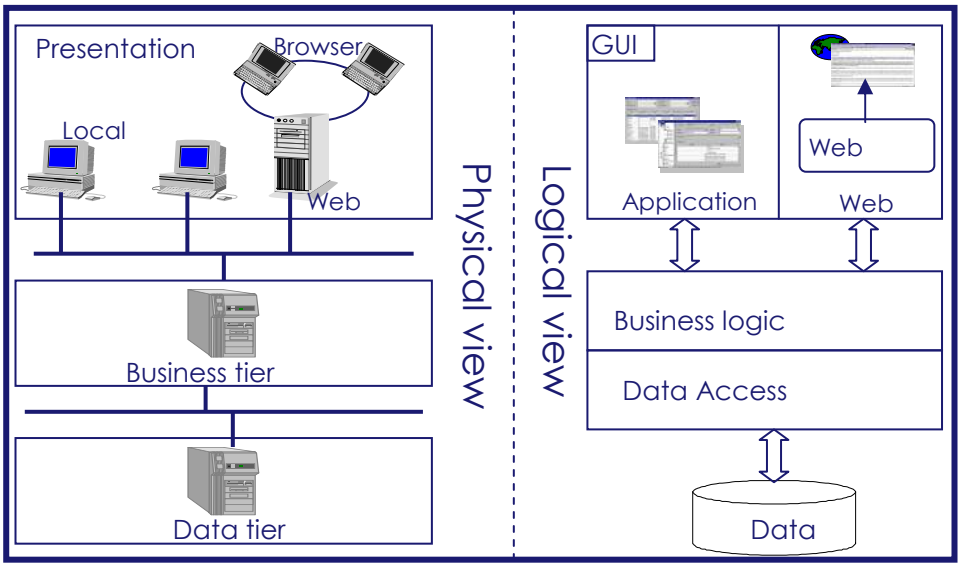
- Lớp thứ nhất : Lớp giao diện (giao tiếp với người sử dụng) : chỉ thuần xử lý việc giao tiếp với người sử dụng, nhập xuất, … mà không thực hiện việc tính toán, kiểm tra, xử lý, hay các thao tác liên quan đến cơ sở dữ liệu.
- Lớp thứ hai : Lớp xử lý : Lớp này chuyên thực hiện các xử lý , kiểm tra các ràng buộc, các qui tắc ứng xử của phần mềm , các chức năng cốt yếu, … Việc thực hiện này độc lập với cách thiết kế cũng như cài đặt giao diện. Thông tin cho lớp này thực hiện các xử lý của mình được lấy từ lớp giao diện.
- Lớp thứ ba : Lớp dữ liệu : Lớp này chuyên thực hiện các công việc liên quan đến dữ liệu. Dữ liệu có thể lấy từ cơ sở dữ liệu (Access, SQL Server …) hoặc tập tin (text, binary, XML …). Đối với cơ sở dữ liệu, lớp này thực hiện kết nối trực tiếp với cơ sở dữ liệu và thực hiện tất cả các thao tác liên quan đến cơ sở dữ liệu mà phần mềm cần thiết. Đối với tập tin, lớp này thực hiện việc đọc, ghi tập tin theo yêu cầu của phần mềm. Việc thực hiện này do lớp xử lý gọi.

Rõ ràng, với mô hình này, các công việc của từng lớp là độc lập với nhau. Việc thay đổi ở một lớp không làm thay đổi các lớp còn lại, thuận tiện hơn cho quá trình phát triển và bảo trì phần mềm.

Một số lưu ý:

- Phân biệt vai trò Business Layer và khái niệm "xử lý"
- Mỗi Layer vẫn có xử lý riêng, đặc trưng của Layer đó
- Đôi khi việc quyết định 1 xử lý nằm ở layer nào chỉ mang tính chất tương đối

Chúng ta cũng cần phân biệt khái niệm 3 tier và 3 layer: 3 tier là mô hình 3 lớp vật lý còn 3 layer là mô hình logic.

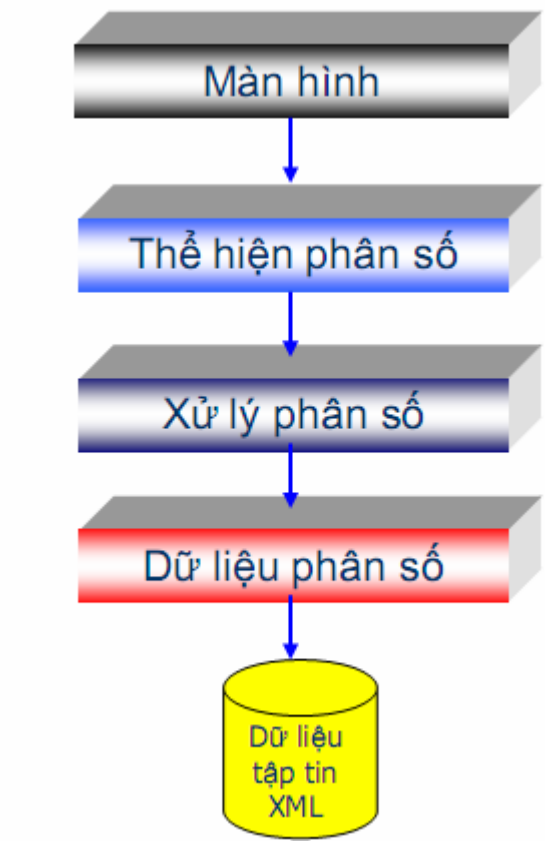


Ví dụ minh họa:

Xây dựng chương trình tính tổng 2 phân số theo kiến trúc 3 lớp. Theo đó dữ liệu của phân số được đọc lên từ tập tin XML, kết quả sau khi được tính sẽ được ghi xuống tập tin XML.

Cách làm thông thường là mọi việc đều được đẩy vào trong 1 form và xử lý trực tiếp trong form đó. Tuy nhiên, khi có sự thay đổi xảy ra về giao diện, xử lý, hay dữ liệu thì việc chỉnh sửa khá khó khăn. Do vậy, việc xây dựng theo kiến trúc 3 lớp sẽ khắc phục nhược điểm này.

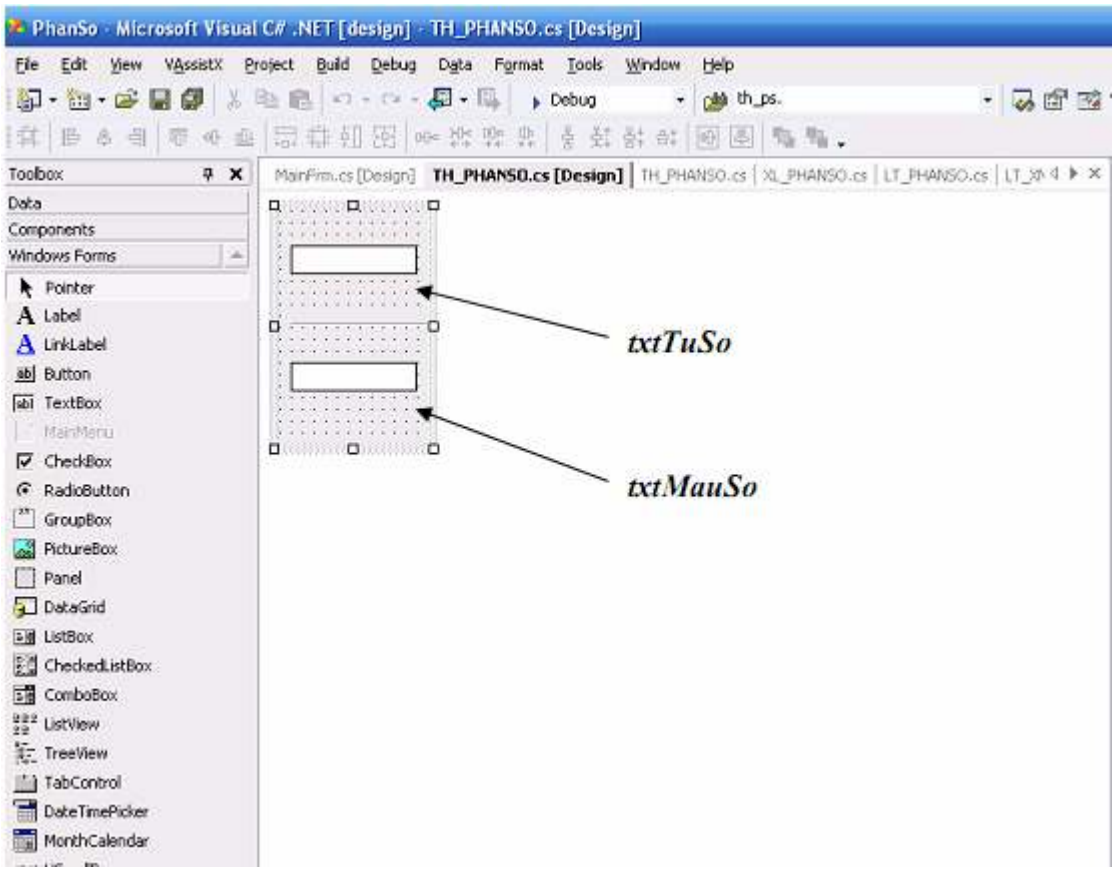
Kiến trúc của chương trình như sau:



Xây dựng lớp thể hiện phân số (TH_PHANSO)

Sử dụng User Control để cài đặt cho TH_PHANSO. Thêm User Control vào project bằng cách chọn Project > Add User Control. Đặt tên User Control đó.

Ta có TH_PHANSO.cs



Do thể hiện tử số và thể hiện mẫu số đều là TextBox do đó trong lớp TH_PHANSO cần thiết lập các properties là tuso và mauso có kiểu int.

```
public int tuso{
  set{
   this.txtTuSo.Text = value.ToString();
 }
  get{
   return int.Parse(this.txtTuSo.Text);
 }
}
 public int mauso {
  set  {
   this.txtMauSo.Text = value.ToString();
 }
  get  {
   return int.Parse(this.txtMauSo.Text);
 }
}
```

Lớp lưu trữ phân số (LT_PHANSO)

Tập tin XML lưu trữ có định dạng như sau:

```
<?xml version ="1.0" encoding = "utf-8"?>
<PHANSO>
 <Tu_so>5</Tu_so>
 <Mau_so>3</Mau_so>
</PHANSO>
```

Để thực hiện việc đọc và ghi dữ liệu XML ta sử dụng DOM.

Khai báo tuso và mauso để thực hiện việc lưu trữ

public int tuso;

public int mauso;

Thực hiện cài đặt hàm khởi tạo mặc định với tham số truyền vào là đường dẫn file XML

```
public LT_PHANSO(string strFilename)
{
  //
  // TODO: Add constructor logic here
  //
  XmlDocument doc = LT_XML.DocTaiLieu(strFilename);
  if(doc == null)
  {
        tuso = 0;
```

```
        mauso = 0;
        return;
 }
 XmlElement ele = doc.DocumentElement;
 tuso = int.Parse(ele.SelectSingleNode("Tu_so").InnerText);
 mauso = int.Parse(ele.SelectSingleNode("Mau_so").InnerText);
}
```

Thực hiện cài đặt hàm ghi phân số với tham số truyền vào là đường dẫn file XML

```
public void GhiPhanSo(string strFilename)
{
  XmlDocument doc = new XmlDocument();
  XmlElement root = doc.CreateElement("PHANSO");
 doc.AppendChild(root);
  XmlElement ele_Tuso =
root.OwnerDocument.CreateElement("Tu_so");
 ele_Tuso.InnerText = this.tuso.ToString();
 root.AppendChild(ele_Tuso);
  XmlElement ele_Mauso =
root.OwnerDocument.CreateElement("Mau_so");
 ele_Mauso.InnerText = this.mauso.ToString();
 root.AppendChild(ele_Mauso);
  LT_XML.GhiTaiLieu(strFilename,doc);
}
```

Lớp lưu trữ XML (LT_XML)

Việc load và save XmlDocument được tách ra thành một lớp riêng là lớp LT_XML

```
public static XmlDocument DocTaiLieu(string strFilename) {
  XmlDocument kq = new XmlDocument();
  try  {
         kq.Load(strFilename);
 }
  catch{
         return null;
 }
  return kq;
}
 public static void GhiTaiLieu(string strFilename, XmlDocument doc) {
  try{
           doc.Save(strFilename);
 }
  catch{
 }
}
```

Lớp xử lý phân số (XL_PHANSO)

Lớp này sẽ thực hiện cài đặt các hàm liên quan đến xử lý và tính toán trên phân số như định nghĩa phép cộng 2 phân số, rút gọn phân số hay cập nhật giá trị từ đối tượng thể hiện.

Khai báo 2 đối tượng lần lượt thuộc về lớp LT_PHANSO và TH_PHANSO để giúp tạo liên kết với tầng xử lý với 2 tầng còn lại là tầng dữ liệu và tầng giao diện.

```
private LT_PHANSO lt_ps = null;
private TH_PHANSO th_ps = null;
```

Cài đặt hàm khởi tạo mặc định để tạo liên kết với đối tượng thể hiện và đối tượng xử lý

```
public XL_PHANSO(LT_PHANSO lt_ps, TH_PHANSO th_ps)
{
  this.lt_ps = lt_ps;
     this.th_ps = th_ps;
  this.th_ps.tuso = this.lt_ps.tuso;
  this.th_ps.mauso = this.lt_ps.mauso;
}
```

Cài đặt phương thức ghi

```
public void Ghi(string strFilename)
{
  this.lt_ps.tuso = this.th_ps.tuso;
  this.lt_ps.mauso = this.th_ps.mauso;
  this.lt_ps.GhiPhanSo(strFilename);
}
```

Cài đặt toán tử +

```
public static XL_PHANSO operator +(XL_PHANSO ps1,XL_PHANSO ps2)
{
  XL_PHANSO kq = new XL_PHANSO(new LT_PHANSO(), new
TH_PHANSO());
  kq.th_ps.tuso = ps1.th_ps.tuso * ps2.th_ps.mauso +
ps2.th_ps.tuso * ps1.th_ps.mauso;
  kq.th_ps.mauso = ps1.th_ps.mauso * ps2.th_ps.mauso;
  return kq;
}
```

Cài đặt hàm cập nhật từ đối tượng xử lý phân số khác

```
public void CapNhat(XL_PHANSO ps)
{
  this.th_ps.tuso = ps.th_ps.tuso;
  this.th_ps.mauso = ps.th_ps.mauso;
}
```

Cài đặt hàm rút gọn phân số

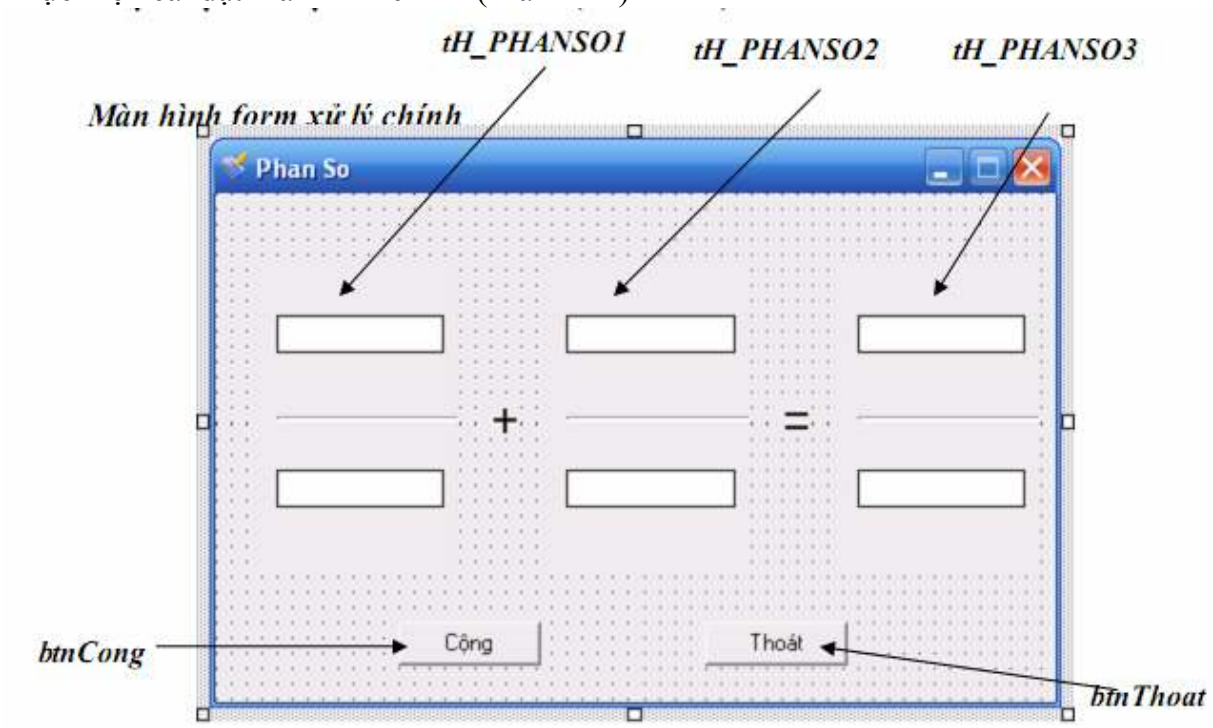
```
public void RutGon()
{
  int tuso = this.th_ps.tuso;
  int mauso = this.th_ps.mauso;
  int maxUC = TimMaxUocChung(tuso,mauso);
  tuso = tuso/maxUC;
  mauso = mauso/maxUC;
```

```
  this.th_ps.tuso = tuso;
  this.th_ps.mauso = mauso;
}
```

Để rút gọn ta cần tính ước chung lớn nhất, có thể cài đặt hàm này chung với lớp XL_PHANSO

```
public int TimMaxUocChung(int a, int b)
{
  while(a!=b)
 {
   if(a>b)
   a -= b;
   else
   b -= a;
 }
  return a;
}
```

Thực hiện cài đặt màn hình chính (MainFrm)



Trong form chính sẽ thực hiện khai báo 3 đối tượng xử lý phân số

```
private PhanSo.XL_PHANSO xl_PhanSo1;
private PhanSo.XL_PHANSO xl_PhanSo2;
private PhanSo.XL_PHANSO xl_PhanSo3;
```

Thực hiện khởi tạo 3 đối tượng xử lý phân số vừa khai báo

```
public MainFrm()
{
  //
  // Required for Windows Form Designer support
 InitializeComponent();
  xl_PhanSo1 = new XL_PHANSO(new LT_PHANSO("phanso1.xml"),
                             tH_PHANSO1);
```

```
 xl_PhanSo2 = new XL_PHANSO(new LT_PHANSO("phanso2.xml"),
                          tH_PHANSO2);
 xl_PhanSo3 = new XL_PHANSO(new LT_PHANSO(""),tH_PHANSO3);
}
```

Viết hàm xử lý cho các nút chức năng trên form:

Hàm xử lý cho nút Cộng

```
private void btnCong_Click(object sender, System.EventArgs e)
{
  XL_PHANSO kq = xl_PhanSo1 + xl_PhanSo2;
 xl_PhanSo3.CapNhat(kq);
 xl_PhanSo3.Ghi("ketqua.xml");
 xl_PhanSo3.RutGon();
}
```

Hàm xử lý cho nút Thoat

```
 private void btnThoat_Click(object sender, System.EventArgs e)
{
  this.Close();
}
```

Tạo các tập tin phanso1.xml, phanso2.xml, có định dạng như ví dụ ở trên.

Thực hiện biên dịch và chạy thử chương trình.

Nhận xét :

Thực hiện cài đặt với kiến trúc 3 lớp sẽ giúp chương trình dễ dàng thay đổi, tái sử dụng lại chương trình.

Ví dụ:

TH_PHANSO không thể hiện tử số và mẫu số bằng TextBox nữa mà thay bằng control khác (ví dụ như MyControl thì cũng không ảnh hưởng, lúc đó chỉ cần thay đổi code trong phần property tử số và mẫu số mà thôi.

```
public int tuso{
  set{
   this.MyControl.Value = value.ToString();
 }
  get{
   return int.Parse(this.MyControl.Value);
 }
}
public int mauso {
  set  {
   this.MyControl.Value = value.ToString();
 }
  get  {
   return int.Parse(this.MyControl.Value);
 }
}
```

Khi không lưu trữ bằng XML mà chuyển sang dùng cơ sở dữ liệu thì ta chỉ cần thay code phần LT_PHANSO, mà không cần thay đổi code phần TH_PHANSO, cũng như XL_PHANSO.

Chú ý:

Không phụ thuộc phương pháp lập trình.

Mỗi nghiệp vụ không nhất thiết chỉ được giải quyết bởi 3 đối tượng.

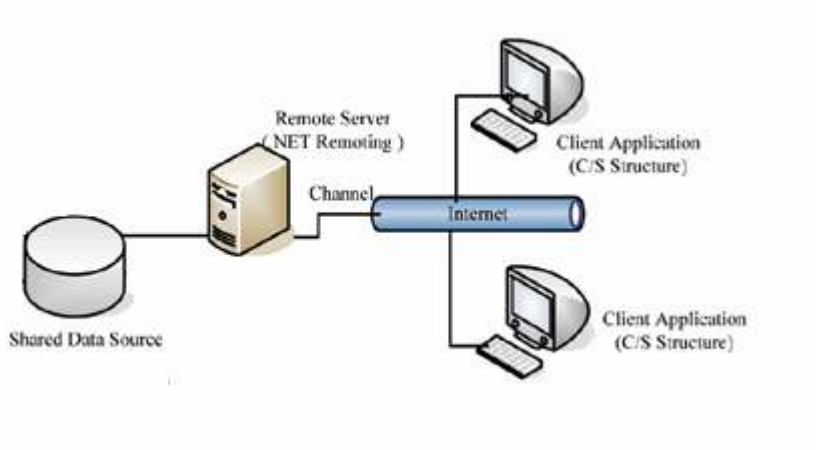
Không là một kiến trúc "siêu việt".

4.2. Remoting

4.2.1. Giới thiệu về Remoting

.NET Remoting là gì?

- Trước hết .NET Remoting là một kĩ thuật .NET được giới thiệu từ .NET framework 1.1. Cùng với .NET Webservice, .NET remoting là lựa chọn cho giải pháp xử lý tính toán từ xa. .NET Remoting là một kĩ thuật cho phép một đối tượng này truy xuất đến một đối tượng khác nằm ở các Application Domain khác nhau. Và nếu giải thích theo kiểu bình dân, ta có thể sử dụng .NET Remoting đế gọi một chương trình hoặc một service chạy trên một máy vi tính khác để xử lý một cái gì đó và trả kết quả tính toán lại cho ta.



Hình 4.1: *.NET Remoting Overview*

.NET Remoting và Distributed COM

- Vào năm một ngàn chín trăm hồi đó, người ta thường thực hiện việc giao tiếp giữa các process bằng cách sử dụng Distributed COM hay còn gọi là DCOM. DCOM đã rất hữu ích cho những chương trình chạy trên các máy tính cùng loại và nằm trong cùng một mạng. Tuy nhiên, DCOM trở nên lỗi thời vì nó không thể chạy trên Internet. DCOM dựa trên một tập giao thức mà không phải object nào cũng hỗ trợ và điều này khiến DCOM không chạy được trên những platform khác nhau. Ngoài ra, DCOM sử dụng nhiều port trong khi các port ấy thường bị chặn bởi firewall. Tất nhiên mở những port đó để nó hoạt động được không khó nhưng đó là một trong những phiền phức.
- .NET Remoting khắc phục những yếu kém của DCOM bằng cách hỗ trợ nhiều giao thức khác nhau.

.NET Remoting và Web Services

- Về khía cạnh xử lý từ xa thì Web Services hoàn toàn tương tự như .NET Remoting. Thậm chí người ta có thể làm cho .NET Remoting trở thành 1 Web Services bằng cách

host nó trong IIS. Web Services cho phép các ứng dụng có thể giao tiếp với nhau mà không phụ thuộc platform, ngôn ngữ lập trình, … Tuy nhiên Web Services là một môi trường "stateless", có nghĩa là nó không lưu lại bất kì trạng thái gì của lần gọi trước và nó cũng không biết gì về phía client đang thực hiện request. Client và server Web Services chỉ có thể trao đổi với nhau bằng các thông điệp SOAP. Những điều sau đây là các điểm khác nhau chính giữa .NET Remoting và Web Serices, chúng cũng là những nhân tố để ta chọn lựa giữa 2 công nghệ này:

• ASP.NET Web Services chỉ có thể được truy xuất qua HTTP còn .NET Remoting có thể được dùng trên nhiều giao thức khác nhau như TCP, HTTP.

• Web Services là một môi trường stateless. Khi có một request từ phía client, sẽ có một object mới được tạo ra để thực hiện request đó trên server. Còn .NET Remoting lại hỗ trợ nhiều lựa chọn state management và có thể thực hiện nhiều request từ một client, đồng thời có hỗ trợ callbacks.

• Web Services serialize các đối tượng thành XML bên trong SOAP message và vì thế có thể truyền tải thông tin của bất cứ thành phần nào miễn có thể chuyển thành XML. Còn đối với .NET Remoting thì tùy giao thức và định dạng message mà nó có thể truyền đi thông tin như thế nào. Ngoài ra theo như giới thiệu thì .NET Remoting có cho phép đối tượng được truyền vào theo cả kiểu tham chiếu(reference) và tham trị (value)

• Web services có thể hoạt động trên các platform môi trường khác nhau trong khi .NET Remoting yêu cầu phía clients phải là .NET application.

Channels

- Trong kĩ thuật .NET Remoting thì Channel được hiểu như là một kênh để giao tiếp giữa client và server. Một object từ client sẽ thông qua Channel để giao tiếp với object phía server, Channel sẽ truyền tải những message từ hai phía. Như giới thiệu phía trên thì có hai channel chính là TcpChannel và HttpChannel tương ứng với các giao thức TCP và HTTP. Ngoài ra, TcpChannel và HttpChannel đều có khả năng extend thành những Custom Channel của bạn.

Làm sao để tạo một Object có thể Remote được trong .NET Remoting?

- Một Object remote được chỉ là một object thông thường nhưng phải được inherit từ MarshalByRefObject. Đoạn code sample ở hình 4.2 là một ví dụ đơn giản về Remotable Object. Đối tượng SampleObject trong hình có một số method đơn giản trả về phép tính tổng, hiệu, tích, thương của hai số nguyên. Giá trị trả về của hàm là kiểu số nguyên, kiểu built-in của .NET framework. Nếu bạn muốn trả về kiểu dữ liệu bạn tự định nghĩa, hoặc một instance của class bạn định nghĩa thì lớp đó của bạn phải được khai báo với attribute Serializable.

```
using System;
public class SampleObject: MarshalByRefObject
{
  public int Add(int a, int b)
  {
   int c = a + b;
```

```
    return c;
  }
  public int Subtract(int a, int b)
  {
    int c = a - b;
    return c;
  }
  public int Multiply(int a, int b)
  {
    int c = a * b;
    return c;
  }
  public int Divide(int a, int b)
  {
    int c;
    if (b != 0)
      c = a / b;
    else
      c = 0;
    return c;
  }
}
```

Hình 4.2: *Remotable Object Sample*

Tạo chương trình Server để host Remotable Object

- Kế tiếp, chúng ta cần tạo ra một chương trình server để lắng nghe những request từ phía client. Trong ví dụ này chúng ta sẽ sử dụng TCP/IP channel. Đầu tiên chúng ta tạo một instance channel và đăng kí một port tương ứng cho nó. Khi có một Request từ phía client, server sẽ nhận request đó và Remote Object của chúng ta sẽ thực thi Request này. Trong .NET Remoting, có hai cơ chế để tạo instance của Remote Object rồi từ đó thực thi request: Singleton và Singlecall. Tùy vào mục đích sử dụng, nhu cầu của chương trình mà server của bạn có thể khai báo theo cơ chế WellKnownObjectMode.SingleCall, hay WellKnownObjectMode.Singleton. Khi khai báo Singleton, Remote Object sẽ được sinh ra, thực thi request, reply lại phía client và sau đó, object này vẫn được lưu lại chứ không bị hủy đi. Đến khi nào process chạy chương trình server kết thúc thì instance này mới bị trình hốt rác Garbage Collector hốt đi. Và ngược lại, khi khai báo là SingleCall, Remote Object sẽ được khởi tạo và hủy đi đối với mỗi lần nhận request từ phía client, cơ chế này tương tự như mô hình .NET Web Service truyền thống.
- Nếu bạn muốn sử dụng .NET Remoting trong IIS thì không cần tạo một chương trình server như thế này. Và tất nhiên, IIS chỉ hỗ trợ HttpChannel. Nếu host 1 .NET Remoting bên trong IIS bạn sẽ mặc nhiên sử dụng được cơ chế Authentication của IIS, ngược lại nếu làm một chương trình server để host như trên thì bạn phải cài đặt cơ chế Authentication của riêng mình. Để host một Remote Object bên trong IIS, trước tiên phải tạo 1 Virtual Directory cho application, sau đó đặt đoạn code đăng kí service bên trong event Application_Start (file global.asax)

- Trong ví dụ này, chúng ta sẽ không sử dụng IIS mà sẽ tạo một console application. Có nhiều lựa chọn khi không sử dụng IIS, ta có thể sử dụng console application, Win form application nhưng trong thực tế, người ta sẽ sử dụng một Windows Service để làm. Còn Console application hay Winform Application thường chỉ dùng để minh họa. Trong ví dụ này, chúng ta sẽ sử dụng port 9999 cho may mắn. Có thể một chương trình nào đó trong máy của bạn đã sử dụng port này, nếu bị như vậy bạn phải chọn port khác. Và sau cùng, để kiểm tra xem máy bạn đang lắng nghe trên những port nào (port nào đã bị sử dụng) thì ta dùng lệnh "netstat –a" trong command prompt.
- Còn bây giờ, hãy xem một console application project với 1 class tên là SampleSerrver. Trong project này tôi đã thêm reference tới System.Runtime.Remoting vào trong project để nó có thể chạy được.

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
public class  Server
{
 public static int Main()
 {
  TcpChannel chan = new TcpChannel(9999);
  ChannelServices.RegisterChannel(chan, false);
  RemotingConfiguration.RegisterWellKnownServiceType(typeof(SampleObject)
        , "SampleNetRemoting", WellKnownObjectMode.SingleCall);
  Console.WriteLine("Hit <enter> to exit...");
  Console.ReadLine();
  }
}
```

Hình 4. 3: *Sample Server host Remotable Object*

Tạo chương trình client để sử dụng Remote Object.

- Chương trình client trong ví dụ này cũng khá đơn giản, nó sẽ connect vào server, tạo một instance của Remote Object và excute method tính tổng, hiệu, tích, thương.
- Các bạn lưu ý rằng trong cả chương trình client và chương trình server đều phải reference tới class SampleObject. Client sẽ gọi method của instance SampleObject, nhưng server sẽ thực thi xử lý nó chứ không phải phía client.

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Http;
public class Client
{
```

```
  public static int Main (string[] argv)
  {
   TcpChannel chan = new TcpChannel();
   ChannelServices.RegisterChannel(chan, false);
   SampleObject obj = (SampleObject)Activator.GetObject(
     typeof(SampleObject), "tcp://localhost:9999/SampleNetRemoting");
   if (obj == null)
     System.Console.WriteLine("Could not locate server");
   else
   {
     int a = Convert.ToInt32(argv[0]);
     int b = Convert.ToInt32(argv[1]);
     int c = obj.Add(a, b);
     Console.WriteLine("a + b = {0}", c);
     c = obj.Subtract(a, b);
     Console.WriteLine("a - b = {0}", c);
     c = obj.Multiply(a, b);
     Console.WriteLine("a * b = {0}", c);
     c = obj.Divide(a, b);
     Console.WriteLine("a / b = {0}", c);
   }
     Console.ReadKey();
  }
}
```

Hình 4: *Sample Client Application*

Test thử chương trình

- Trước tiên chạy chương trình server, bạn sẽ thấy message "Press the enter key to exit" trong cửa sổ console. Như vậy server của bạn đang lắng nghe trên port 9999. Bây giờ bạn hãy chạy chương trình client và sẽ nhìn thấy kết quả trả về trên màn hình. Bạn có thể chạy nhiều client để cùng request đến 1 server nhưng không thể chạy nhiều server. Bạn có thể copy chương trình server sang một máy của bạn mình và nhờ chạy thử, còn bạn sửa lại chương trình client, sửa "localhost" thành IP của máy bạn mình và chạy thử để thấy kết quả.

Tóm tắt:

- Ví dụ ở trên đã sử dụng code C# để khai báo các cấu hình cho server và client tuy nhiên .NET Remoting cho phép ta cấu hình trước trong file config (App.config). Các bạn có thể tham khảo một số resource phía dưới để biết cách làm.
- .NET Remoting là một trong những kĩ thuật tiện lợi cho những chương trình dạng Distributed Computing. Cách sử dụng nó phức tạp hơn Web Service tuy nhiên nếu bạn

muốn tăng performance thì .NET Remoting với Singleton và TCP channel sẽ là lựa chọn rất tốt.

- Với sự ra đời của .NET Framework 3.x, Microsoft đã giới thiệu nền tảng mới hơn cho các kĩ thuật RPC, đó là WCF mạnh hơn .NET Remoting rất nhiều.

4.2.2. Khai báo, cài đặt và đăng ký giao diện từ xa

Để cho chương trình có tính khả chuyển cao thay vì người ta xây dựng lớp Remote Object như ví dụ trên chúng ta khai báo một giao diện là lớp Remote Object và trong chương trình phía Server ta sẽ cài đặt giao diện này và đăng ký giao diện từ xa. Như vậy để triển khai một hệ thống Remoting ta có 3 chương trình: Giao diện Remote Object, chương trình Server triển khai giao diện và đăng ký giao diện từ xa, chương trình Client triệu gọi phương thức từ xa.

- Khai báo giao diện từ xa
- Cài đặt và đăng ký giao diện từ xa

4.2.3. Triệu gọi phương thức từ xa

- Chương trình phía Client chúng ta triệu gọi phương thức được cung cấp bởi giao diện tử xa đã được đăng ký và cung cấp bởi Server

4.3. Web Services

4.3.1. Giới thiệu về Web Services

1. Web Service là gì?

Web service là một Modul chương trình cung cấp chức năng của các ứng dụng cho phép triệu gọi và truy cập từ xa thông qua Internet. Web service sử dụng các chuẩn của Internet như XML và HTTP. Việc sử dụng Web service phụ thuộc nhiều vào sự chấp nhận của XML, một ngôn ngữ mô tả dữ liệu mới dùng để truyền tải dữ liệu thông qua Web.

Bất kỳ một Web service nào cũng có thể được sử dụng, hoặc là trong ứng dụng cục bộ hoặc truy cập từ xa qua Internet bởi nhiều ứng dụng. Do có khả năng truy cập qua các giao diện chuẩn mà một Web service cho phép nhiều hệ thống khác nhau cùng làm việc với nhau như một tiến trình duy nhất trên Web.

2. Vai trò của Web service

Web service ra đời đã mở ra một hướng mới cho việc phát triển các ứng dụng trên Internet. Web services tạm dịch là các dịch vụ trên web. Công nghệ web services ra đời là một cuộc cách mạng hóa cách thức hoạt động của các dịch vụ B2B và B2C. Web services kết hợp sử dụng nhiều công nghệ khác nhau cho phép hai ứng dụng cùng ngôn ngữ, độc lập hệ điều hành trao đổi được với nhau thông qua môi trường mạng Internet. Tuy nhiên những công nghệ sử dụng ở đây không nhất thiết phải là những công nghệ mới. Đây là điểm khác biệt của web services so với các công nghệ khác, đó chính là khả năng kết hợp các công nghệ đã có như là XML, SOAP, WSDL, UDDI để tạo ra các service, đặc điểm này làm nổi bật vai trò của web services.

Web Service được thiết kế nhằm cung cấp một cơ chế cho phép các chương trình giao tiếp với nhau qua Internet (sử dụng các giao thức Internet như HTTP GET, HTP POST và SOAP).

3. Đặc điểm Web service

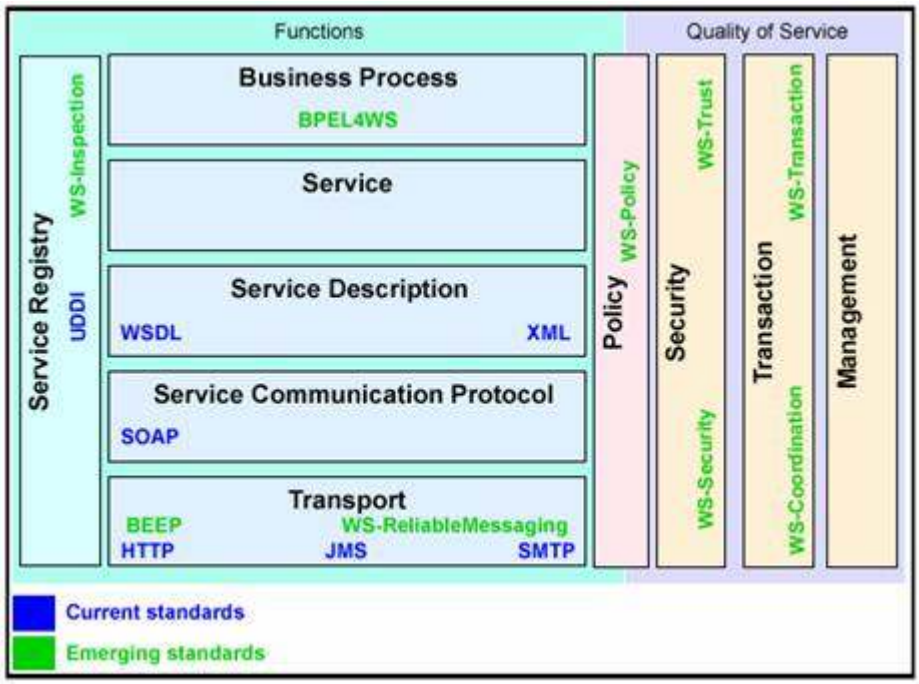
- Web service cho phép client và server tương tác được với nhau mặc dù trong những môi trường khác nhau.
- Web Service thì có dạng mở và dựa vào các tiêu chuẩn XML và HTTP là nền tảng kỹ thuật cho web service. Phần lớn kỹ thuật của web service được xây dựng là những dự án nguồn mở. Bởi vậy chúng độc lập và vận hành được với nhau.
- Web Service thì rất linh động: Vì với UDDI và WSDL, thì việc mô tả và phát triển web service có thể được tự động hoá.
- Web service được xây dựng trên nền tảng những công nghệ đã được chấp nhận.
- Web service có dạng modul.
- Web service có thể được công bố (publish) và gọi thực hiện qua mạng.

Ngày nay web service được sử dụng rất nhiều trong những lĩnh vực khác nhau của cuộc sống như:

- Dịch vụ chọn lọc và phân loại tin tức: là những hệ thống thư viện kết nối đến các web portal để tìm kiếm các thông tin từ các nhà xuất bản có chứa những khoá muốn tìm.
- Dịch vụ hiển thị danh sách đĩa nhạc dành cho các công ty thu thanh.
- Ứng dụng đại lý du lịch có nhiều giá vé đi du lịch khác nhau do có chọn lựa phục vụ của nhiều hãng hàng không.
- Bảng tính toán chính sách bảo hiểm dùng công nghệ Excel/COM với giao diện web.
- Thông tin thương mại bao gồm nhiều nội dung, nhiều mục tin như: dự báo thời tiết, thông tin sức khoẻ, lịch bay, tỷ giá cổ phiếu,…
- Những giao dịch trục tuyến cho cả B2B và B2C như: đặt vé máy bay, làm giao kèo thuê xe.
- Hệ thống thông tin dùng java để tính toán tỷ giá chuyển đổi giữa các loại tiền tệ. Hệ thống này sẽ được các ứng dụng khác dùng như một web service.

4. Kiến trúc Web service

Kiến trúc của Web service bao gồm các tầng như sau:



Hình 1: Kiến trúc Web service

Trong đó bao gồm các tầng như sau:

- Tầng vận chuyển: có nhiệm vụ truyền thông điệp giữa các ứng dụng mạng, bao gồm những giao thức như HTTP, SMTP, FTP, JSM và gần đây nhất là giao thức thay đổi khối mở rộng (Blocks Extensible Exchange Protocol- BEEP).
- Tầng giao thức tương tác dịch vụ (Service Communication Protocol) với công nghệ chuẩn là SOAP. SOAP là giao thức nằm giữa tầng vận chuyển và tầng mô tả thông tin về dịch vụ, SOAP cho phép người dùng triệu gọi một service từ xa thông qua một message XML.
- Tầng mô tả dịch vụ (Service Description) với công nghệ chuẩn là WSDL và XML. WSDL là một ngôn ngữ mô tả giao tiếp và thực thi dựa trên XML. Web service sử dụng ngôn ngữ WSDL để truyền các tham số và các loại dữ liệu cho các thao tác, các chức năng mà web service cung cấp.
- Tầng dịch vụ (Service): cung cấp các chức năng của service.
- Tầng đăng ký dịch vụ (Service Registry) với công nghệ chuẩn là UDDI. UDDI dùng cho cả người dùng và SOAP server, nó cho phép đăng ký dịch vụ để người dùng có thể gọi thực hiện service từ xa qua mạng, hay nói cách khác một service cần phải được đăng ký để cho phép các client có thể gọi thực hiện
- Bên cạnh đó để cho các service có tính an toàn, toàn vẹn và bảo mật thông tin trong kiến trúc web service chúng ta có thêm các tầng Policy, Security, Transaction, Management giúp tăng cường tính bảo mật, an toàn và toàn vẹn thông tin khi sử dụng service.

4.3.2. Giao thức SOAP

SOAP là chữ viết tắt của cụm từ "Simple Object Access Protocol – Giao thức truy cập đối tượng đơn giản", nhưng với sự xem xét mới nhất thì, SOAP sẽ không còn là một từ viết tắt nữa. Chuẩn SOAP ghi nhận XML được thể hiện thế nào bên trong tài liệu SOAP, làm thế nào nội dung của thông điệp được truyền tải, và làm thế nào thông điệp được xử lý ở cả hai phía gởi và nhận. SOAP cũng cung cấp một tập các từ vựng chuẩn.

Các thuật ngữ:

Như bất kỳ công nghệ nào, SOAP cũng có tập các thuật ngữ của riêng nó. Có nhiều thuật ngữ được sử dụng thường xuyên để mô tả các khía cạnh khác nhau của chuẩn SOAP. Nhiều lập trình viên dùng các thuật ngữ này mà không thật sự hiểu ý nghĩa của nó. Để hiểu thật sự các khái niệm đòi hỏi phải tốn một thời gian để hiểu ý nghĩa của từng thuật ngữ và làm thế nào để áp dụng cho cả chuẩn SOAP và một Web Services thực thụ

Chú ý: Chuẩn SOAP không chỉ là chuẩn XML mà chuẩn này còn bao gồm các thông điệp SOAP có hành vi như thế nào, các phương tiện vận chuyển khác nhau, cách mà các lỗi được xử lý..

Sự truyền tải dữ liệu

SOAP Binding

Thuật ngữ mô tả làm thế nào một thông điệp SOAP tương tác được với một giao thức vận chuyển như HTTP, SMTP hay FTP để di chuyển trên Internet. Điều quan trọng là SOAP di chuyển bằng một giao thức chuẩn để liên lạc với các sản phẩm Web Services khác.

Trước SOAP, nhiều người phát triển đã tạo ra các phương pháp của riêng họ để chuyển tải một tài liệu XML trên mạng. Các cách này vẫn hoạt động tốt trong phạm vi một nhóm cụ thể. Tuy nhiên khi bạn cần làm việc với một nhóm khác ở trong hay bên ngoài công ty thì điều này trở nên khó khăn vì phải huấn luyện và có thể thay đổi để làm việc với việc truyền tải tài liệu XML mà họ đang sử dụng. Bằng cách sử dụng một tài liệu XML chuẩn trên các giao thức chuẩn, công việc cần làm khi cộng tác với nhau sẽ được giảm thiểu tối đa

SOAP Message Exchang Pattern (MEP)

Thuật ngữ mô tả làm thế nào mà một tài liệu SOAP trao đổi giữa phía máy khách và chủ. Thông điệp SOAP sở hữu một liên kết, như là HTTP, để nó có thể truyền trên Internet. Việc nói chuyện giữa máy khách và chủ, ở đây là các nút, xác định các hành động mà cả hai phía thực hiện.

Nhắc lại SOAP là một XML đóng gói RPC. Vì thế, MEP hoàn toàn là yêu cầu và phản hồi giữa máy khách và chủ (hay các nút khác). Như vậy nếu có nhu cầu liên lạc giữa các nút thì việc này được thực hiện bằng nhiều yêu cầu và phản hồi để hoàn tất việc truyền thông điệp. Cách này khác hẳn với các công nghệ đối tượng từ xa khác như CORBA, công nghệ đó được thực hiện chỉ trong một kết nối.

SOAP Application

Một ứng dụng SOAP đơn giản là một ứng dụng dùng SOAP theo một vài cách khác nhau. Vài ứng dụng hoàn toàn dựa trên chuẩn SOAP, như là Web Services cổ phiếu, hoặc dùng chuẩn SOAP để nhận mã và các cập nhật của phần mềm. Chú ý là một ứng dụng có thể tạo, sử dụng hoặc là nút trung gian của Web Services.

SOAP Node

Trách nhiệm của một nút có thể bao gồm gởi, nhận, xử lý hoặc truyền tải lại một thông điệp SOAP. Một nút chỉ là một phần nhỏ của phần mềm, xử lý một tài liệu SOAP phụ thuộc vào

vai trò của nó. Bên cạnh việc truyền dữ liệu, một nút có trách nhiệm đảm bảo thông tin XML trong tài liệu SOAP phải đúng ngữ pháp theo chuẩn SOAP.

SOAP Role

Một vai trò của SOAP định nghĩa một nút cụ thể hoạt động như thế nào. Nó có thể là nút gởi, nhận hoặc nút trung gian.

SOAP Sender

Nút gởi là nút gởi yêu cầu SOAP. Nếu bạn nghĩ đến ví dụ của ứng dụng khách chủ thì khi ứng dụng khách thực hiện yêu cầu, nó gởi thông điệp tới ứng dụng chủ để yêu cầu vài thông tin.

SOAP Receiver

Ngược lại với SOAP sender là nút nhận.

SOAP Intermediary

Một nút trung gian có thể xem một thông điệp SOAP và tương tác trên vài phần thông tin của thông điệp, và chuyển đến vị trí kế tiếp của thông điệp. Một nút trung gian thường hoạt động như một router. Một router sẽ xem xét thông tin của gói tin chuyển trên mạng, tìm điểm kế tiếp của gói tin và chuyển gói tin đến đó.

Message Path

Một thông điệp SOAP di chuyển từ phía bên gởi đến phía bên nhận thông điệp thông qua nhiều nút trung gian. Tuyến đường đi của thông điệp được gọi là một Message Path.

Initial SOAP Sender

Nút gởi yêu cầu SOAP đầu tiên là nút gởi SOAP ban đầu.

SOAP Feature

Một đặc điểm SOAP là một phần chức năng của phần mềm hỗ trợ chức năng SOAP.

Các thuật ngữ liên quan đến XML

Chuẩn SOAP cũng định nghĩa một tập nhỏ các phần tử XML để đóng gói dữ liệu được truyền giữa các nút. Thật sự chỉ có vài phần tử vì phần thân của thông điệp có thể khác nhau phụ thuộc vào cài đặt. Sự uyển chuyển này được cho phép bởi chuẩn SOAP.

SOAP Message

Đây là tài liệu XML được truyền bởi một nút SOAP gởi hoặc nhận. Một nút gởi hoặc nút khách tạo ra một tài liệu XML chứa thông tin mà phía bên khách cần từ phía chủ. Một khi tài liệu được truyền, phía bên chủ phân giải thông tin trong tài liệu để truy xuất các giá trị khác nhau và tạo một thông điệp SOAP mới để phản hồi.

SOAP Envelope

Đây là phần tử gốc của tài liệu SOAP XML. Tài liệu SOAP chứa nhiều định nghĩa không gian tên (namespace) nhưng các phần tử liên quan tới thông điệp SOAP sẽ có ENV: là tiếp đầu ngữ.

SOAP Header

Phần đầu của một thông điệp SOAP chứa một khối thông tin đầu trong tài liệu XML để định tuyến và xử lý thông điệp SOAP. Dữ liệu này tách rời khỏi phần thân của tài liệu có chứa thông tin liên quan đến đối tượng được gọi.

SOAP Header Block

Phần đầu của SOAP chứa nhiều phần giới hạn hay là nhiều khối thông tin có một khối thông tin của phần đầu. Những khối thông tin của phần đầu này chứa thông tin về các nút trung gian vì một nút cần biết nút kế tiếp để thông điệp được gởi đến.

SOAP Body

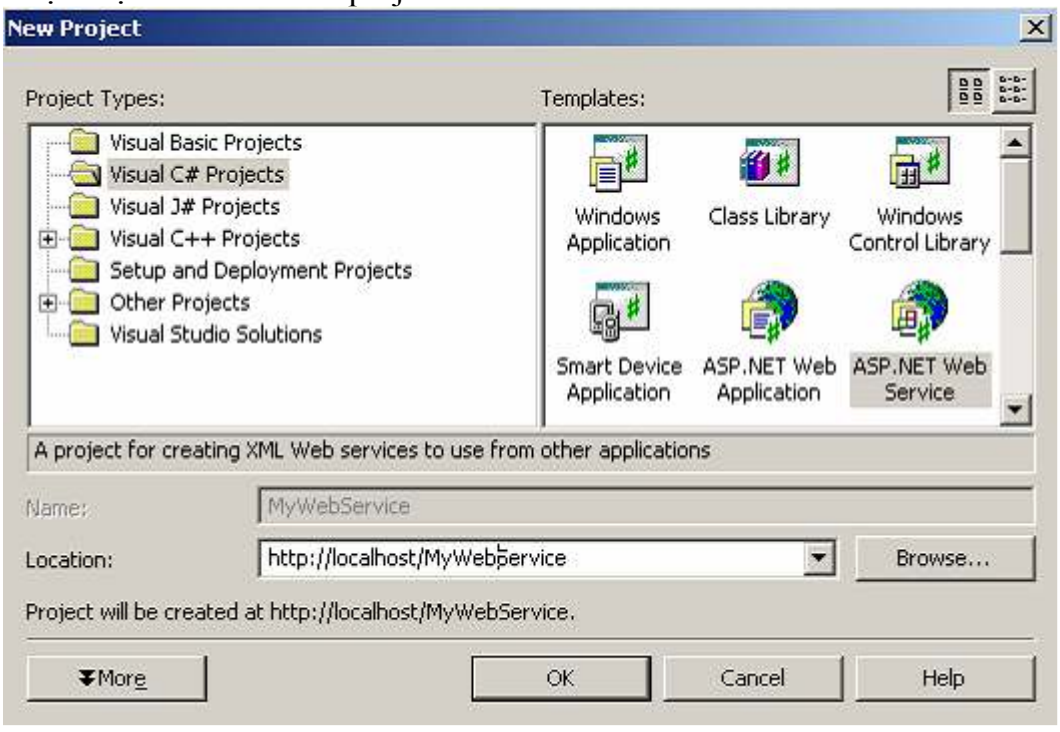
Phần thân của SOAP thật sự chứa thông tin của đối tượng để xử lý thông tin. Phần thân sau khi được phân tách sẽ trở thành đối tượng. Đối tượng xử lý thông tin và kết quả được đặt trong phần thân của tài liệu trả về.

SOAP Fault

Đây là một phần thông tin của SOAP chứa thông tin đến bất kỳ lỗi gì xảy ra tại một nút SOAP.

4.3.3. Xây dựng Web Services

- Tạo một Web Services project

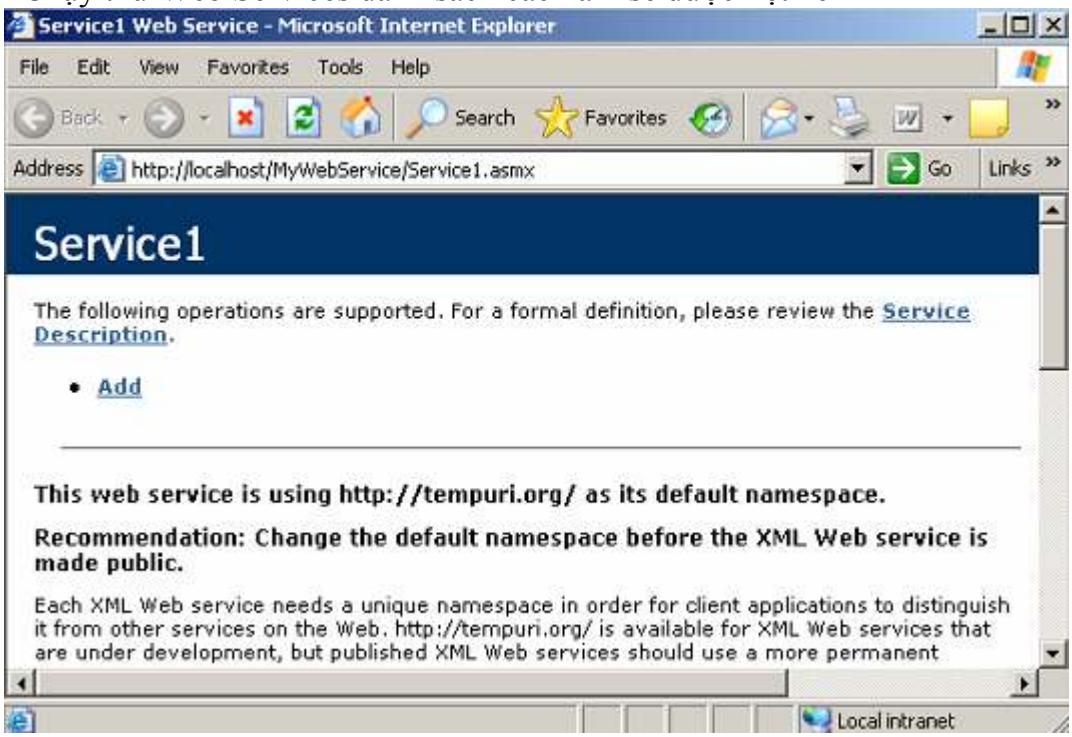


- Tạo Web Method

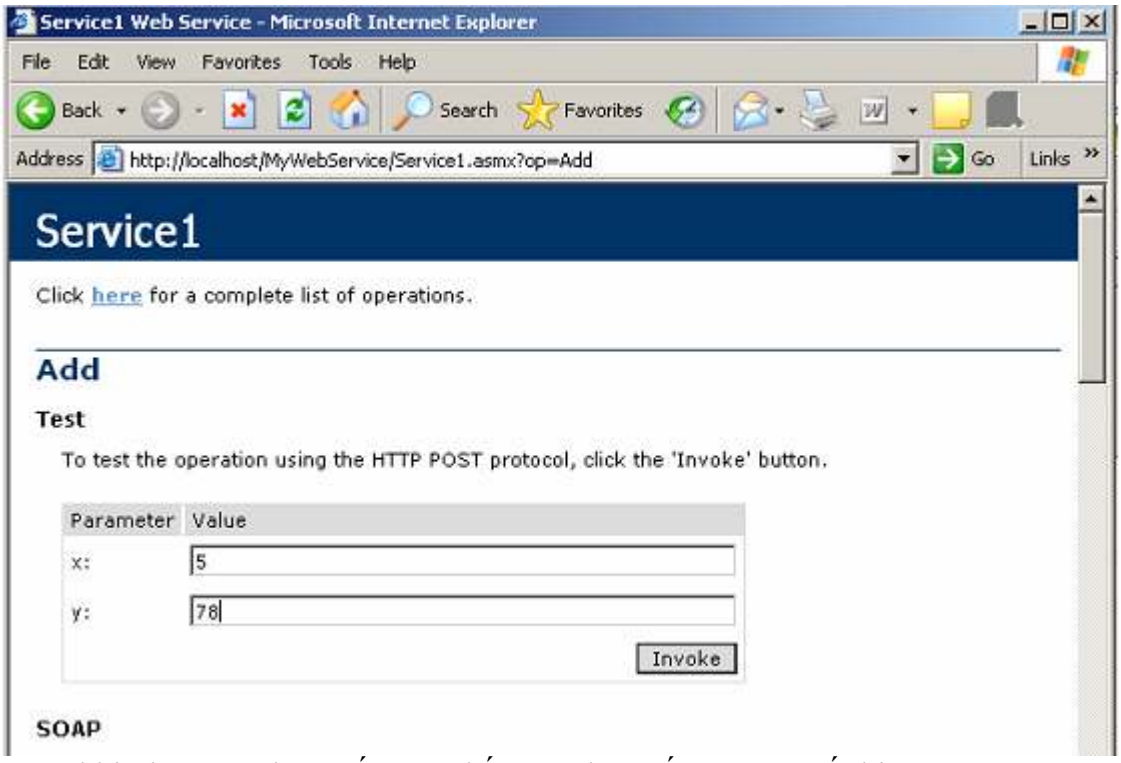
```
MyWebService.Service1                                   Service1()
namespace MyWebService
{
    /// <summary>
    /// Summary description for Service1.
    /// </summary>
    public class Service1 : System.Web.Services.WebService
    {
        public Service1()
        {
            //CODEGEN: This call is required by the ASP.NET Web Services Designer
            InitializeComponent();
        }
        Component Designer generated code

        [WebMethod]
        public int Add(int x, int y)
        {
            return x+y;
        }
    }
}
```

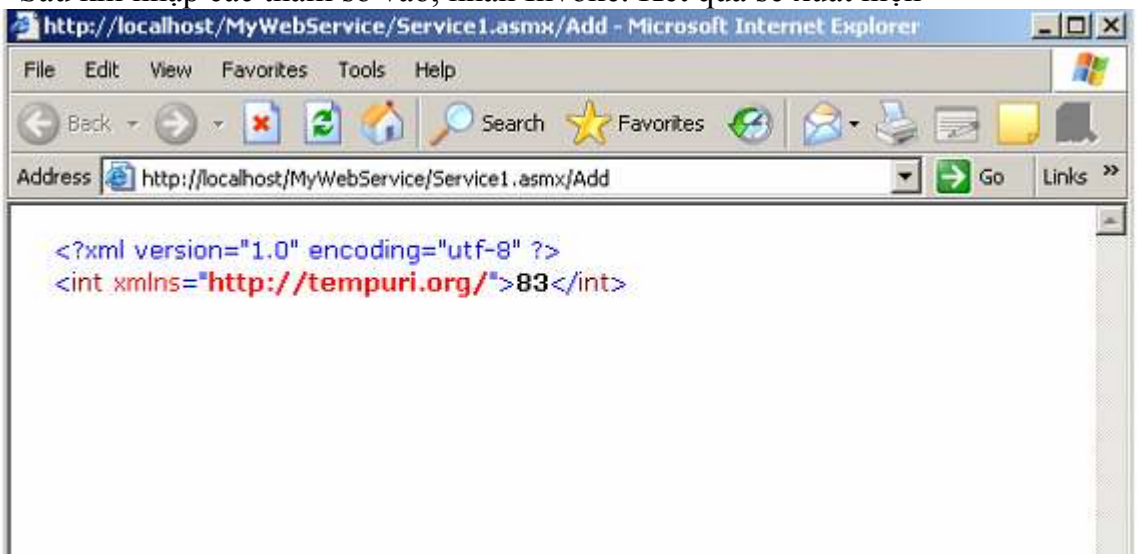
- Chạy thử Web Services danh sách các hàm sẽ được liệt kê



- Chọn hàm Add



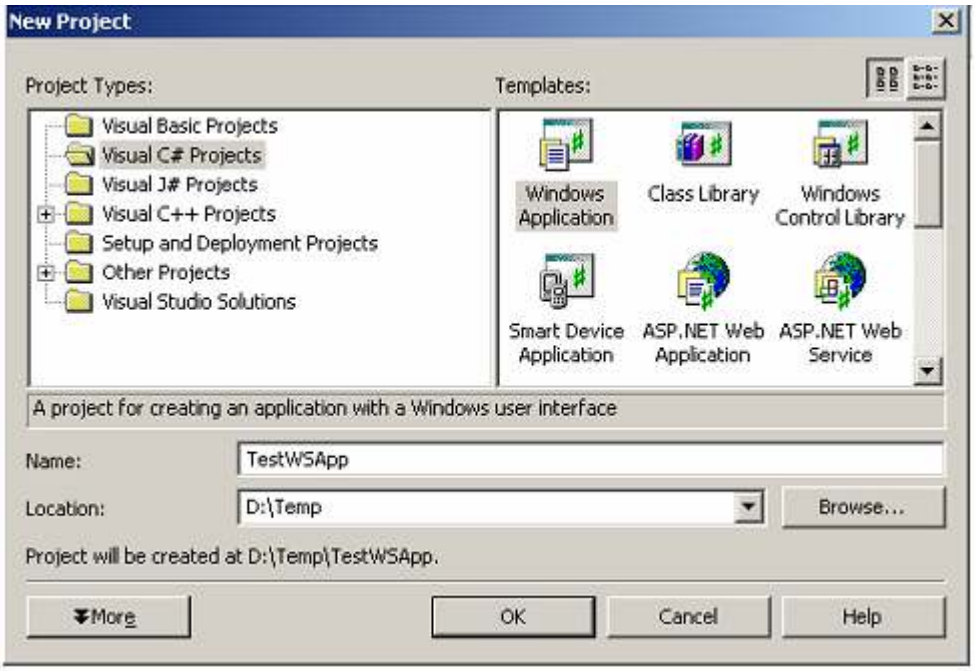
- Sau khi nhập các tham số vào, nhấn Invoke. Kết quả sẽ xuất hiện



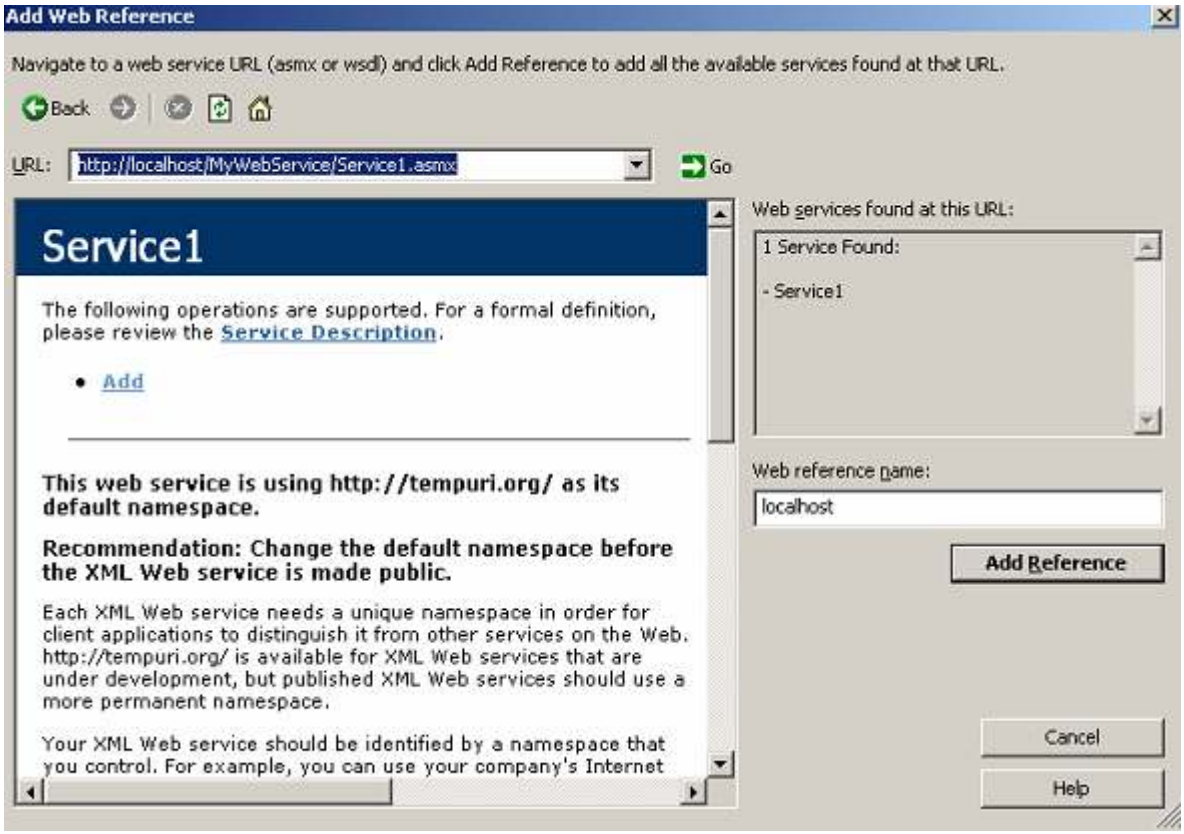
4.3.4. Triệu gọi Web Services từ ứng dụng .NET, Java và các ngôn ngữ khác

- Sau khi xây dựng Web Server song ta có thể triệu gọi nó từ một ứng dụng khác

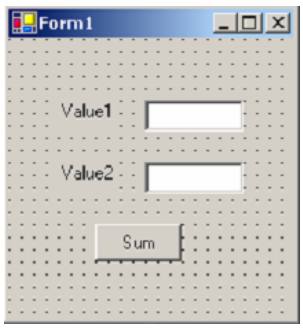
- Tạo một Window Form



- Add Web Reference



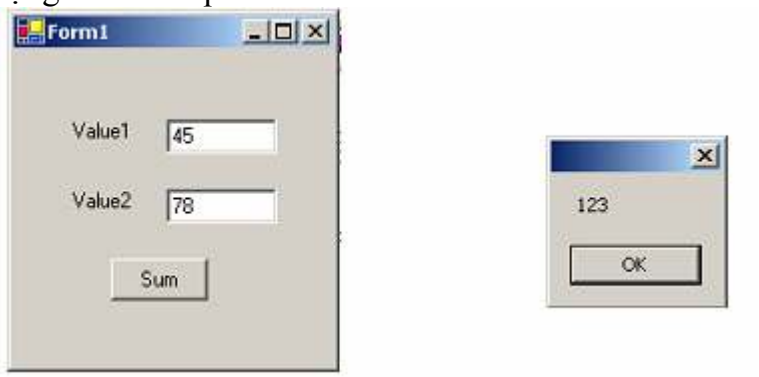
- Tạo màn hình:



- Viết hàm xử lý nút nhấn:

```
private void btnSum_Click(object sender, System.EventArgs e)
{
    localhost.Service1 ws = new localhost.Service1();
    int sum;
    sum = ws.Add(Convert.ToInt32(txtValue1.Text), Convert.ToInt32(txtValue2.Text));
    MessageBox.Show(sum.ToString());
}
```

- Chạy thử ứng dụng ta có kết quả:



4.4 Thảo luận về các ứng dụng phân tán

4.5. Bài tập áp dụng

1. Viết chương trình Chat sử dụng công nghệ Web Services
2. Viết chương trình Calculator bằng công nghệ Web Services
3. Viết chường trình quản lý FileManager bằng công nghệ Web Services.

TÀI LIỆU THAM KHẢO

1. Richard Blum, C# Network Programming, 2003
2. Fiach Reid, Network programming in NET with C# and VB.NET, Digital Press, 2003
3. Bài giảng "Nhập môn Công nghệ phần mềm", Đại học KHTN
4. Bài giảng "Xây dựng phần mềm hướng đối tượng", Đại học KHTN
5. Bài giảng "Lập trình truyền thông", Đại học Cần Thơ
6. Bài giảng "Công nghệ .NET", Khoa CNTT – Đại học SPKT Hưng Yên
7. Bài giảng "Java Nâng cao", Khoa CNTT- Đại học SPKT Hưng Yên
8. Các ví dụ tại Website: www.java2s.com