Richard Stretanski

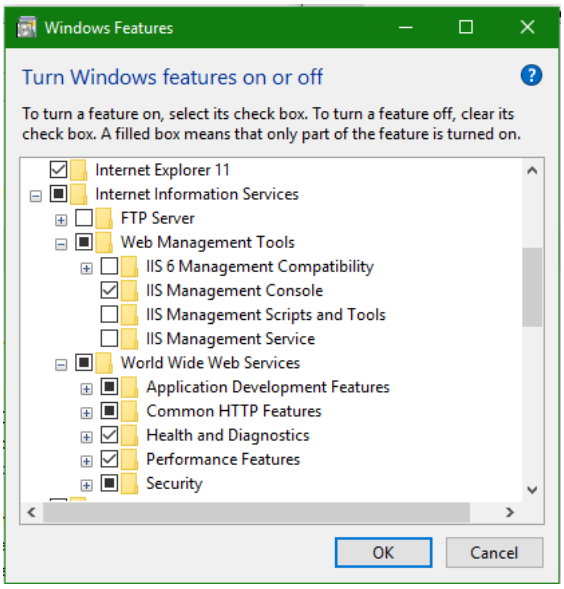
CIS-550-101

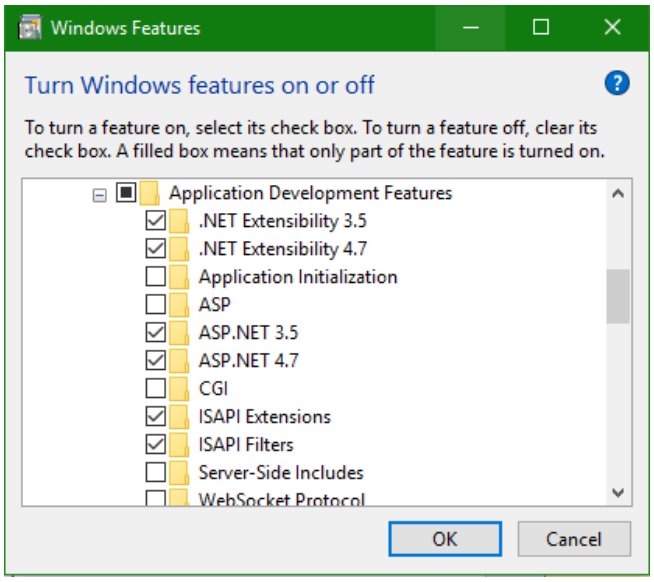
Calculator Web Service/Client

References and Environment: When creating these programs, I used Visual Studio Community 2015 Version 14.0.25431.01 Update 3 and Internet Information Services (IIS) Manager 10.0 on Windows 10. For help setting up IIS, I referenced Mohamad Ayyash's document: "Configuring IIS Server to run C# Web Services with ASP Clients" and I also referenced Perry James's document: "Program 7 – A Simple Calculator Web Service".

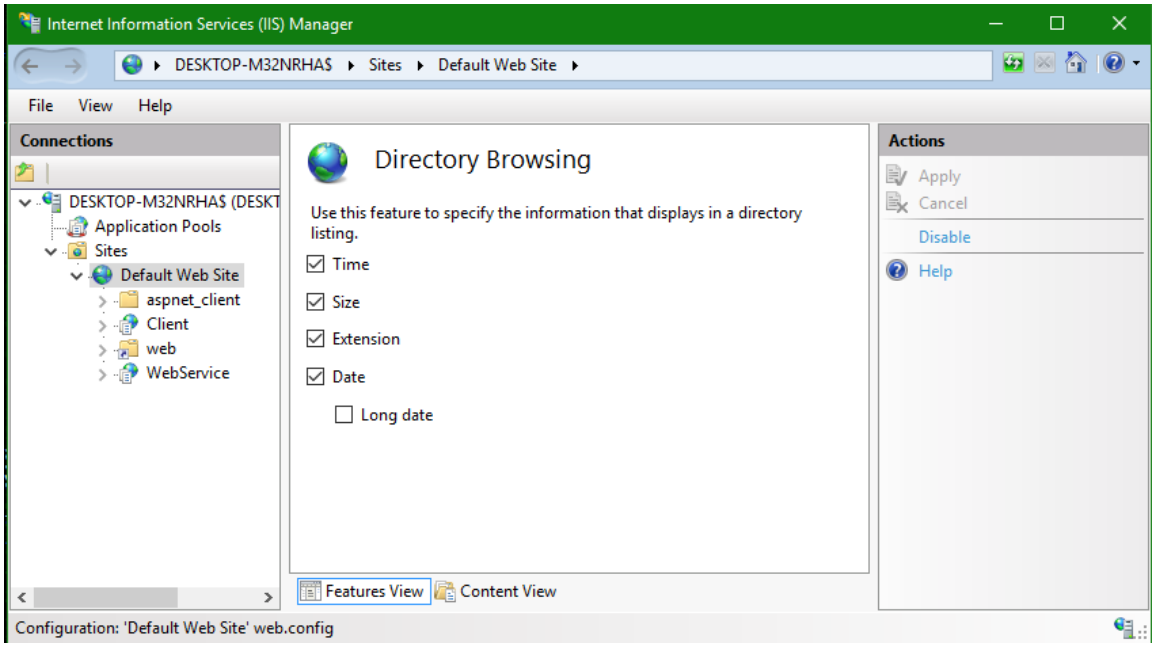
Project Explanation: The purpose of this project was to set up a local WCF service which used simple calculation methods and set up a Client web form that communicated with the service to execute the method calculations.

Install IIS Manager 10: The first thing I needed to do was install IIS because I had not already. IIS (Internet Information Service) is a service that allows you (for the purposes of this project) to locally host websites and website applications built in Visual Studio. To install stalled IIS 10.0 and the development features shown in the image below including ASP.NET 4.7.

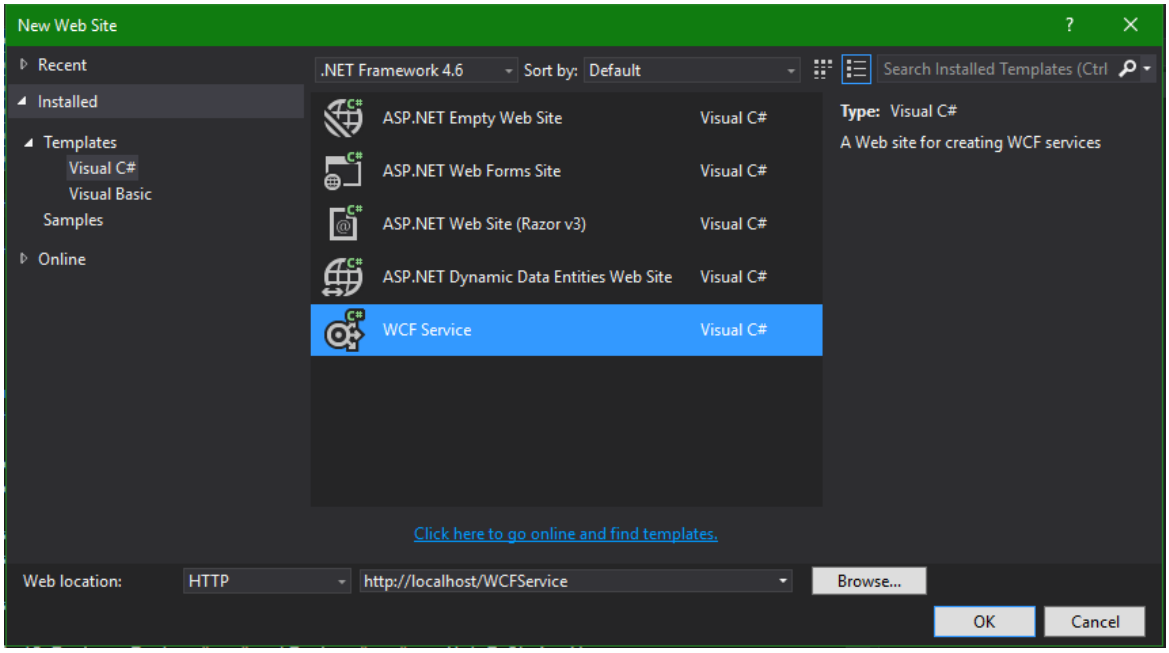




Setting up IIS for a Visual Studio Website: I created all my websites on Default Website: localhost. The only thing I needed to change in IIS is that I enabled directory browsing in Default Website >> Features View >> IIS >> Directory Browsing as shown below.



Creating a WCF Service: Next I created my WCF service website. To do this I ran Visual Studio as an Administrator and then created a new web site. The web location http://localhost is fine since that is the Default Website in IIS and I named the website: WebService and clicked OK so the location for the new website was http://localhost/WebService. The new website will automatically load in IIS as a program in the Default Web Site tree.

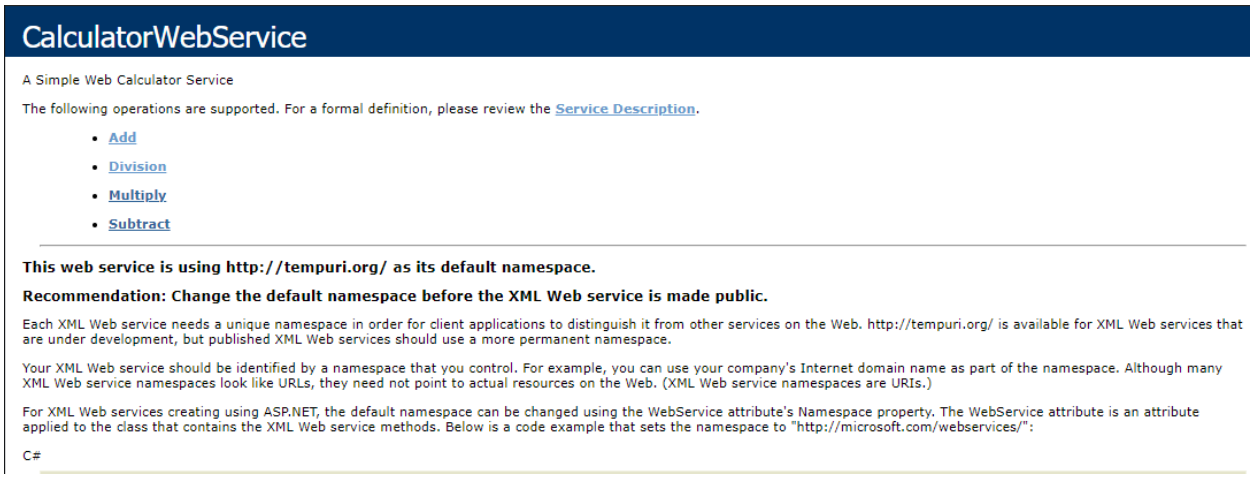


Adding Methods for Calculations: I then added a method for each calculation operation, Addition, Subtract, Multiply and Division as suggested by Perry James.

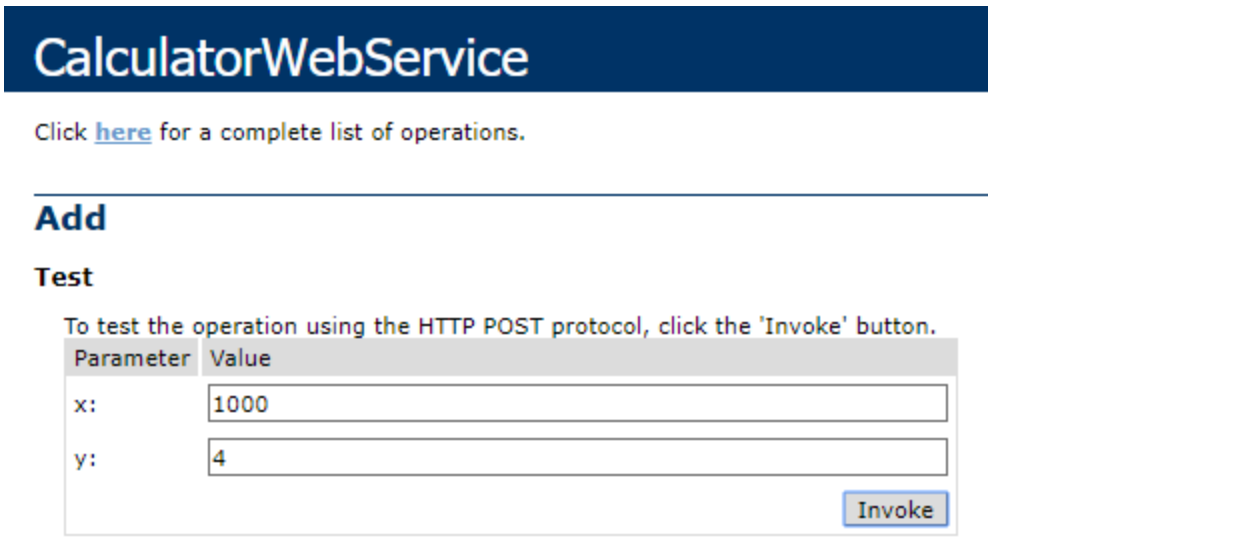
```
[WebMethod]
public float Add(float x, float y)
{
    return x + y;
}

[WebMethod]
public float Subtract(float x, float y)
{
    return x - y;
}

[WebMethod]
public float Multiply(float x, float y)
{
    return x * y;
}

[WebMethod]
public float Division(float x, float y)
{
    return x / y;
}
```

Testing the New Web Service: Now that the methods had been added I was able to test the web service. I could do this through Visual Studio, ISS or using a browser and going to the localhost link. The service appeared to work so I further tested the methods. Results are shown below: First is the page

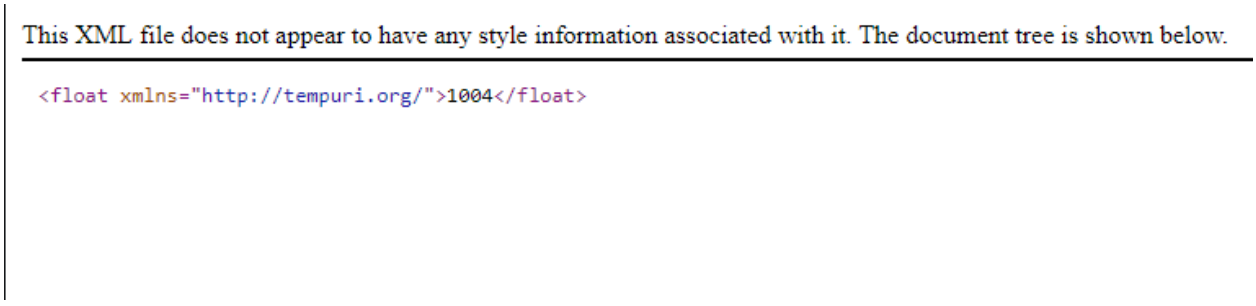
itself, where you can choose which method of calculation you want to use.



This is the 'Add' Method:



And 'Add' Result:



The 'Division' Method:

CalculatorWebService

Click here for a complete list of operations.

Division

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
x:	1000
y:	4

Invoke

And 'Division' Result:

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<float xmlns="http://tempuri.org/">250</float>
```

The 'Multiply' Method:

Click here for a complete list of operations.

Multiply

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
x:	1000
y:	4

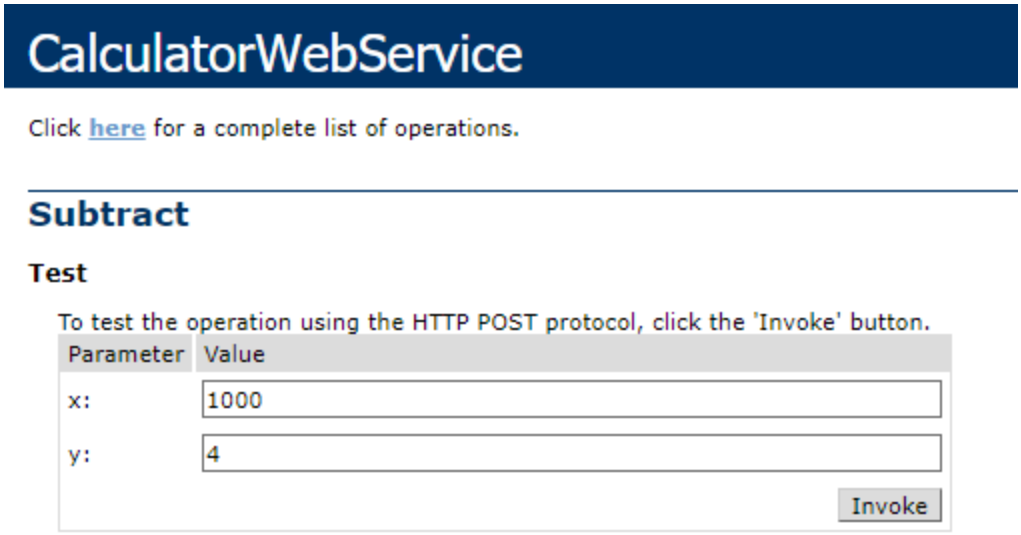
Invoke

And 'Multiply' Result:

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<float xmlns="http://tempuri.org/">4000</float>
```

The 'Subtract' Method:



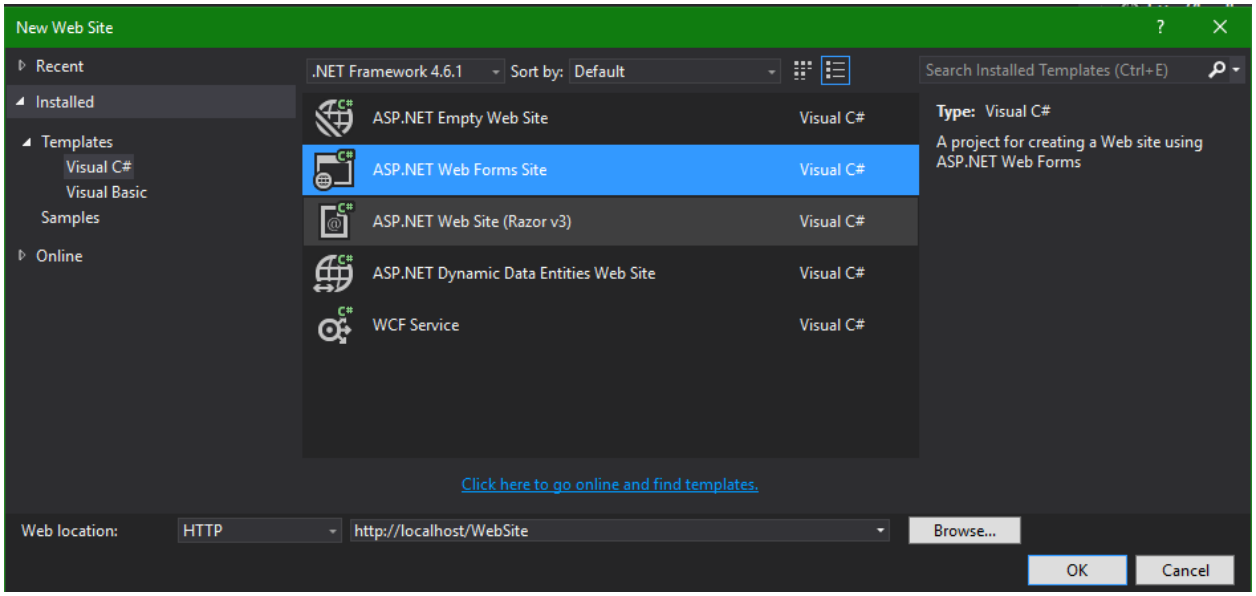
And 'Subtract' Results:

This XML file does not appear to have any style information associated with it. The document tree is shown below.

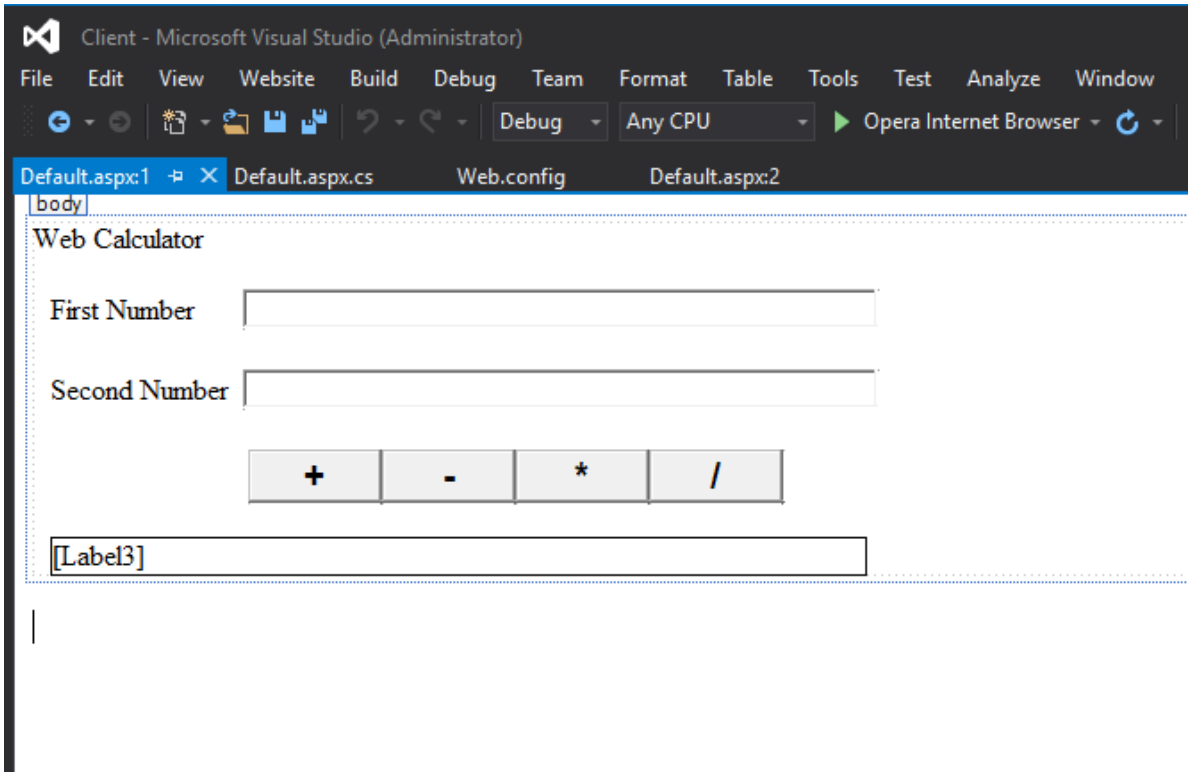
```
<float xmlns="http://tempuri.org/">996</float>
```

These methods appear to all be working fine. It doesn't matter that there isn't any style to the page because the end user should never see it. Instead we will be creating a client program using a web form, which will communicate with the WebService website, which is also what the user will end up seeing.

Creating the Client Website: For the client website I created a ASP.NET Web Forms Site and placed it in the same localhost directory as the WebService site. I will call it Client and it will have this path: http://localhost/Client. After creating the website ISS will, like before, automatically populate the Default Web Site tree with a project called Client.



Building the Web Form in the Client Website: The next step for me was to make the form in the Default.aspx file. I used 2 labels for the number fields and 2 text boxes right next to them and 4 buttons for each of the calculation methods and a third label for the return finished calculation. The design is as shown below:



Here is the code behind the design:

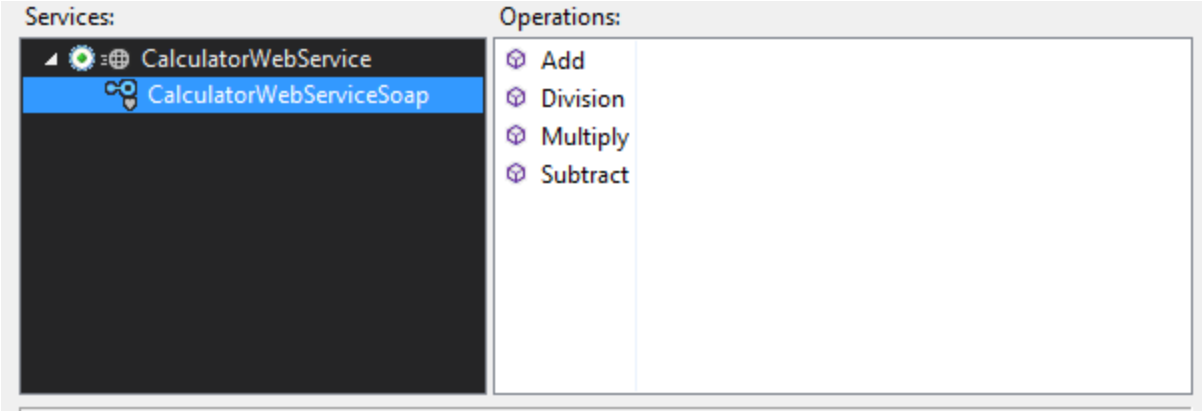
```
<body>
    <form id="form1" runat="server">
    <div>

        Web Calculator</div>
        <p style="margin-left: 10px">
            <asp:Label ID="Label1" runat="server" Height="24px" Text="First Number" Width="108px"></asp:Label>
            <asp:TextBox ID="TextBox1" runat="server" Height="16px" Width="348px"></asp:TextBox>
        </p>
        <p style="margin-left: 10px">
            <asp:Label ID="Label2" runat="server" Height="24px" Text="Second Number" Width="108px"></asp:Label>
            <asp:TextBox ID="TextBox2" runat="server" Height="16px" Width="348px"></asp:TextBox>
        </p>
        <p style="margin-left: 10px">
            <asp:Button ID="Button1" runat="server" Font-Bold="True" Font-Size="15pt" Height="30px" OnClick="Button1_Click" style="margin-left: 111px" Text="+" Width="75px" />
            <asp:Button ID="Button2" runat="server" Font-Bold="True" Font-Size="15pt" Height="30px" OnClick="Button2_Click" Text="-" Width="75px" />
            <asp:Button ID="Button3" runat="server" Font-Bold="True" Font-Size="15pt" Height="30px" OnClick="Button3_Click" Text="*" Width="75px" />
            <asp:Button ID="Button4" runat="server" Font-Bold="True" Font-Size="15pt" Height="30px" OnClick="Button4_Click" Text="/" Width="75px" />
        </p>
        <p style="margin-left: 10px">
            <asp:Label ID="Label3" runat="server" BorderColor="Black" BorderWidth="1px" Height="20px" Width="456px"></asp:Label>
        </p>

    </form>
</body>
```

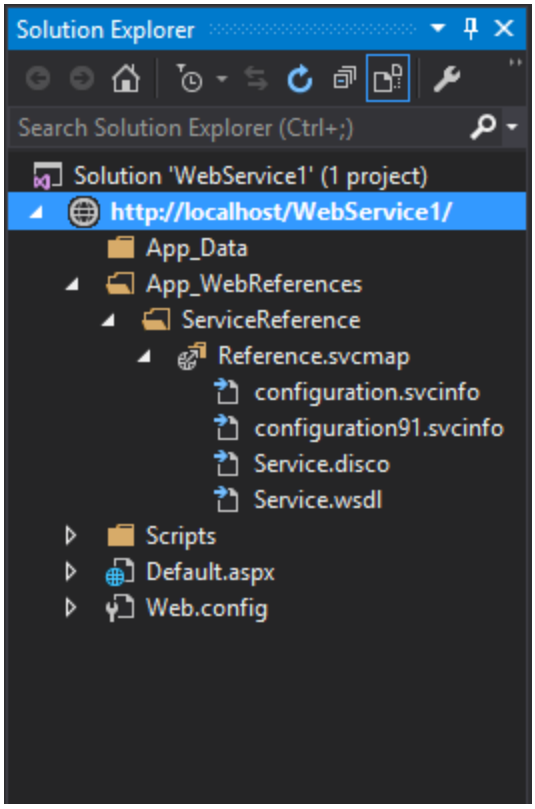
Adding Click EventArgs to the Buttons: The next step for me was to add Event methods that would complete an action when the buttons were clicked. I tried to keep the code as close as possible to the Windows form calculator I made for Project 6. Code for these methods are shown below:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

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

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        ServiceReference.CalculatorWebServiceSoapClient client =
            new ServiceReference.CalculatorWebServiceSoapClient();
        string aText, bText;
        float aVal, bVal, cVal;

        aText = TextBox1.Text;
        bText = TextBox2.Text;

        aVal = float.Parse(aText);
        bVal = float.Parse(bText);

        cVal = aVal + bVal;

        Label3.Text = aText + " + " + bText + " = " + cVal.ToString();
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        ServiceReference.CalculatorWebServiceSoapClient client =
            new ServiceReference.CalculatorWebServiceSoapClient();
        string aText, bText;
        float aVal, bVal, cVal;
```

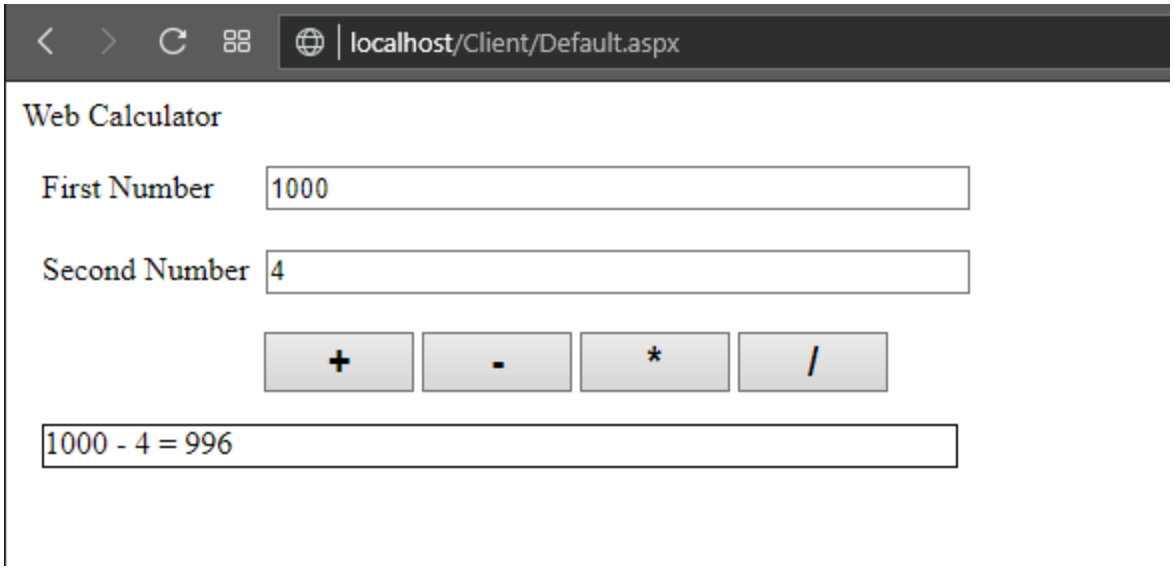
```
        aText = TextBox1.Text;
        bText = TextBox2.Text;

        aVal = float.Parse(aText);
        bVal = float.Parse(bText);

        cVal = aVal - bVal;

        Label3.Text = aText + " - " + bText + " = " + cVal.ToString();
    }

    protected void Button3_Click(object sender, EventArgs e)
    {
        ServiceReference.CalculatorWebServiceSoapClient client =
            new ServiceReference.CalculatorWebServiceSoapClient();
        string aText, bText;
        float aVal, bVal, cVal;

        aText = TextBox1.Text;
        bText = TextBox2.Text;

        aVal = float.Parse(aText);
        bVal = float.Parse(bText);

        cVal = aVal * bVal;

        Label3.Text = aText + " * " + bText + " = " + cVal.ToString();
    }

    protected void Button4_Click(object sender, EventArgs e)
    {
        ServiceReference.CalculatorWebServiceSoapClient client =
            new ServiceReference.CalculatorWebServiceSoapClient();
        string aText, bText;
        float aVal, bVal, cVal;

        aText = TextBox1.Text;
        bText = TextBox2.Text;

        aVal = float.Parse(aText);
        bVal = float.Parse(bText);

        cVal = aVal / bVal;

        Label3.Text = aText + " / " + bText + " = " + cVal.ToString();
    }
```

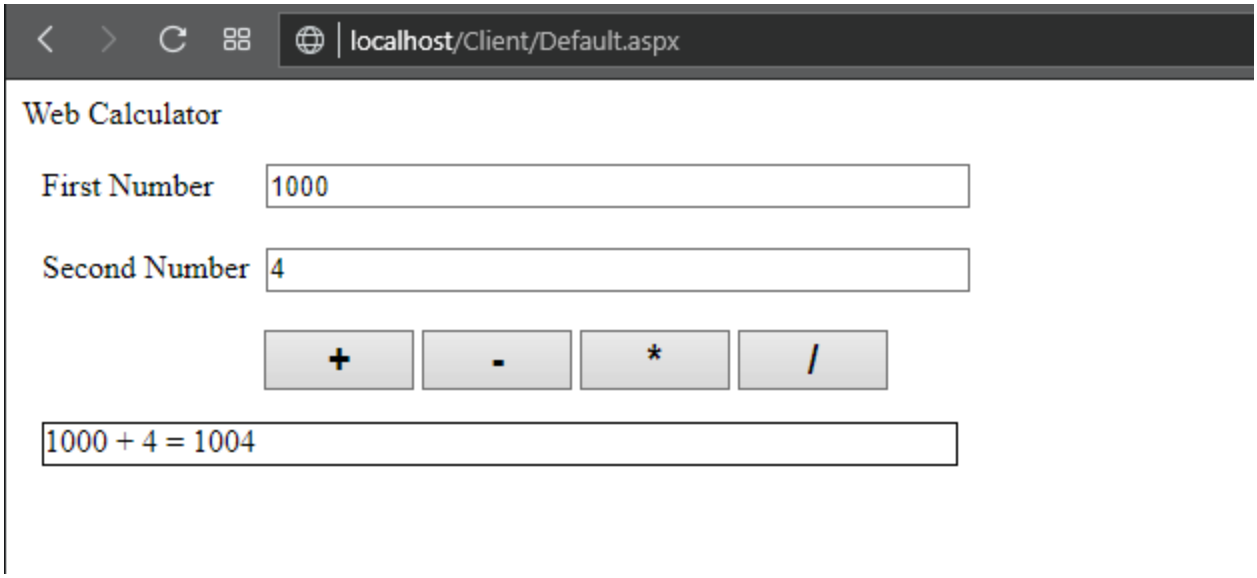
Adding the Service Website as a Service Reference: I now needed to add a reference in Visual Studio so that the Client website could communicate with the Service website. To do this I needed to right click on the localhost project in Visual Studio and click Add >> Service Reference. Then I entered the location of the Service file: http://localhost/WebService/Service.svc.

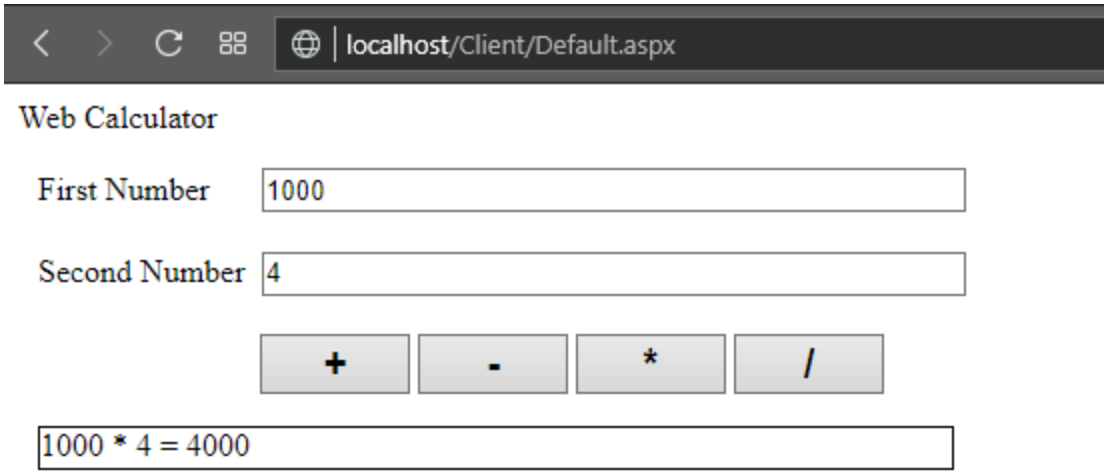


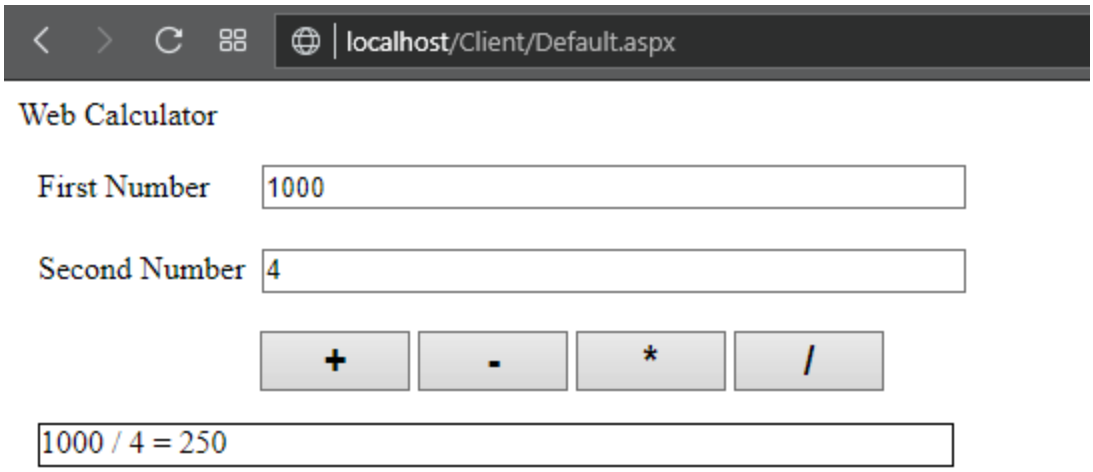


Testing the New Client Website: Now that I had finished adding the methods and reference to the Client website I could now test it. To do this I went to http://localhost/Client/Default.aspx and the results are posted below:


localhost/Client/Default.aspx
Web Calculator
First Number
1000
Second Number
4
+
-
*
/
1000 - 4 = 996


localhost/Client/Default.aspx
Web Calculator
First Number
1000
Second Number
4
+
-
*
/
1000 + 4 = 1004


localhost/Client/Default.aspx
Web Calculator
First Number
1000
Second Number
4
+
-
*
/
1000 * 4 = 4000


localhost/Client/Default.aspx
Web Calculator
First Number
1000
Second Number
4
+
-
*
/
1000 / 4 = 250