

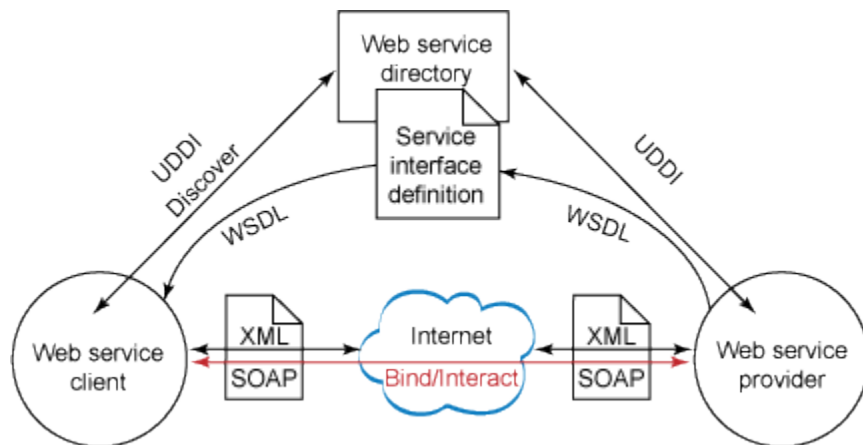
Web Services and Web API

Objectives

1. ศึกษาและทำความเข้าใจ Web Services
2. ทดลองสร้างและเรียกใช้ Web Services
3. ศึกษาและทำความเข้าใจ Web API
4. ทดลองสร้างและเรียกใช้ Web Services

ASP.NET Web Services (ASMX)

Web Services คือ มาตรฐานกลางในการสื่อสารข้อมูลระหว่าง Machine กับ Machine หรืออาจจะทำความเข้าใจง่าย ๆ ได้ว่า Web Services ก็คือ การให้บริการ Operation หรือ Method ที่สามารถเรียกใช้งานได้ผ่านเว็บ (HTTP) เช่น เรียกใช้ Method ส่ง SMS, เรียกใช้ Method การแปลงหน่วยเงิน Currency เป็นต้น โดยมีมาตรฐานย่อย 3 ส่วนหลัก คือ SOAP, WSDL และ UDDI



SOAP ย่อมาจาก Simple Object Access Protocol เป็นรูปแบบเอกสาร (XML Base) ที่มีข้อตกลงในการส่งข้อมูลระหว่าง Web Services Client และ Web Services Provider ซึ่งตรงนี้เทียบได้กับการตกลงเรื่องภาษาที่ใช้จะคุยกัน ถ้าคุยกันคนละภาษาก็คงคุยกันไม่รู้เรื่อง

WSDL ย่อมาจาก Web Services Definition Language เป็นเหมือนคู่มือการใช้งาน Web Services (XML Base) โดย WSDL จะบอกทั้งหมดว่าสามารถเรียกใช้ Methods/Operations อะไรได้บ้าง มี Input/Output อย่างไร

UDDI ย่อมาจาก **Universal Description, Discovery, and Integration** เปรียบเสมือนเป็นสมุดหน้าเหลืองของ **Web Services** รวบรวมบริการต่างๆ ไว้ในที่เดียว เพื่อให้สะดวกในการค้นหาบริการและนำมาใช้งาน

จุดเด่น คือ มีความเป็นมาตรฐาน (W3C) พัฒนาง่าย และเรียกใช้งานง่าย ยังมีการใช้งานอยู่ในระดับ Enterprise

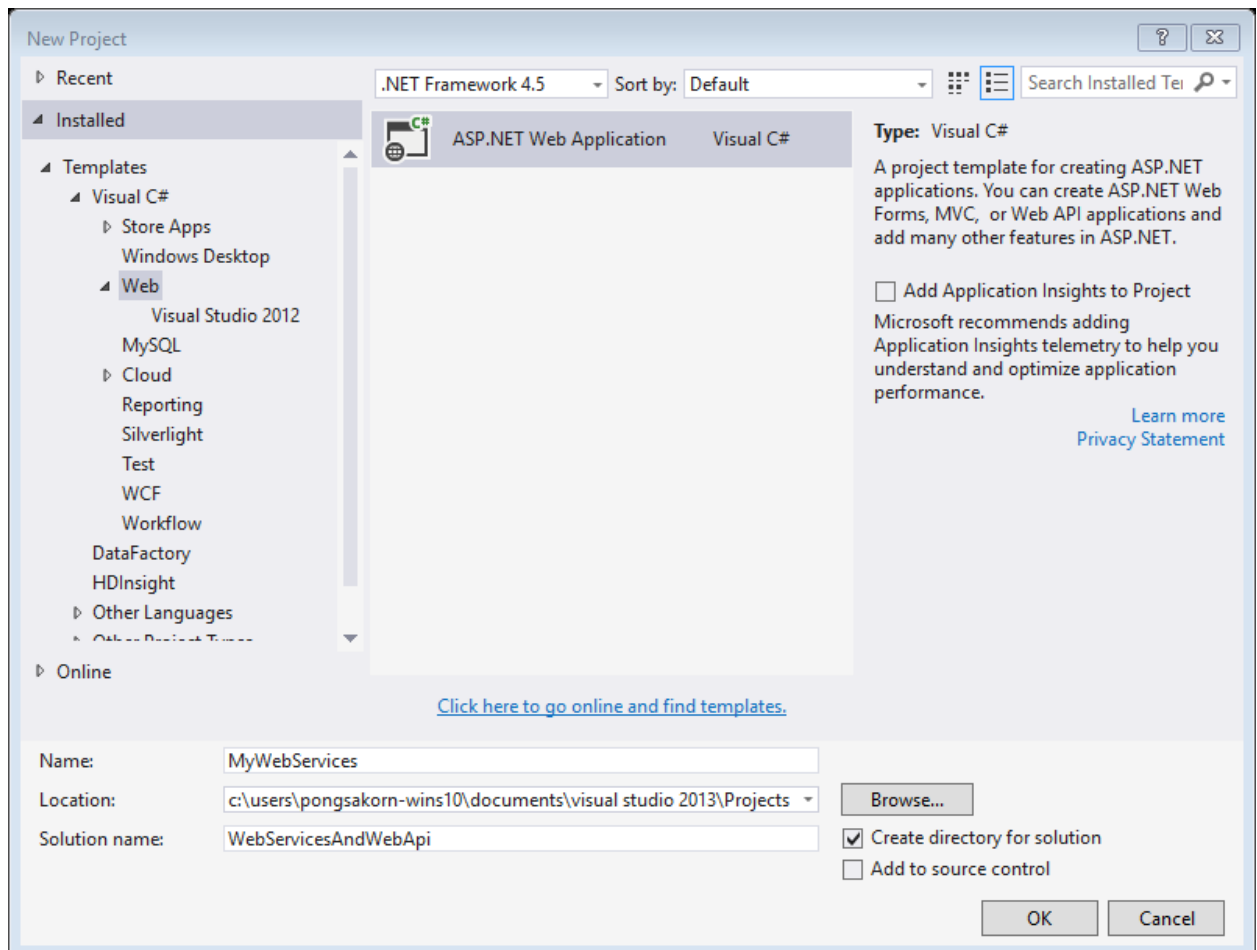
จุดด้อย คือ ใช้เวลาในการเรียนรู้นาน ขนาดข้อมูลที่ส่งไปมาระหว่างกันมีขนาดใหญ่กว่าที่ควรจะเป็น (Overhead)

Creating and Invoking Web Services using ASP.NET Technology

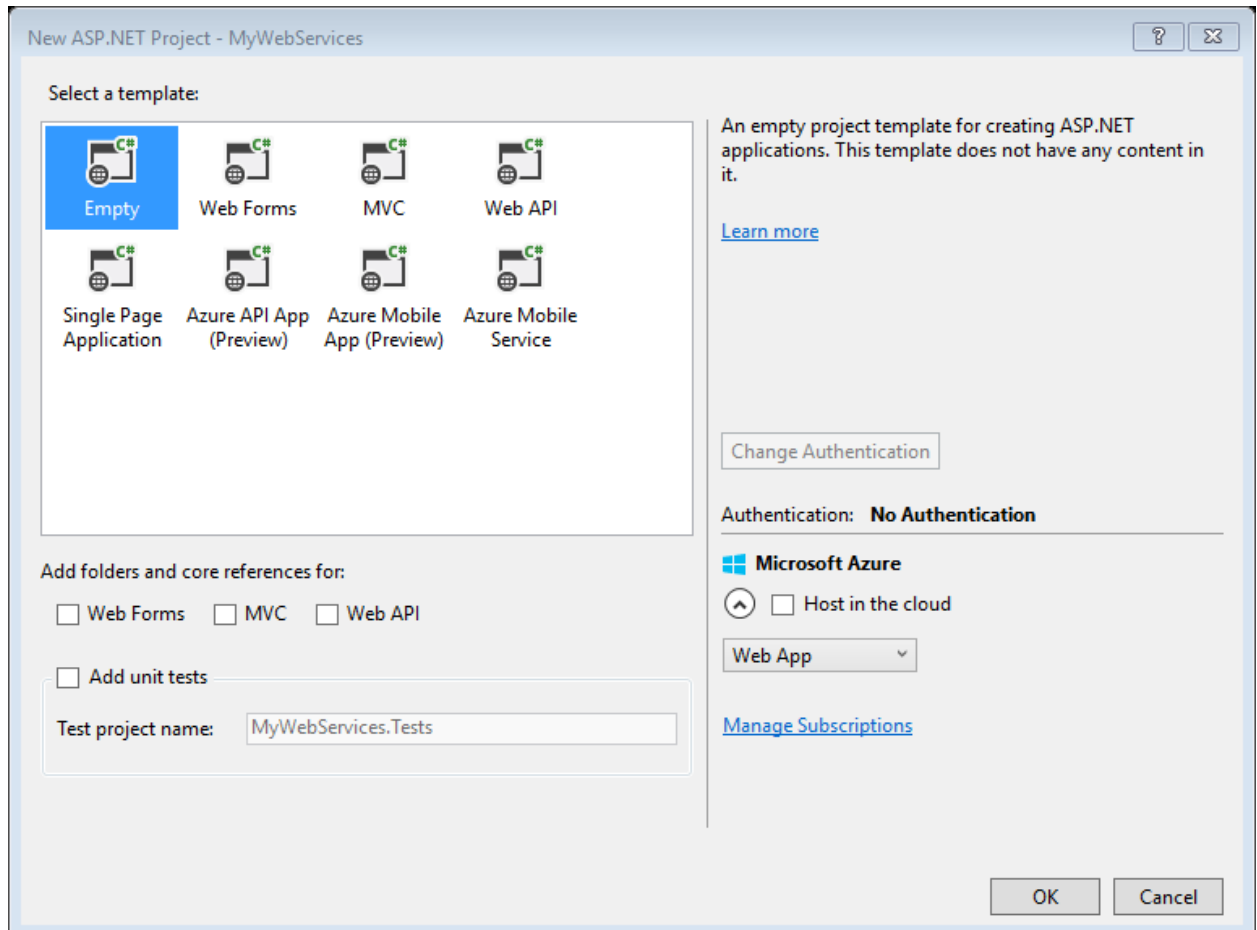
สำหรับ **Workshop** นี้จะให้ทดลองสร้าง **Web Services** สำหรับคำนวณหาการแปลงอุณหภูมิระหว่าง องศาเซลเซียสและองศาฟาเรนไฮต์ (Temperature Converter) โดยมี 2 Methods เพื่อทำหน้าที่เป็น Web Services ดังนี้

C#
<pre>public double CelsiusToFahrenheit(double celsius) { double fahrenheit = 0; fahrenheit = (((celsius) * 9) / 5) + 32); return fahrenheit ; } public double FahrenheitToCelsius(double fahrenheit){ double celsius = 0 ; celsius = (((fahrenheit) - 32) / 9) * 5) ; return celsius ; }</pre>

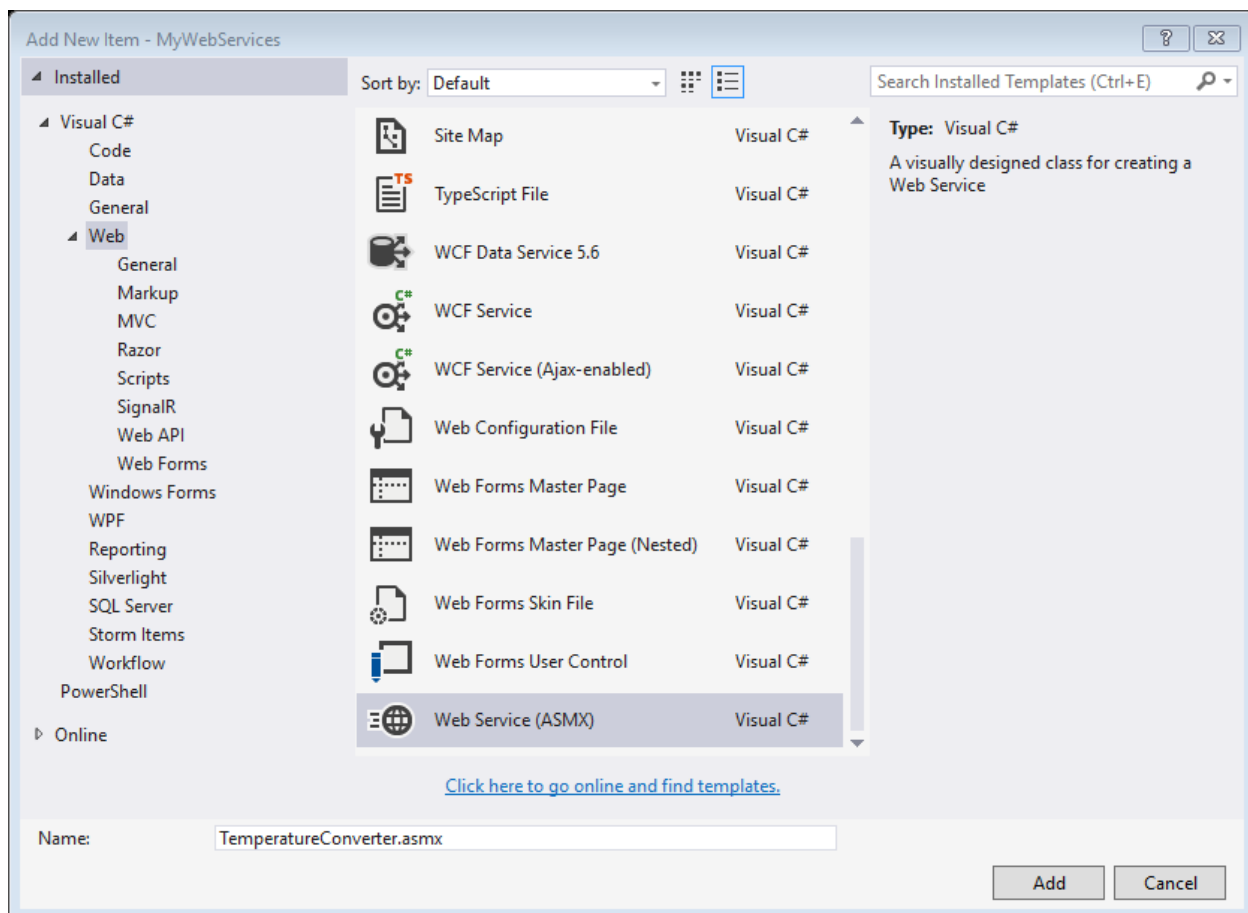
1. เปิด Visual Studio 2013 ไปที่ FILE -> New -> Project ตั้งชื่อ Project ว่า MyWebServices และ Solution ชื่อว่า WebServicesAndWebApi จากนั้นกดปุ่ม OK



2. เลือก ASP.NET Project Template แบบ Empty จากนั้นกดปุ่ม OK



3. คลิกขวาที่ MyWebServices Project จากนั้นไปที่ Add -> New Item เลือก Web Service (ASMX) จากนั้นตั้งชื่อไฟล์ว่า TemperatureConverter.asmx จากนั้นกดปุ่ม Add



4. จะได้ไฟล์ Web Service ตามตัวอย่างด้านล่าง โดย Visual Studio จะสร้าง WebMethod มาให้ 1 ตัวเพื่อเป็นตัวอย่าง ทดลองรันโปรแกรมโดยกดปุ่ม F5

Code TemperatureConverter.asmx

```

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Web;
5. using System.Web.Services;
6.
7. namespace MyWebServices
8. {
9.     /// <summary>
10.    /// Summary description for TemperatureConverter
11.    /// </summary>
12.    [WebService(Namespace = "http://tempuri.org/")]
13.    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]

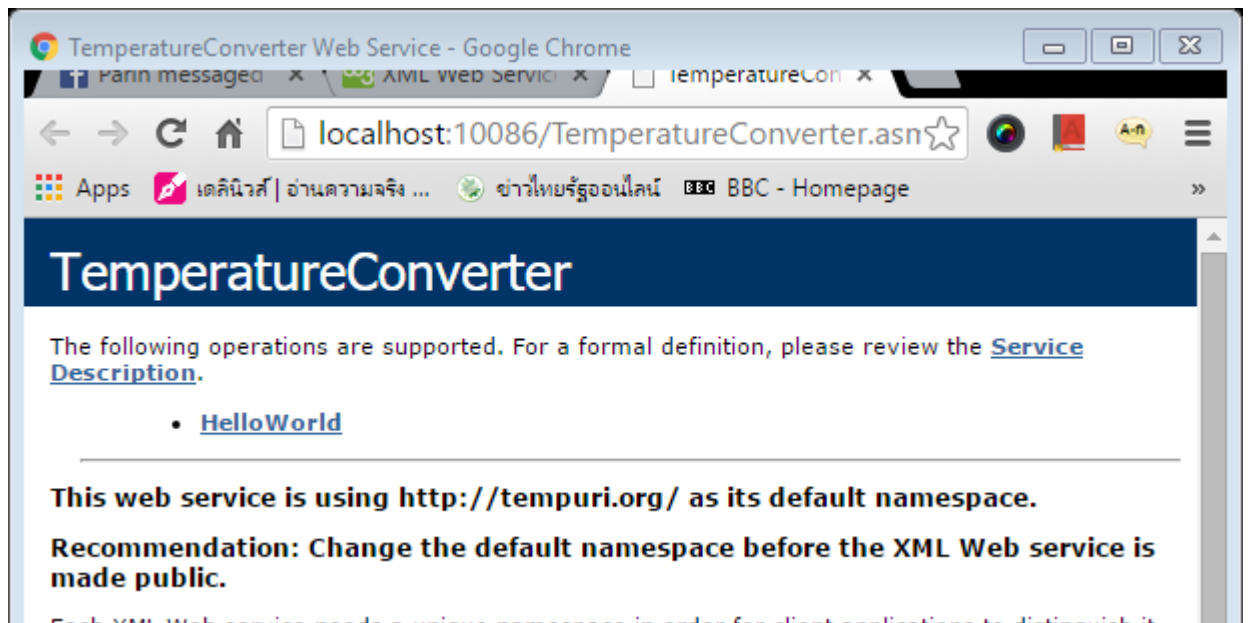
```

```

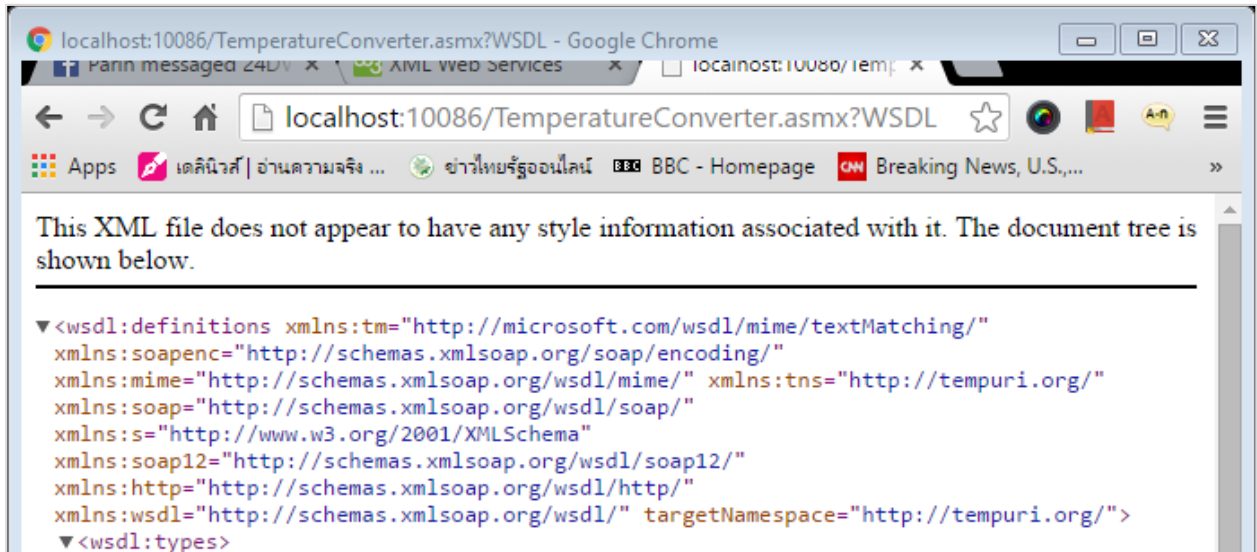
14. [System.ComponentModel.ToolboxItem(false)]
15. // To allow this Web Service to be called from script, using ASP.NET
    AJAX, uncomment the following line.
16. // [System.Web.Script.Services.ScriptService]
17. public class TemperatureConverter : System.Web.Services.WebService
18. {
19.
20.     [WebMethod]
21.     public string HelloWorld()
22.     {
23.         return "Hello World";
24.     }
25. }
26. }

```

5. เมื่อรันแล้วจะเปิด Web Browser ไปที่ URL: <http://localhost:10086/TemperatureConverter.asmx> ซึ่งเรียกว่า Endpoint ของ WebServices จะเห็นว่ามีกาแสดงลิสต์ของ Method/Operation ที่สามารถเรียกใช้งานได้ ให้ทดลองกดที่ลิงค์ Service Description (WSDL) และ HelloWorld (Operation Manual)

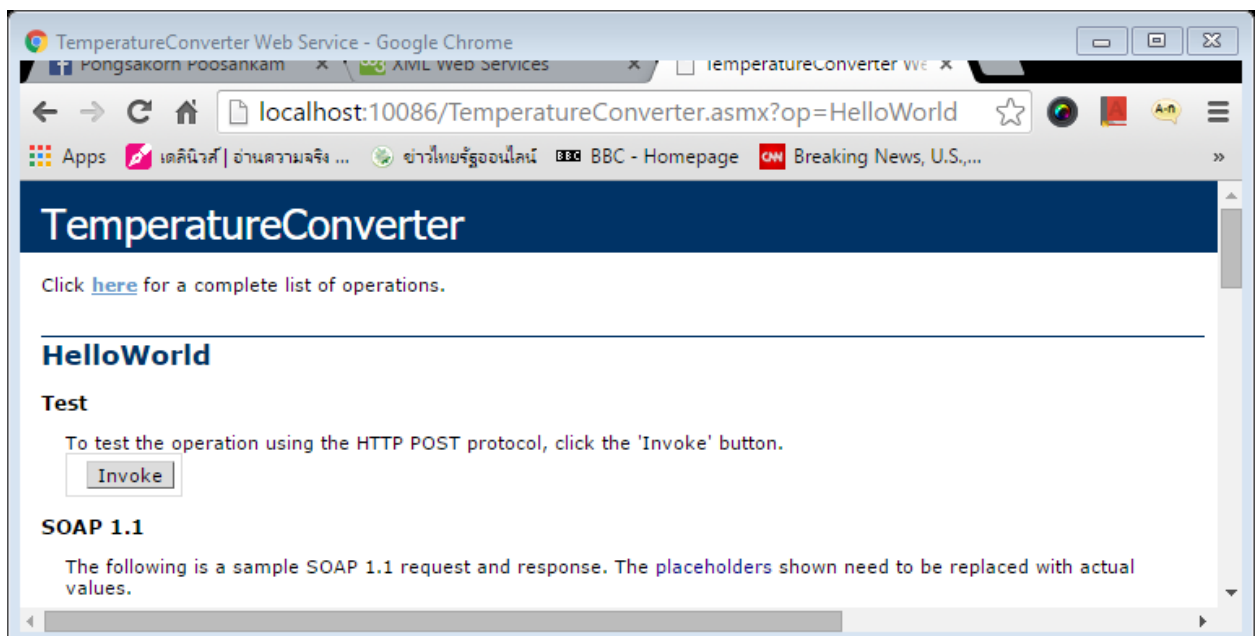


6. แสดงหน้าจอของ WSDL ซึ่งเป็น XML Format อธิบายถึง Operation ต่างๆ และวิธีการส่ง Input/Output โดยปกติแล้วจะมีโปรแกรมอ่านและสรุปความหมายของเอกสาร WSDL มาให้ ซึ่งก็คือหน้าจอที่มีการลิ้งค์ Operation ในข้อ 5



7. แสดงหน้าจอรายละเอียดของ Operation HelloWorld เมื่อกดปุ่ม Invoke จะมีการเรียกใช้ Method ที่ HelloWorld ในโปรแกรม โดยจะแสดงข้อความ Hello World ออกมา

```
<string xmlns="http://tempuri.org/">Hello World</string>
```



8. ทำการเพิ่ม Method สำหรับ Convert อุณหภูมิเข้าไปในไฟล์ TemperatureConverter.asmx

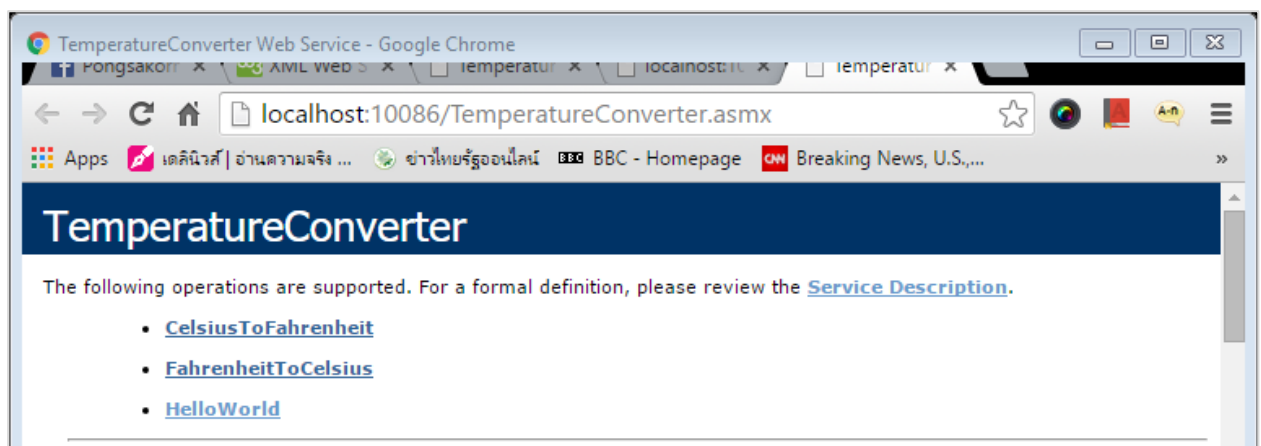
Code TemperatureConverter.asmx

```
public class TemperatureConverter : System.Web.Services.WebService
{
    [WebMethod]
    public string HelloWorld()
    {
        return "Hello World";
    }

    [WebMethod]
    public double CelsiusToFahrenheit(double celsius)
    {
        double fahrenheit = 0;
        fahrenheit = (((celsius) * 9) / 5) + 32;
        return fahrenheit;
    }

    [WebMethod]
    public double FahrenheitToCelsius(double fahrenheit)
    {
        double celsius = 0;
        celsius = (((fahrenheit) - 32) / 9) * 5;
        return celsius;
    }
}
```

9. ทดลองรันโดยกดปุ่ม F5 อีกครั้ง จะปรากฏลิสต์ของ Operations เพิ่มขึ้นอีกสองตัวคือ CelsiusToFahrenheit และ FahrenheitToCelsius จากนั้นให้ทดลองกดปุ่ม Invoke เพื่อเรียกใช้ Operation แต่ละตัว



10. เมื่อกดลิงค์ CelsiusToFahrenheit และกดปุ่ม Invoke จะปรากฏ Input Box ให้กรอกตัวเลข

TemperatureConverter

Click [here](#) for a complete list of operations.

CelsiusToFahrenheit

Test

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

Parameter	Value
celsius:	<input type="text" value="30"/>

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

```
<double xmlns="http://tempuri.org/">86</double>
```

TemperatureConverter

Click [here](#) for a complete list of operations.

FahrenheitToCelsius

Test

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

Parameter	Value
fahrenheit:	<input type="text" value="100"/>

สามารถทดลองเรียกใช้ Web Services โดยส่ง HTTP Message ไปยัง Web Services Endpoint ได้ โดยระบุ SOAP Message ตามตัวอย่างใน Operation Detail Page

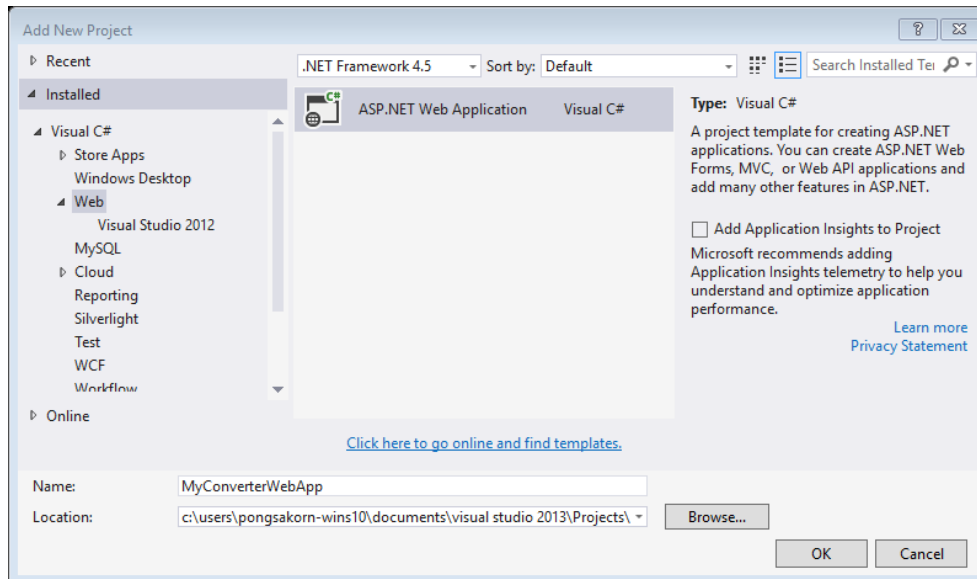
ติดตั้ง Chrome Plugin ชื่อว่า “Postman Launcher” เพื่อทดสอบการส่ง SOAP Message

11. ทดลองเรียนใช้ **WebServices** ที่ได้สร้างขึ้นมา โดยจะทำการสร้าง **WebServices Client** จาก **WSDL**

<http://temperatureconverter.azurewebsites.net/TemperatureConverter.asmx?WSDL>

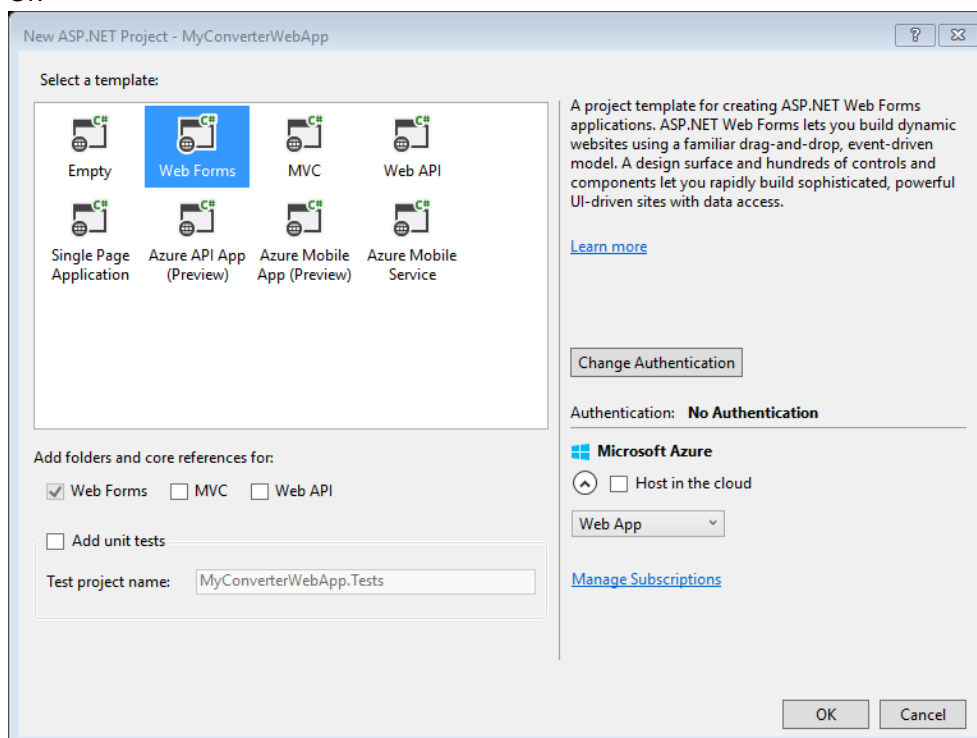
โดยทำการเรียกใช้ผ่านลิงค์ที่ได้ทำการติดตั้งไว้แล้วที่ **Windows Azure**

12. คลิกขวาที่ **Solution** เลือก **Add -> New Project** และเลือก **Web** ตั้งชื่อ **Project** ว่า **MyConverterWebApp**

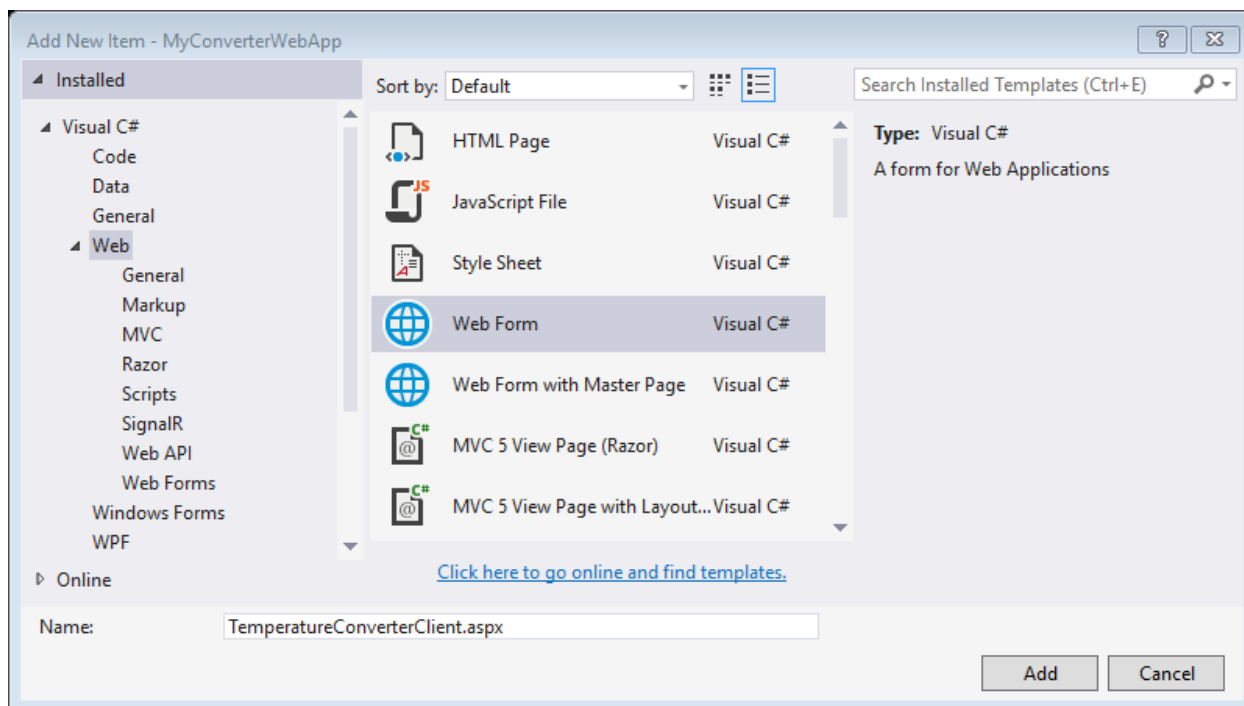


13. เลือก **Project Template** แบบ **Web Form** และ **Authentication Type : No Authentication** จากนั้นกดปุ่ม

OK



14. คลิกขวาที่ Project MyConverterWebApp เลือก Add-> New Item ... จากนั้นเลือก Web Form และตั้งชื่อไฟล์ว่า **TemperatureConverterClient.aspx** จากนั้นกดปุ่ม OK



15. ทำการเพิ่มโค้ดลงไปตามตัวอย่างด้านล่าง เพื่อสร้างฟอร์มรับข้อมูลสำหรับส่งไปเรียกใช้ Web Services

Code TemperatureConverterClient.aspx

```

1. <%@ Page Language="C#" AutoEventWireup="true"
   CodeBehind="TemperatureConverterClient.aspx.cs"
   Inherits="MyConverterWebApp.TemperatureConverterClient" %>
2.
3. <!DOCTYPE html>
4.
5. <html xmlns="http://www.w3.org/1999/xhtml">
6. <head runat="server">
7. <title></title>
8. </head>
9. <body>
10.
11. <form id="form1" runat="server">
12. <div>
13. <asp:Label ID="lb" runat="server" Text="Celcius"></asp:Label>
14. <br />
15. <asp:TextBox ID="txtCelcius" runat="server"></asp:TextBox>
16. <br />
17. <asp:Label ID="lbTextFarenhite" runat="server" Text="Farenhite :
   "></asp:Label>
18. <asp:Label ID="lbResult" runat="server" Text=""></asp:Label>
19. <br />

```

```

20.      <asp:Button ID="btnConvert" OnClick="btnConvert_Click" runat="server"
      Text="Convert" />
21.    </div>
22.  </form>
23. </body>
24. </html>

```

16. ในส่วนของโค้ดจะมี Method btnConvert_Click เพื่อเรียก Web Services เมื่อกดปุ่ม Convert

Code TemperatureConverterClient.aspx.cs

```
using System;
```

```
namespace MyConverterWebApp
```

```
{
```

```
    public partial class TemperatureConverterClient : System.Web.UI.Page
    {
```

```
        protected void Page_Load(object sender, EventArgs e)
        {

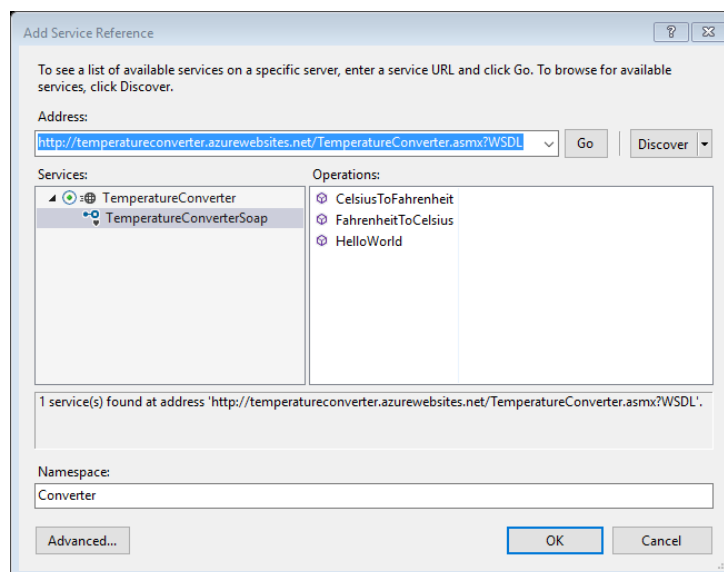
        }
    }
```

```
        protected void btnConvert_Click(object sender, EventArgs e)
        {
            //Call Web Services Put Here
        }
    }
```

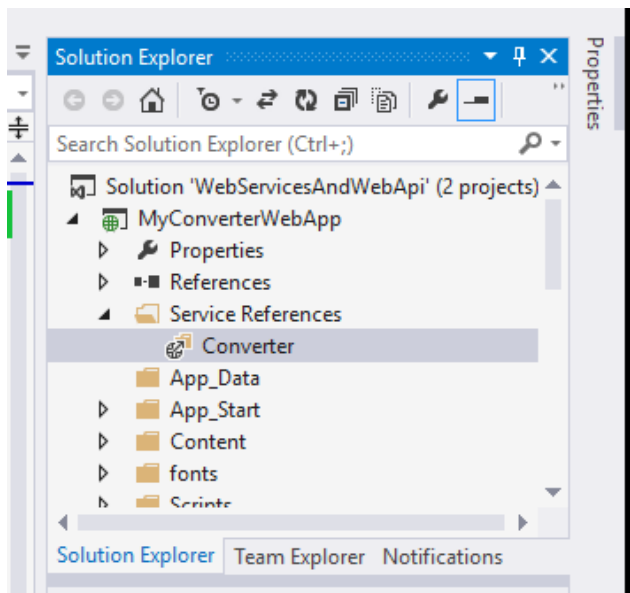
```
}
```

17. คลิกขวาที่ Project จากนั้นเลือก Add -> Service Reference ... จากนั้นระบบ WSDL และกดปุ่ม GO ตั้งชื่อ

Namespace เป็น Converter และกดปุ่ม OK เพื่อให้ Visual Studio ช่วยสร้างโค้ดสำหรับเรียกใช้ Web Services



18. สังเกตว่าจะมี Converter ปรากฏอยู่ใน Service References จากนั้นเราจะสามารถเรียกใช้งาน Web Services TemperatureConverter ที่อยู่บน Cloud ได้ผ่านทาง Namespace Converter

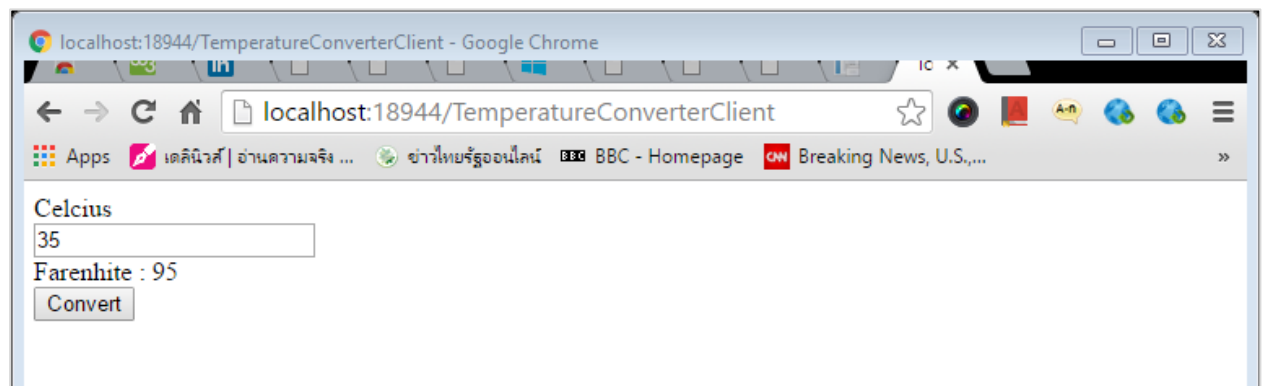


19. เพิ่มโค้ดสำหรับเรียกใช้ Converter ในไฟล์ TemperatureConverterClient.aspx.cs

Code

```
protected void btnConvert_Click(object sender, EventArgs e)
{
    Converter.TemperatureConverterSoapClient client =
        new Converter.TemperatureConverterSoapClient();
    lblResult.Text =
        client.CelsiusToFahrenheit(Double.Parse(txtCelcius.Text)).ToString();
}
```

20. ทดลองรันโปรแกรมโดยกดปุ่ม F5

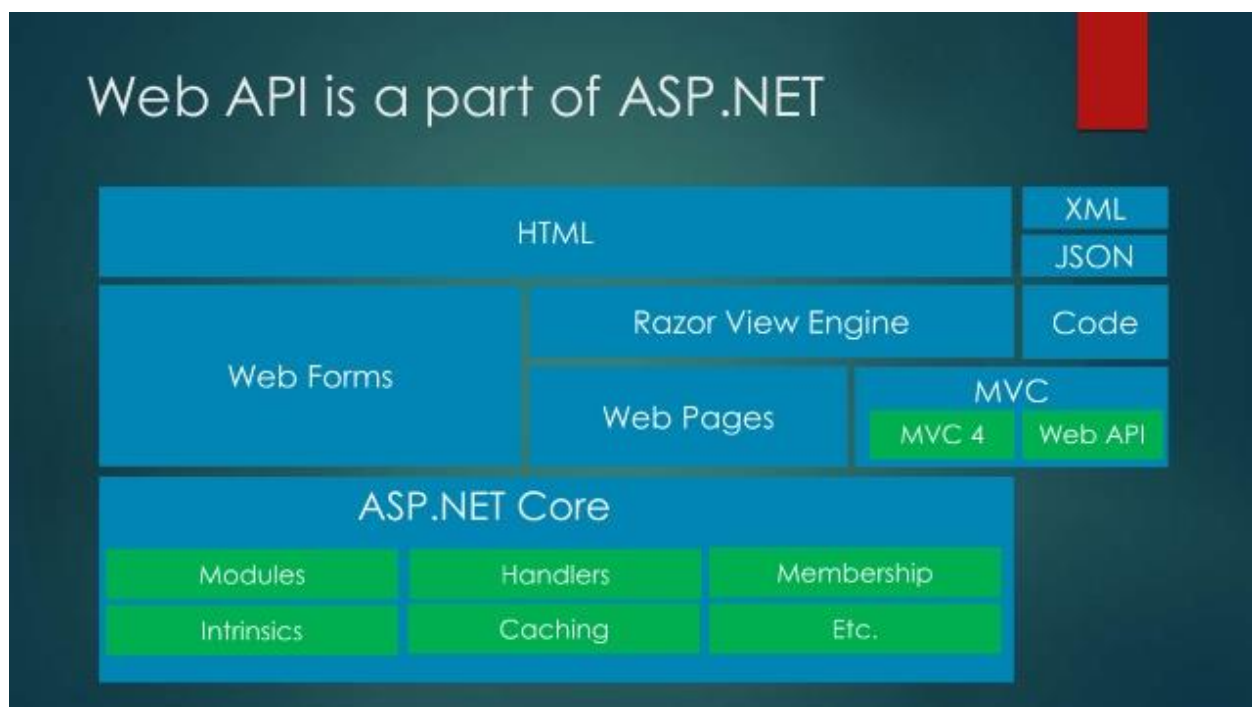


ASP.NET MVC 5 – Web API

เป็นพัฒนาการเพื่อลดจุดด้อยของ **Web Services** คือ ใช้เวลาเรียนรู้สั้น มีขนาดของ **Message** เหมาะสม **Overhead** น้อยลง โดยการสื่อสารจะใช้ **JSON** เป็นรูปแบบของการสื่อสาร ส่วนจุดด้อยของ **Web API** ก็คือ การขาดมาตรฐานในการกำหนดรูปแบบของ **JSON**

JSON ย่อมาจาก **Java Script Object Notation** ซึ่งสะดวกและใช้งานง่ายเมื่อใช้ร่วมกับ **jQuery** ในการพัฒนาเว็บ

ใน **ASP.NET MVC 4/5** จะมีเทคโนโลยีช่วยในการสร้าง **Web API** โดยอาศัย **Pattern MVC** ในลักษณะเดิม สามารถนำโค้ดเดิมมาใช้ใหม่ได้ โดยไม่ต้องเขียนใหม่แต่อย่างใด โดยเปลี่ยนแค่ **View** ที่ส่งออกไปจะเป็น **JSON** แทน **HTML**



Workshop Web API

<http://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api>