Introduction

We can now use ASP.NET to create Web Services based on industrial standards including XML, SOAP and WSDL.

A Web Service is a software program that uses XML to exchange information with other software via common internet protocols. In a simple sense, Web Services are a way for interacting with objects over the Internet.

A web service is

- Language Independent**.**
- Protocol Independent**.**
- Platform Independent**.**
- It assumes a stateless service architecture.
- Scalable (e.g. multiplying two numbers together to an entire customer-relationship management system).
- Programmable (encapsulates a task).
- Based on XML (open, text-based standard).
- Self-describing (metadata for access and use).
- Discoverable (search and locate in registries)- ability of applications and developers to search for and locate desired Web services through registries. This is based on UDDI.

Key Web Service Technologies

- **XML**- Describes only data. So, any application that understands XML-regardless of the application's programming language or platform-has the ability to format XML in a variety of ways (well-formed or valid).
- **SOAP**- Provides a communication mechanism between services and applications.
- **WSDL**- Offers a uniform method of describing web services to other programs.
- **UDDI**- Enables the creation of searchable Web services registries.

When these technologies are deployed together, they allow developers to package applications as services and publish those services on a network.

Web services advantages

- Use open, text-based standards, which enable components written in various languages and for different platforms to communicate.
- Promote a modular approach to programming, so multiple organizations can communicate with the same Web service.
- Comparatively easy and inexpensive to implement, because they employ an existing infrastructure and because most applications can be repackaged as Web services.
- Significantly reduce the costs of enterprise application (EAI) integration and B2B communications.
- Implemented incrementally, rather than all at once which lessens the cost and reduces the organizational disruption from an abrupt switch in technologies.
- The Web Services Interoperability Organization (WS-I) consisting of over 100 vendors promotes interoperability.

Web Services Limitations

- SOAP, WSDL, UDDI- require further development.
- Interoperability.
- Royalty fees.
- Too slow for use in high-performance situations.
- Increase traffic on networks.
- The lack of security standards for Web services.
- The standard procedure for describing the quality (i.e. levels of performance, reliability, security etc.) of particular Web services â€“ management of Web services.
- The standards that drive Web services are still in draft form (always will be in refinement).

- Some vendors want to retain their intellectual property rights to certain Web services standards.

Web Service Example

A web service can perform almost any kind of task.

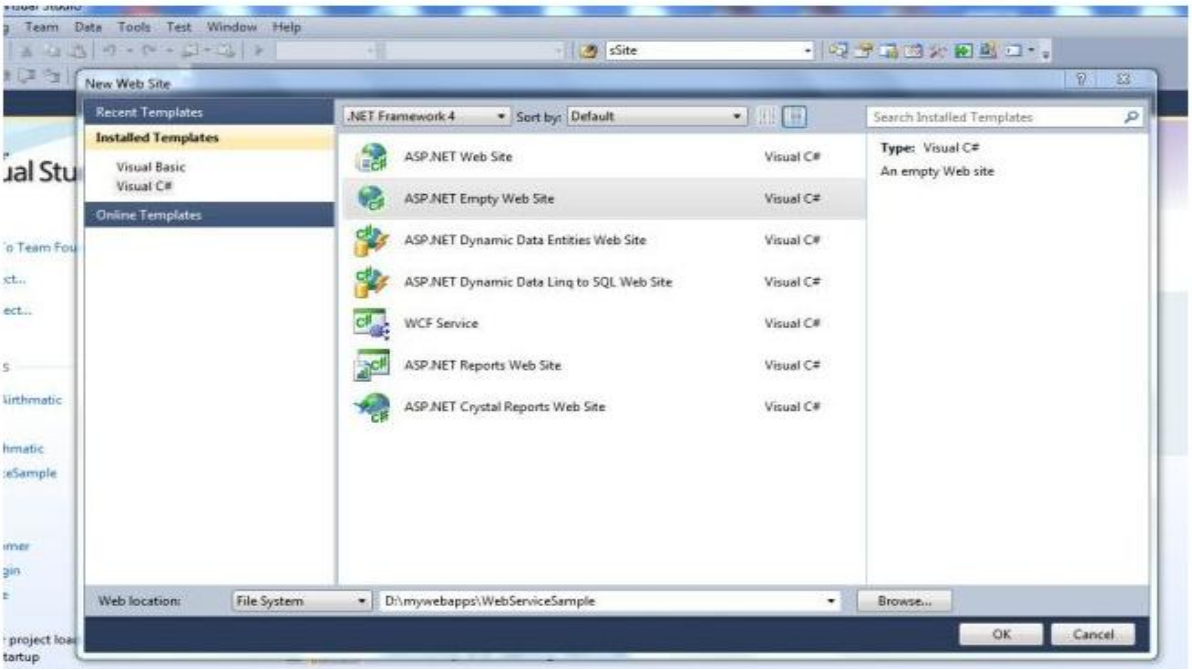
- **Web Portal-** A web portal might obtain top news headlines from an Associated press web service.
- **Weather Reporting-** You can use Weather Reporting web service to display weather information in your personal website.
- **Stock Quote-** You can display latest update of Share market with Stock Quote on your web site.
- **News Headline:** You can display latest news update by using News Headline Web Service in your website.
- You can make your own web service and let others use it. For example you can make Free SMS Sending Service with footer with your companies advertisement, so whosoever uses this service indirectly advertises your company. You can apply your ideas in N no. of ways to take advantage of it.

How to create a Web Service

Step 1

Go to Visual Studio then click on "File" -> "Website" -> "ASP.NET empty website template".

Then provide the website name (for example: WebServiceSample).

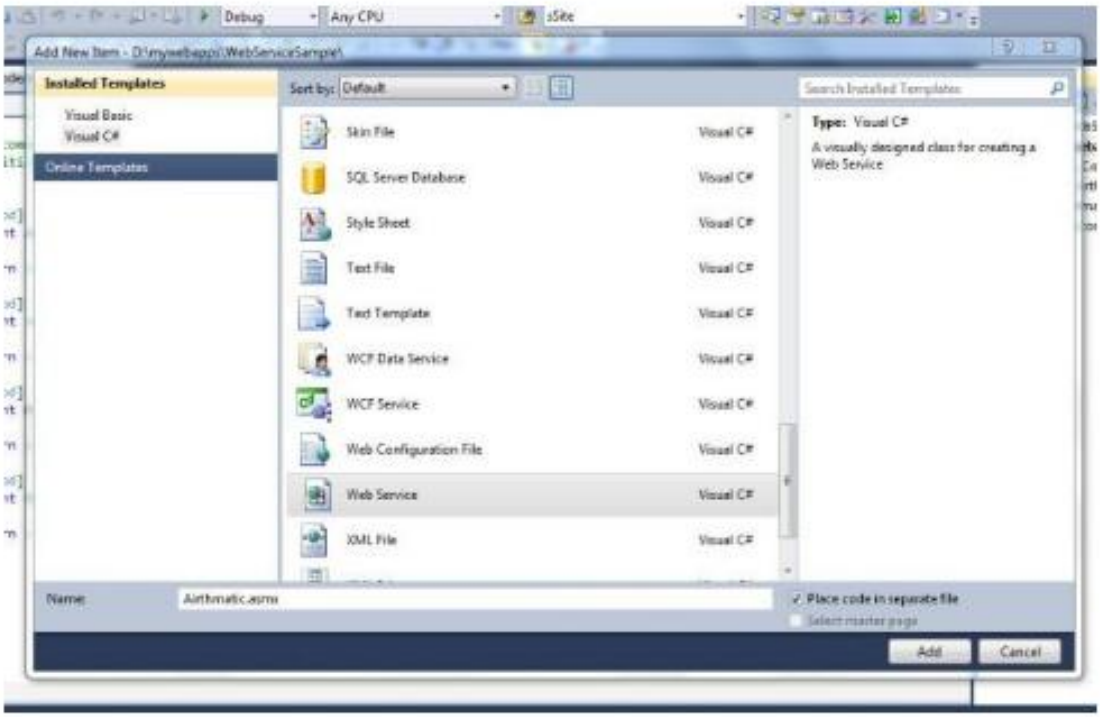


Step 2 Add a Web Service File

Go to Solution Explorer, then select the solution then click on "Add new item".

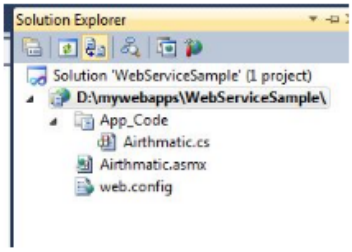
Choose the Web Service template.

Enter the name (for example: Airthmatic.cs) then click on "Add".



This will create the following two files:

1. Airthmatic.asmx (the service file)
2. Airthmatic.cs (the code file for the service; it will be in the "App_code" folder)



Open the file Airthmatic.cs and write the following code

```
using System;
 using System.Collections.Generic;
 using System.Linq;
 using System.Web;
 using System.Web.Services;
 /// <summary>
 /// used for Airthmatic calculation
 /// </summary>
 [WebService(Namespace = "http://tempuri.org/")]
 [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
 // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment
 // [System.Web.Script.Services.ScriptService]
 public class Airthmatic : System.Web.Services.WebService
 {
     public Airthmatic()   {
         //Uncomment the following line if using designed components
         //InitializeComponent();
     }
     [WebMethod]
     public int Add(int x, int y)
     {
         return x + y;
     }
     [WebMethod]
     public int Sub(int x, int y)
     {
         return x - y;
     }
     [WebMethod]
     public int Mul(int x, int y)
     {
         return x * y;
     }
     [WebMethod]
     public int Div(int x, int y)
     {
         return x / y;
     }
 }
```

Attaching the WebMethod attribute to a Public method indicates that you want the method exposed as part of the XML Web service. You can also use the properties of this attribute to further configure the behavior of the XML Web service method. The WebMethod attribute provides the following properties

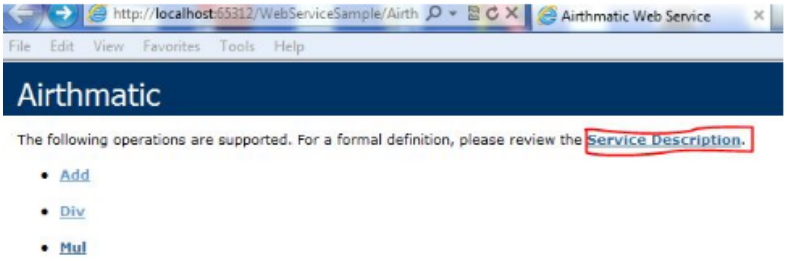
- BufferResponse
- CacheDuration
- Description
- EnableSession
- MessageName
- TransactionOption

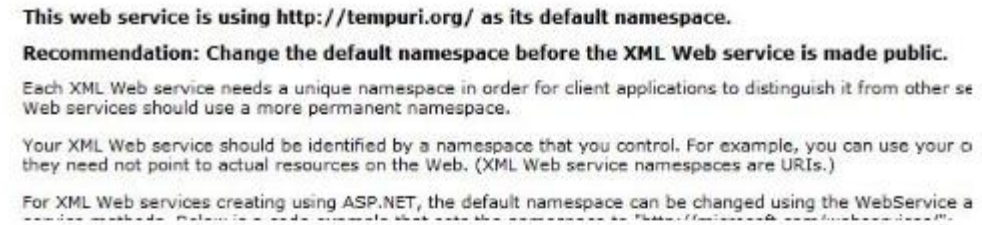
For more details of web methods click here.

Step 3

To see whether the service is running correctly go to the Solution Explorer then open "Airthmatic.asmx" and run your application.

Now you will find all the method names in the browser.





To see the WSDL format click on the service description link or add "?WSDL" to the URL.

Example

http://localhost:65312/WebServiceSample/Airthmatic.asmx?WSDL

It will show the WSDL.

To determine whether the functions are working, click on one of the functions (for example: "Add").

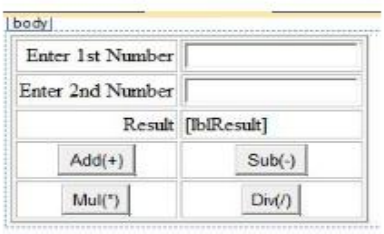
Now you will see two TextBoxes for checking. Enter the value for x and y and click on the "Invoke" button.

Now you will see the result in an open standard form (XML).

Now your service is ready for use.

Step 4 Creating the client application

Now create a website and design your form as in the following screen.



Or you can copy the following source code.

```
<body>
    <form id="form1" runat="server">
    <div>
        <table border="2" cellpadding="2" cellspacing="2">
            <tr>
                <td align="right">
                    <asp:Label ID="Label1" runat="server" Text="Enter 1st Number">
</asp:Label>
                </td>
                <td align="left">
                    <asp:TextBox ID="txtFno" runat="server"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td align="right">
                    <asp:Label ID="Label2" runat="server" Text="Enter 2nd Number">
</asp:Label>
                </td>
                <td align="left">
                    <asp:TextBox ID="txtSno" runat="server"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td align="right">
                    <asp:Label ID="Label3" runat="server" Text="Result">
</asp:Label>
                </td>
                <td align="left">
                    <asp:Label ID="lblResult" runat="server"></asp:Label>
                </td>
            </tr>
            <tr>
                <td align="center">
                    <asp:Button ID="btnAdd" runat="server" Text="Add(+)" OnClick="btn
                </td>
                <td align="center">
                    <asp:Button ID="btnSub" runat="server" Text="Sub(-)" OnClick="btn
                </td>
            </tr>
            <tr>
                <td align="center">
                    <asp:Button ID="BtnMul" runat="server" Text="Mul(*)" OnClick="Btn
                </td>
                <td align="center">
                    <asp:Button ID="btnDiv" runat="server" Text="Div(/)" OnClick="btn
                </td>
            </tr>
        </table>
    </div>
    </form>
</body>
```

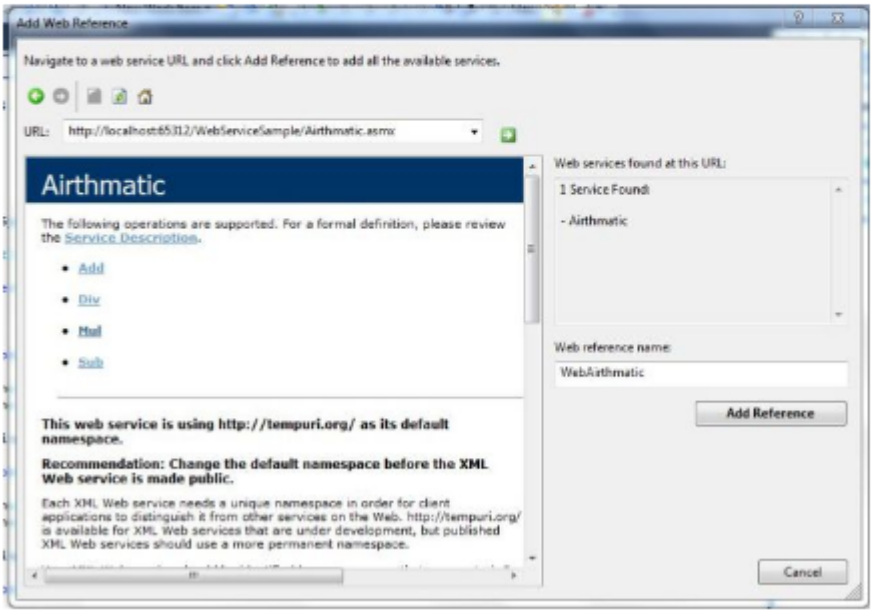
Step 5 Add a web reference to the Website

Go to Solution Explorer then select the solution then click on "AddWeb Reference" then within the URL type the service reference path.

(For example: http://localhost:65312/WebServiceSample/Airthmatic.asmx) then click on the "Go" button.

Now you will see your service methods. Change the web reference name from "localhost" to any other name as you like (for example: WebAirthmatic).

Click on the "Add Reference" button. It will create a Proxy at the client side.



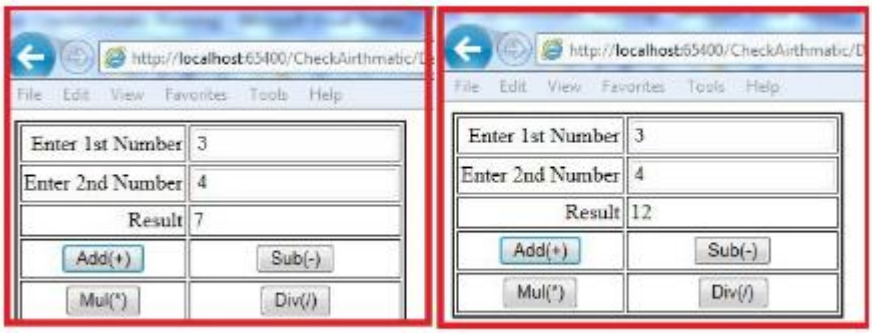
Now go to the cs code and add a reference for the Service.

using WebAirthmatic;

Write the following code.

```
01. using System;
02. using System.Collections.Generic;
03. using System.Linq;
04. using System.Web;
05. using System.Web.UI;
06. using System.Web.UI.WebControls;
07. using WebAirthmatic;
08. public partial class _Default : System.Web.UI.Page
09. {
10.     Airthmatic obj = new Airthmatic();
11.     int a, b, c;
12.     protected void Page_Load(object sender, EventArgs e)
13.     {
14.     }
15.     protected void btnAdd_Click(object sender, EventArgs e)
16.     {
17.         a = Convert.ToInt32(txtFno.Text);
18.         b = Convert.ToInt32(txtSno.Text);
19.         c = obj.Add(a, b);
20.         lblResult.Text = c.ToString();
21.     }
22.     protected void btnSub_Click(object sender, EventArgs e)
23.     {
24.         a = Convert.ToInt32(txtFno.Text);
25.         b = Convert.ToInt32(txtSno.Text);
26.         c = obj.Sub(a, b);
27.         lblResult.Text = c.ToString();
28.     }
29.     protected void BtnMul_Click(object sender, EventArgs e)
30.     {
31.         a = Convert.ToInt32(txtFno.Text);
32.         b = Convert.ToInt32(txtSno.Text);
33.         c = obj.Mul(a, b);
34.         lblResult.Text = c.ToString();
35.     }
36.     protected void btnDiv_Click(object sender, EventArgs e)
37.     {
38.         a = Convert.ToInt32(txtFno.Text);
39.         b = Convert.ToInt32(txtSno.Text);
40.         c = obj.Div(a, b);
41.         lblResult.Text = c.ToString();
42.     }
43. }
```

Now first run the Web service then the application.



Now you will be able to communicate with the web service.