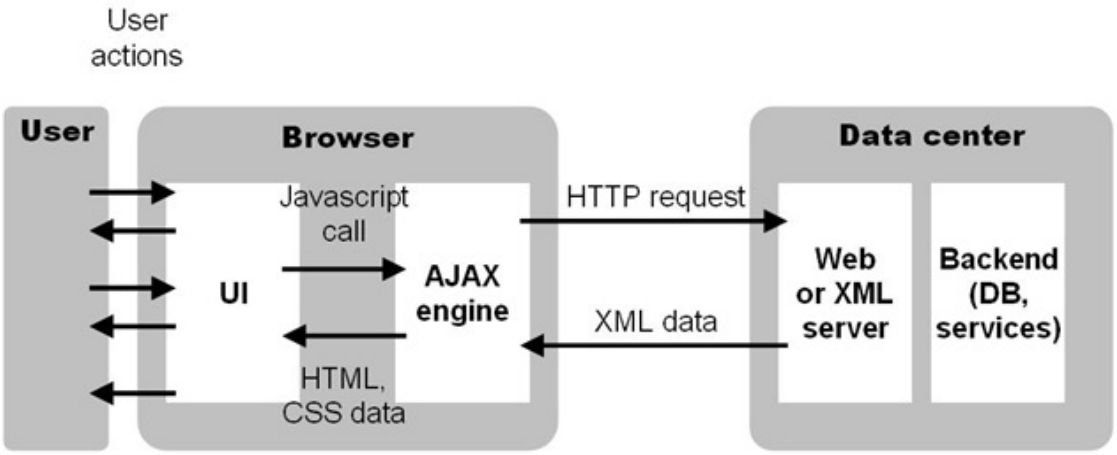
Ajax architecture

AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

- Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
- Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- AJAX is a web browser technology independent of web server software.
- A user can continue to use the application while the client program requests information from the server in the background.



Benefits of Ajax

There are 4 main benefits of using Ajax in web applications:

1. **Callbacks:** Ajax is used to perform a callback, making a quick round trip to and from the server to retrieve and/or save data without posting the entire page back to the server. By not performing a full postback and sending all form data to the server, network utilization

is minimized and quicker operations occur. In sites and locations with restricted bandwidth, this can greatly improve network performance. Most of the time, the data being sent to and from the server is minimal. By using callbacks, the server is not required to process all form elements. By sending only the necessary data, there is limited processing on the server. There is no need to process all form elements, process the ViewState, send images back to the client, or send a full page back to the client.

2. **Making Asynchronous Calls**: Ajax allows you to make asynchronous calls to a web server. This allows the client browser to avoid waiting for all data to arrive before allowing the user to act once more.

3. **User-Friendly:** Because a page postback is being eliminated, Ajax enabled applications will always be more responsive, faster and more user-friendly.

4. **Increased Speed:** The main purpose of Ajax is to improve the speed, performance and usability of a web application. A great example of Ajax is the movie rating feature on Netflix. The user rates a movie and their personal rating for that movie will be saved to their database without waiting for the page to refresh or reload. These movie ratings are being saved to their database without posting the entire page back to the server.

History

In the early-to-mid 1990s, most Web sites were based on complete HTML pages. Each user action required that a complete new page be loaded from the server. This process was inefficient, as reflected by the user experience: all page content disappeared, then the new page appeared. Each time the browser reloaded a page because of a partial change, all of the content had to be re-sent, even though only some of the information had changed. This placed additional load on the server and made bandwidth a limiting factor on performance.

- In 1996, the iframe tag was introduced by Internet Explorer; like the object element, it can load or fetch content asynchronously.
- In 1998, the Microsoft Outlook Web Access team developed the concept behind the XMLHttpRequest scripting object.
- It appeared as XMLHTTP in the second version of the MSXML library, which shipped with Internet Explorer 5.0 in March 1999.
- The functionality of the XMLHTTP ActiveX control in IE 5 was later implemented by Mozilla, Safari, Opera and other browsers as the XMLHttpRequest JavaScript object. Microsoft adopted the native XMLHttpRequest model as of Internet Explorer 7. The ActiveX version is still supported in Internet Explorer, but not in Microsoft Edge. The utility of these background HTTP requests and asynchronous Web technologies remained fairly obscure until it started appearing in large scale online applications such as Outlook Web Access (2000) and Oddpost (2002).
- In October 2004 Kayak.com's public beta release was among the first large-scale e-commerce uses of what their developers at that time called "the xml http thing"

- The term *Ajax* was publicly used on 18 February 2005 by Jesse James Garrett in an article titled *Ajax: A New Approach to Web Applications*, based on techniques used on Google pages.
- On 5 April 2006, the World Wide Web Consortium (W3C) released the first draft specification for the XMLHttpRequest object in an attempt to create an official Web standard
- The latest draft of the XMLHttpRequest object was published on 6 October 2016.

Aaplication of Ajax

AJAX cannot work independently. It is used in combination with other technologies to create interactive WebPages.

JavaScript

- Loosely typed scripting language.
- JavaScript function is called when an event occurs in a page.
- Glue for the whole AJAX operation.

DOM

- API for accessing and manipulating structured documents.
- Represents the structure of XML and HTML documents.

CSS

- Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript

XMLHttpRequest

- JavaScript object that performs asynchronous interaction with the server.

Ajax controls

Script manager

The ScriptManager control is the most important control and must be present on the page for other controls to work.

It has the basic syntax:

```
<asp:ScriptManager ID="ScriptManager1" runat="server">

</asp:ScriptManager>
```

If you create an 'Ajax Enabled site' or add an 'AJAX Web Form' from the 'Add Item' dialog box, the web form automatically contains the script manager control. The ScriptManager control takes care of the client-side script for all the server side controls.

Update panel

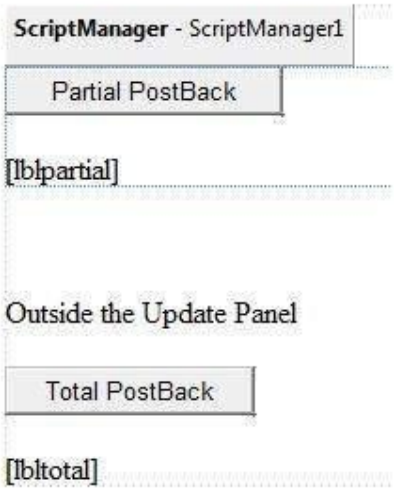
The UpdatePanel control is a container control and derives from the Control class. It acts as a container for the child controls within it and does not have its own interface. When a control inside it triggers a post back, the UpdatePanel intervenes to initiate the post asynchronously and update just that portion of the page.

For example, if a button control is inside the update panel and it is clicked, only the controls within the update panel will be affected, the controls on the other parts of the page will not be affected. This is called the partial post back or the asynchronous post back.

Example

Add an AJAX web form in your application. It contains the script manager control by default. Insert an update panel. Place a button control along with a label control within the update panel control. Place another set of button and label outside the panel.

The design view looks as follows:



The source file is as follows:

```
<form id="form1" runat="server">
  <div>
    <asp:ScriptManager ID="ScriptManager1" runat="server" />
  </div>

  <asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
      <asp:Button ID="btnpartial" runat="server" onclick="btnpartial_Click" Text="Partial
PostBack"/>
      <br />
      <br />
      <asp:Label ID="lblpartial" runat="server"></asp:Label>
    </ContentTemplate>
  </asp:UpdatePanel>

  <p> </p>
  <p>Outside the Update Panel</p>
```

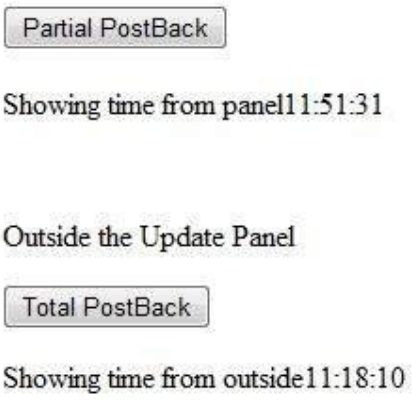
```
  <p>
    <asp:Button ID="btntotal" runat="server" onclick="btntotal_Click" Text="Total PostBack"
/>
  </p>

  <asp:Label ID="lbltotal" runat="server"></asp:Label>
</form>
```

Both the button controls have same code for the event handler:

```
string time = DateTime.Now.ToLongTimeString();
lblpartial.Text = "Showing time from panel" + time;
lbltotal.Text = "Showing time from outside" + time;
```

Observe that when the page is executed, if the total post back button is clicked, it updates time in both the labels but if the partial post back button is clicked, it only updates the label within the update panel.



A page can contain multiple update panels with each panel containing other controls like a grid and displaying different part of data.

When a total post back occurs, the update panel content is updated by default. This default mode could be changed by changing the UpdateMode property of the control. Let us look at other properties of the update panel.

Timer

The timer control is used to initiate the post back automatically. This could be done in two ways:

(1) Setting the Triggers property of the UpdatePanel control:

```
<Triggers>
  <asp:AsyncPostBackTrigger ControlID="btnpanel2" EventName="Click" />
</Triggers>
```

(2) Placing a timer control directly inside the UpdatePanel to act as a child control trigger. A single timer can be the trigger for multiple UpdatePanels.

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Always">

  <ContentTemplate>
    <asp:Timer ID="Timer1" runat="server" Interval="1000">
      </asp:Timer>

    <asp:Label ID="Label1" runat="server" Height="101px" style="width:304px" >
      </asp:Label>
  </ContentTemplate>

</asp:UpdatePanel>
```

Web services

What is Web Service?

A Web Service is a reusable piece of code used to communicate among Heterogeneous Applications.

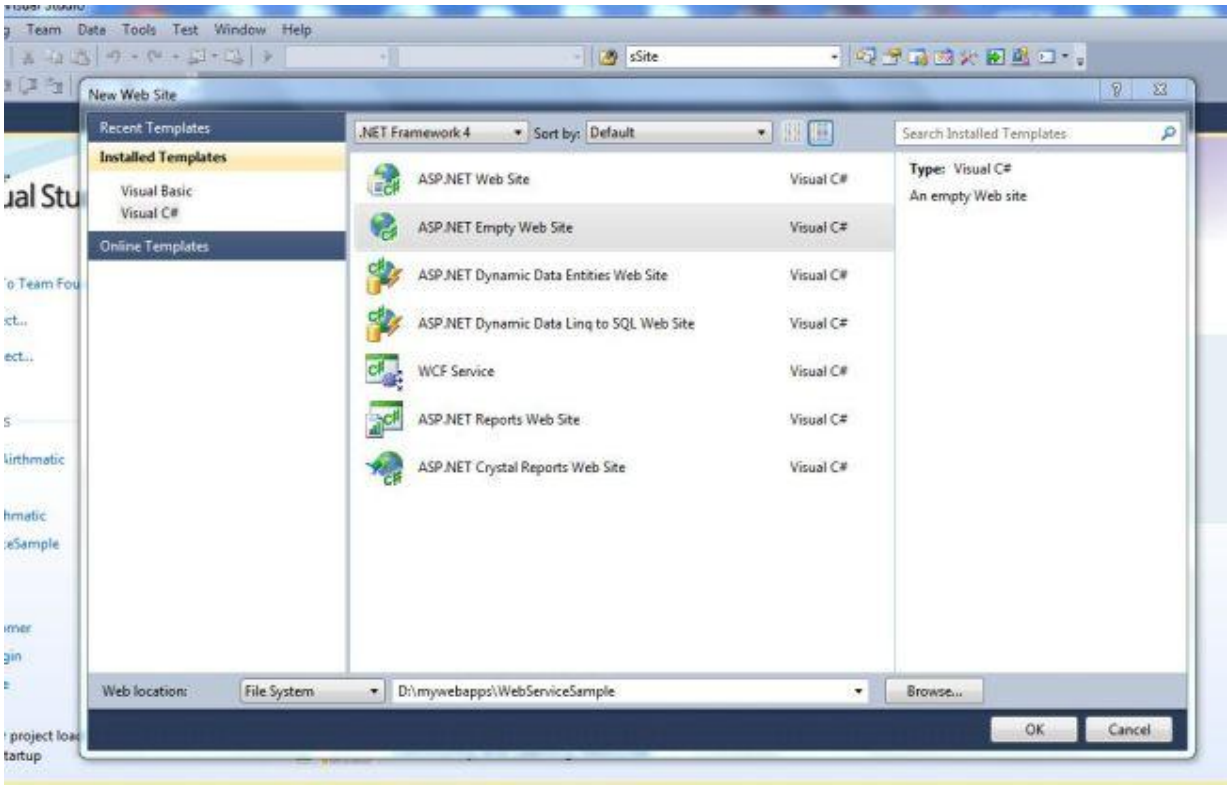
Once a web service is created and hosted on the server in the internet it can be consumed by any kind of application developed in any technology.

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).

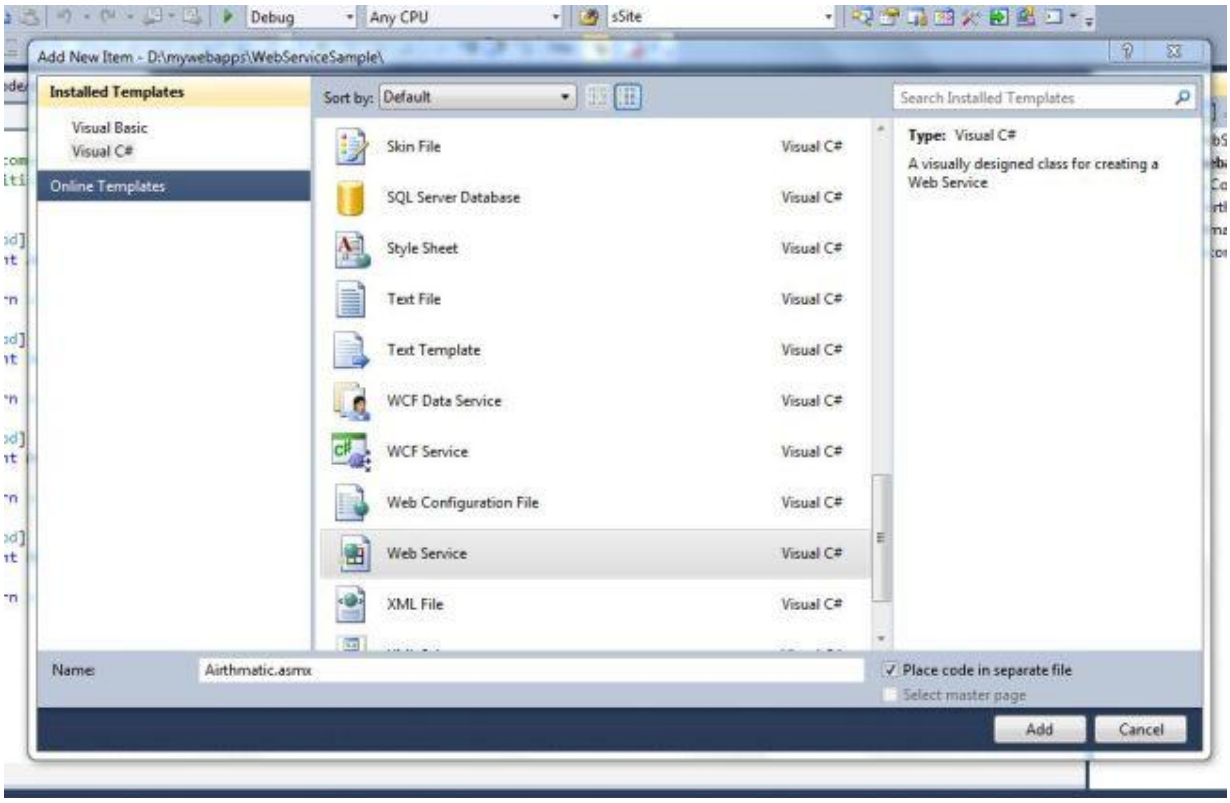


Step 2 Add a Web Service File

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

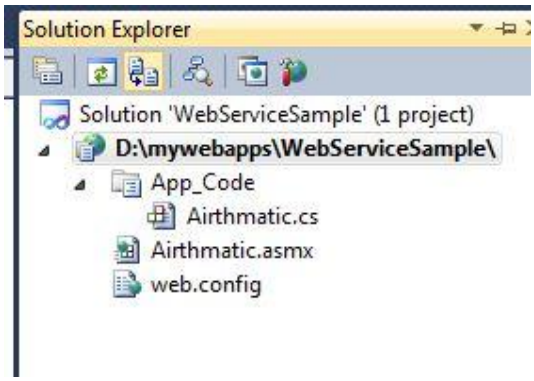
Choose the Web Service template.

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

```
1. using System;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using System.Web;
5.  using System.Web.Services;
```

```
6.  /// <summary>
7.  /// used for Airthmatic calculation
8.  /// </summary>
9.  [WebService(Namespace = "http://tempuri.org/")]
10. [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
11. // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomme
    nt the following line.
12. // [System.Web.Script.Services.ScriptService]
13. public class Airthmatic : System.Web.Services.WebService
14. {
15.     public Airthmatic()   {
16.         //Uncomment the following line if using designed components
17.         //InitializeComponent();
18.     }
19.     [WebMethod]
20.     public int Add(int x, int y)
21.     {
22.         return x + y;
23.     }
24.     [WebMethod]
25.     public int Sub(int x, int y)
26.     {
27.         return x - y;
28.     }
29.     [WebMethod]
30.     public int Mul(int x, int y)
31.     {
32.         return x * y;
33.     }
34.     [WebMethod]
35.     public int Div(int x, int y)
36.     {
37.         return x / y;
38.     }
39. }
```

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.

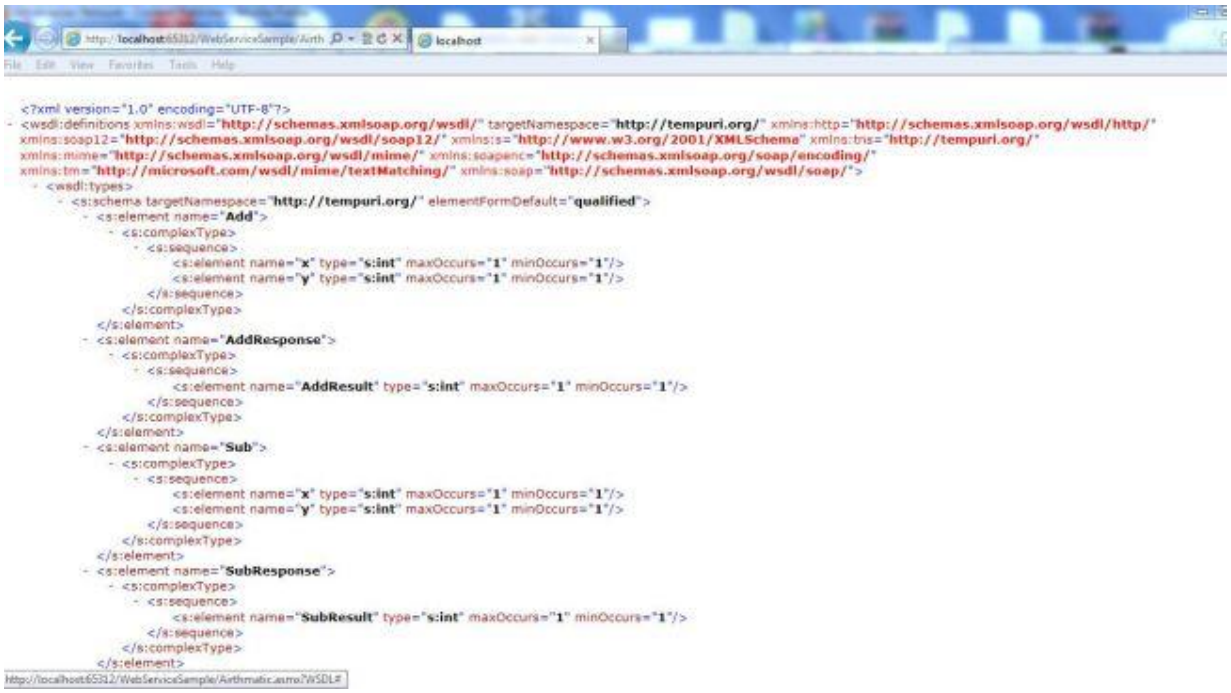
Now you will find all the method names in the browser.

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

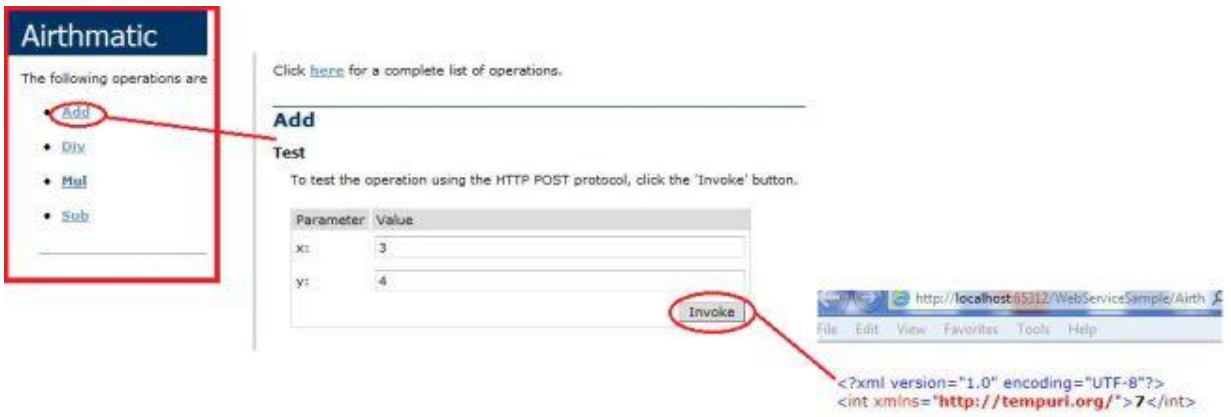
It will show the WSDL.



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.

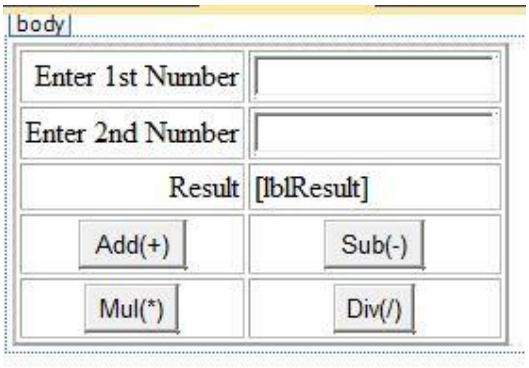
Now you will see the result in an open standard form (XML).



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.



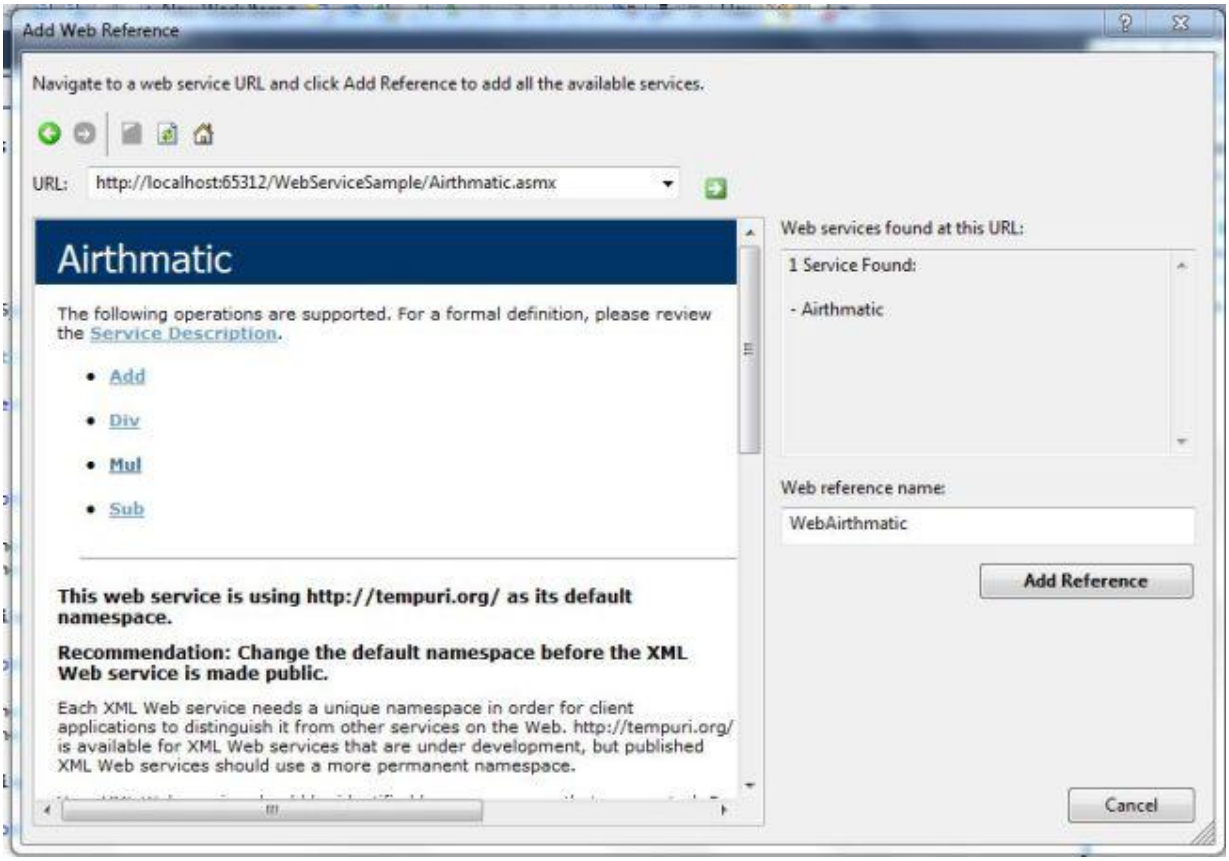
Step 5 Add a web reference to the Website

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.