



Høgskolen i Telemark

Telemark University College

Department of Electrical Engineering, Information Technology and Cybernetics

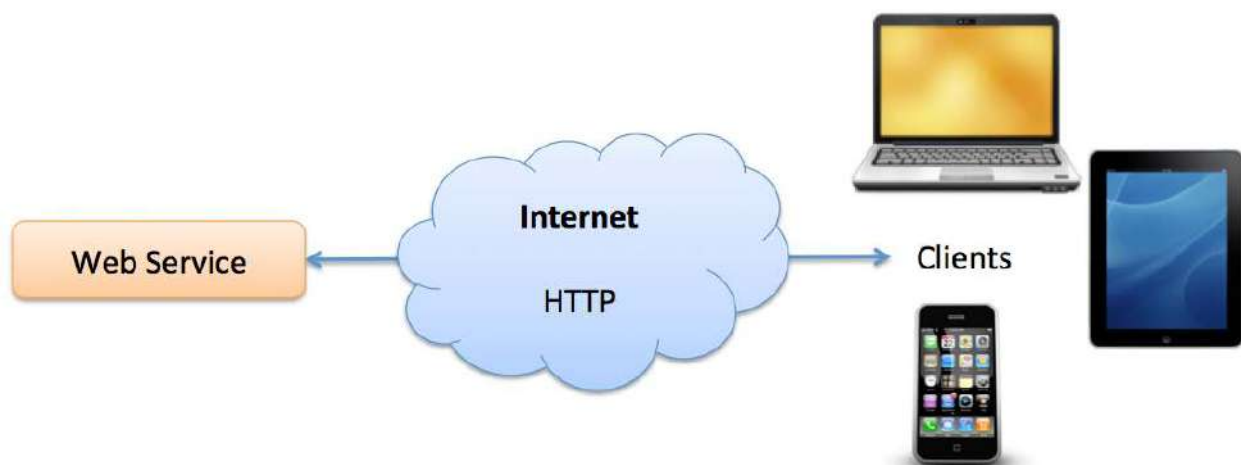


# Web Services

with Examples

---

Hans-Petter Halvorsen, 2014.03.01



# Table of Contents

1. Introduction .....	4
1.1. The Problem.....	4
1.2. The Solution .....	5
1.3. Web Services Overview.....	5
1.4. REST Web Services .....	9
2. Creating Web Services with Visual Studio .....	10
3. ASMX Web Service.....	11
3.1. Create a Web Method .....	14
4. IIS Web Server.....	15
5. Web Service Client in Visual Studio .....	16
5.1. Windows Forms .....	16
6. Get Data from a Temperature Device.....	17
6.1. TC-01 Thermocouple Device .....	17
7. Web Service with Data from a Database .....	18
7.1. Create Database.....	18
7.2. Data Access Tier .....	20
7.3. Create the Web Service .....	23
7.4. Using the Web Service .....	27
8. Web Services in LabVIEW.....	30
8.1. Web Service .....	30
8.2. LabVIEW Client.....	33

8.2.1. Temperature Conversion .....	33
8.2.2. Weather Station Example .....	36
9. Data Dashboard for LabVIEW .....	44
9.1. Example.....	<b>Error! Bookmark not defined.</b>
10. Python.....	47
11. MATLAB .....	48

# 1. Introduction

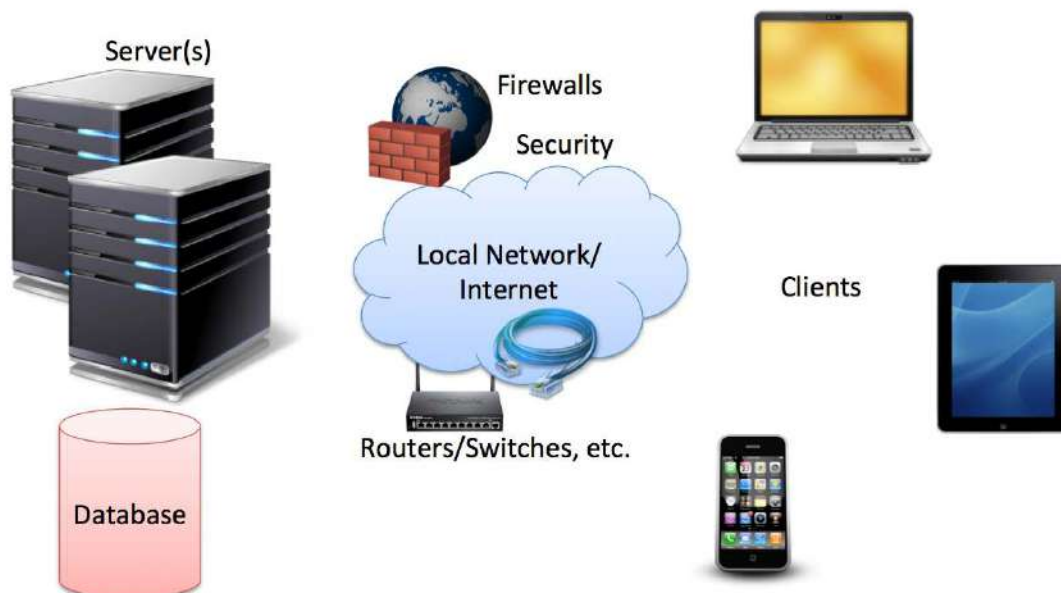
In addition to this written Tutorial, you will find additional resources here:

<http://home.hit.no/~hansha/?tutorial=webservice>

## 1.1. The Problem

Our problem is: How to Share Data between Devices in a Network?

How to Share Data between Devices in a Network?



Direct Connection between the Database and the Clients that need the Data is normally not possible, due to security, compatibility issues, etc. (Firewalls, Hacker Attacks, etc.).

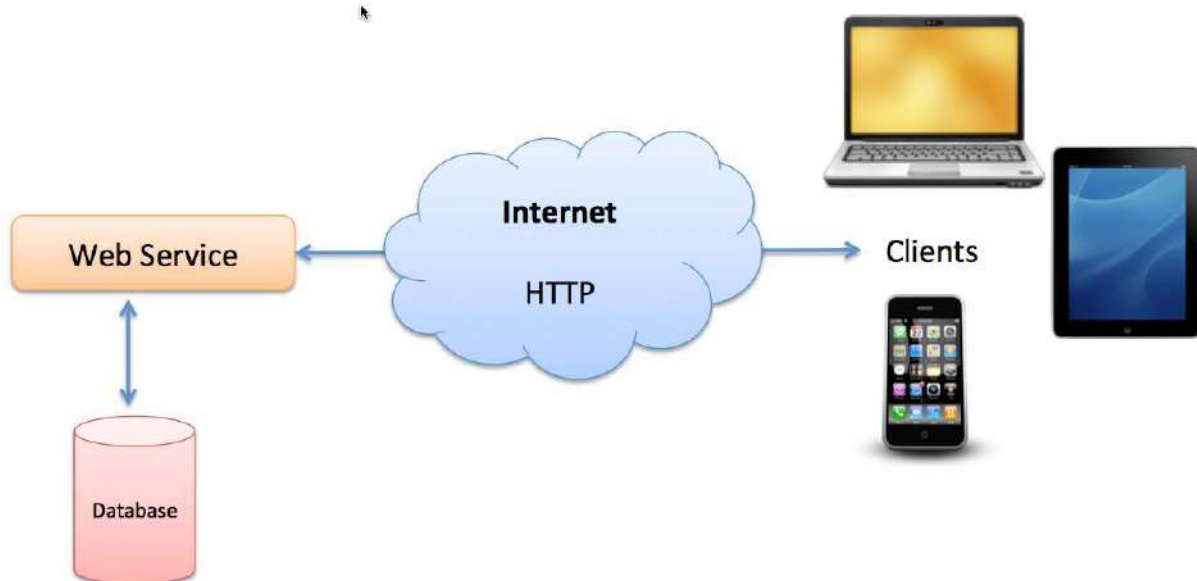
Direct Connection in a Local Network (behind the Firewall) is normally OK – but not over the Internet.



## 1.2. The Solution

Solution: Web Service. Web Services uses standard web protocols like HTTP, etc.

HTTP is supported by all Web Browser, Servers and many Programming Languages



Today Web Services have been very popular. A Web service is a method of communications between two devices over the World Wide Web, and makes it easy to share data over a network or the internet.

## 1.3. Web Services Overview

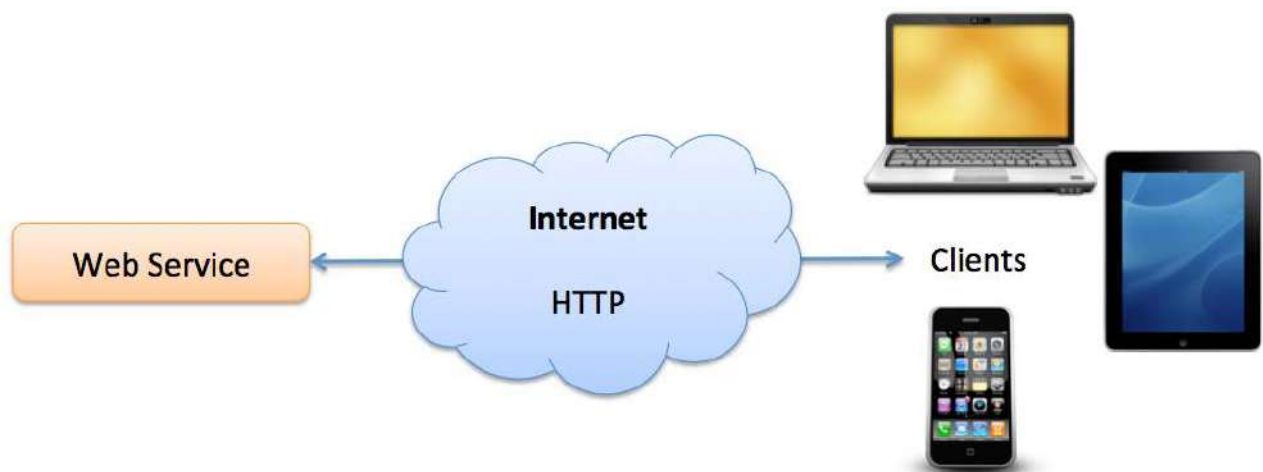
A Web Service is

- A Web API

- A Standard defined by W3C
- Cross-platform and Platform-independent Communication
- Distributed Application Development

Web Services can be implemented and used in most Programming Languages (C#/ASP.NET, PHP, LabVIEW, Objective-C, Java, etc.)

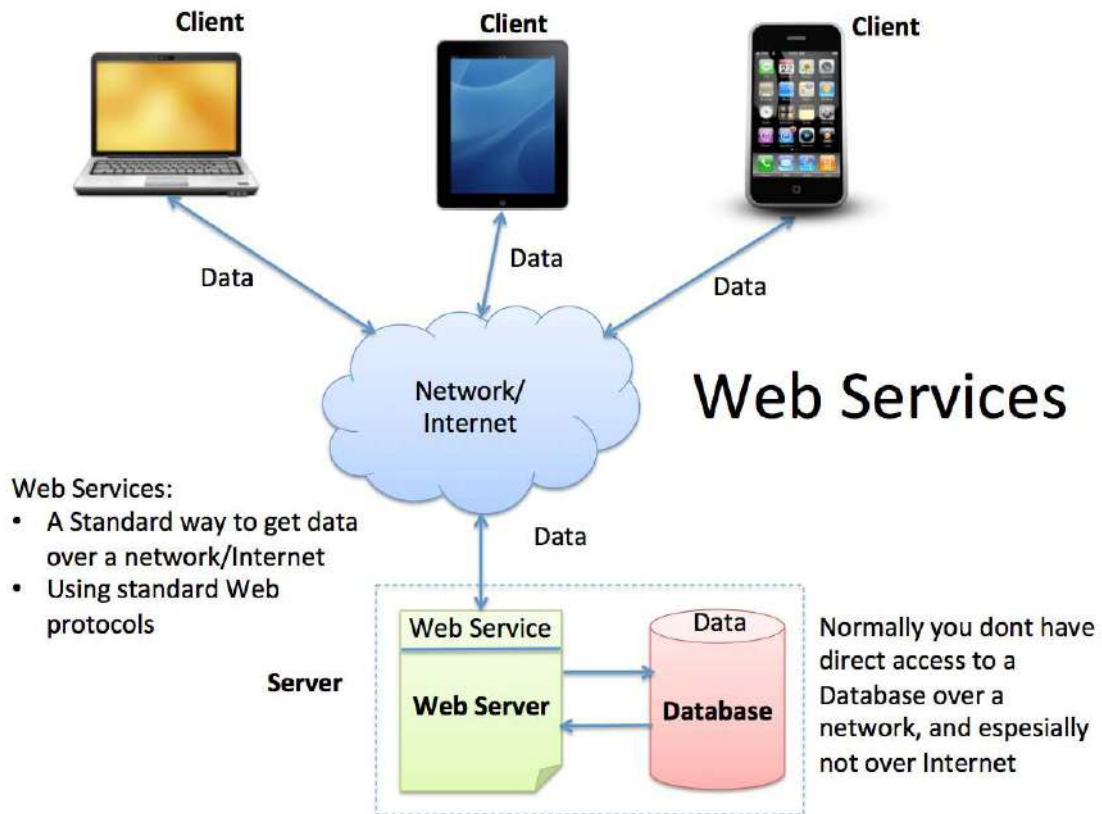
Web Services uses standard Web technology (Web protocols) such as HTTP, REST, SOAP, XML, WSDL, JSON, etc.



Web Services technology used in Web Services:

- HTTP - Hypertext Transfer Protocol
- XML – Extensible Markup Language
- WSDL - Web Services Description Language
- SOAP - Simple Object Access Protocol
- REST - Representational State Transfer

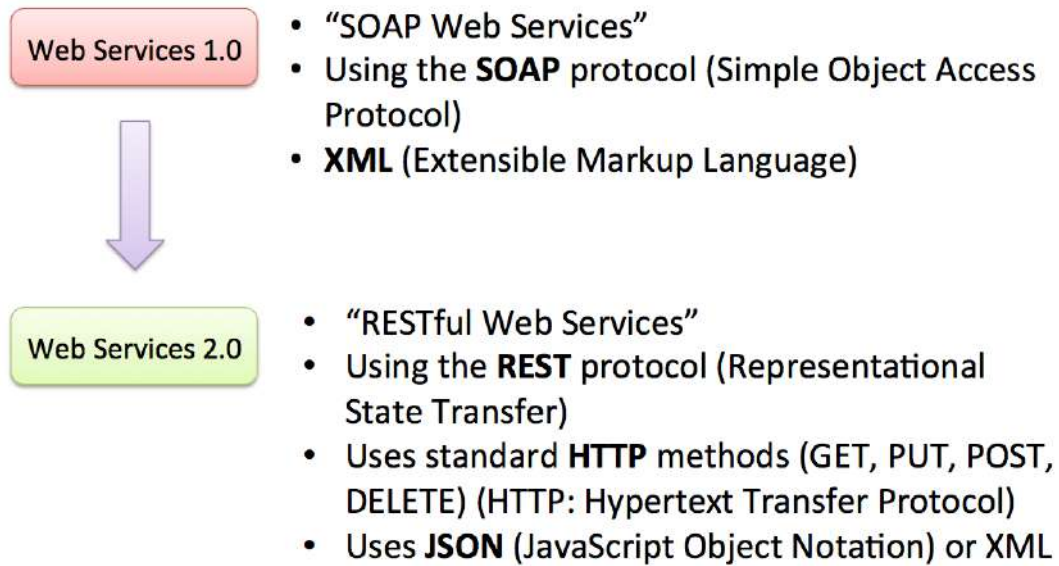
A Web Service is typically deployed on a web server, similar as ordinary web pages.



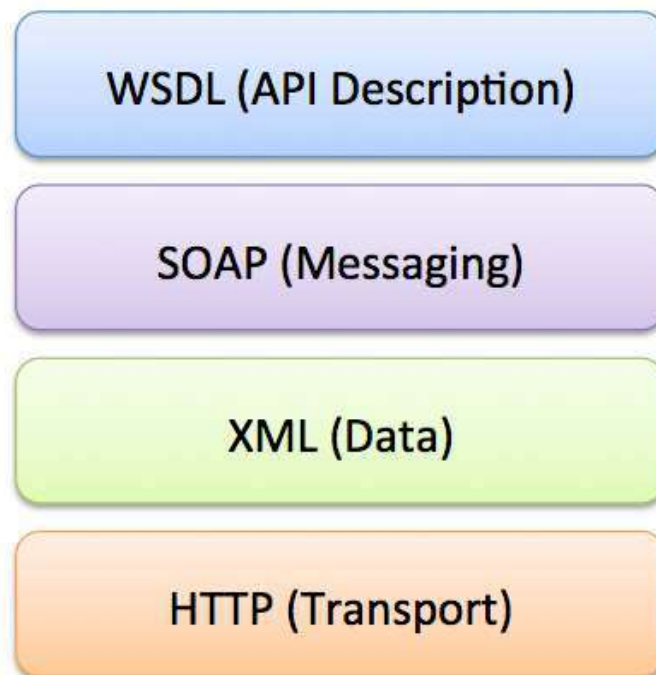
We have 2 different types of Web Services:

- Web Services 1.0: SOAP Web Services
  - “Complex”
- Web Services 2.0: REST Web Services
  - Lightweight and Flexible
  - A new and simpler version of WS
  - All major WS on the Internet today use REST

Below we summarize Web Services 1.0 vs. 2.0.



Below we see the different “layers” a “SOAP Web Service” consists of.



### XML:

XML stands for eXtensible Markup Language. XML is designed to transport and store data.

Below we see an XML document example.

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
```



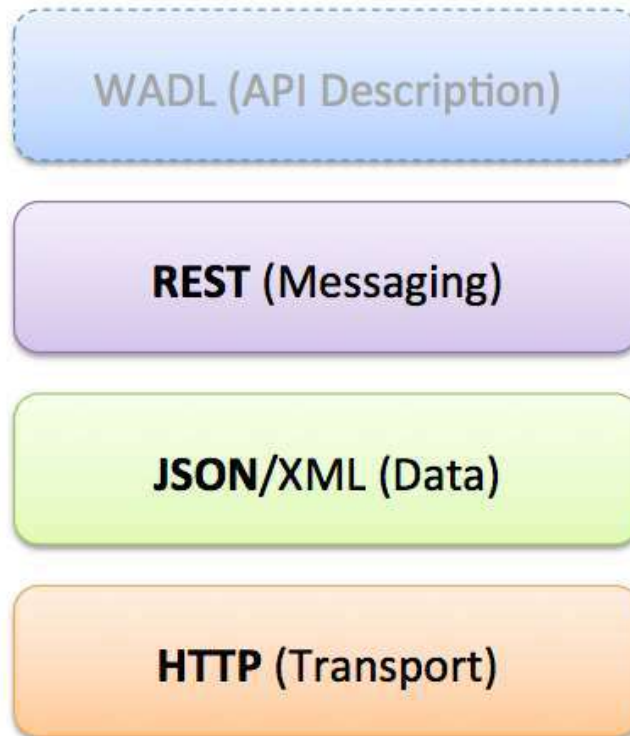
---

</note>

---

## 1.4. REST Web Services

Below we see the different “layers” a “REST Web Service” consists of.

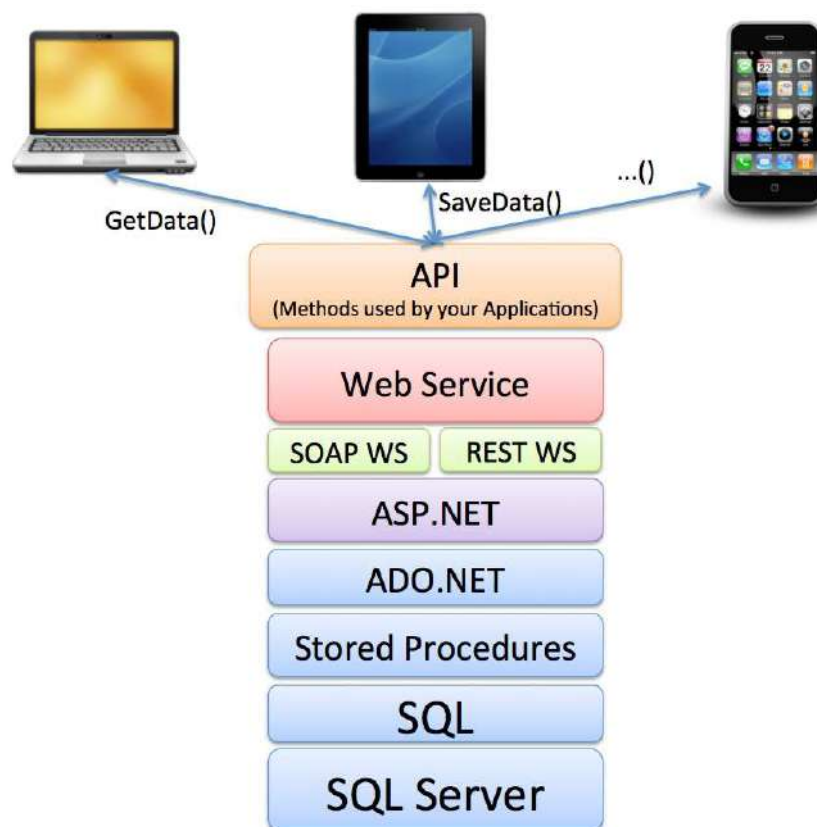


## 2. Creating Web Services with Visual Studio

Visual Studio has powerful features for creating Web Services.

3 ways to do it:

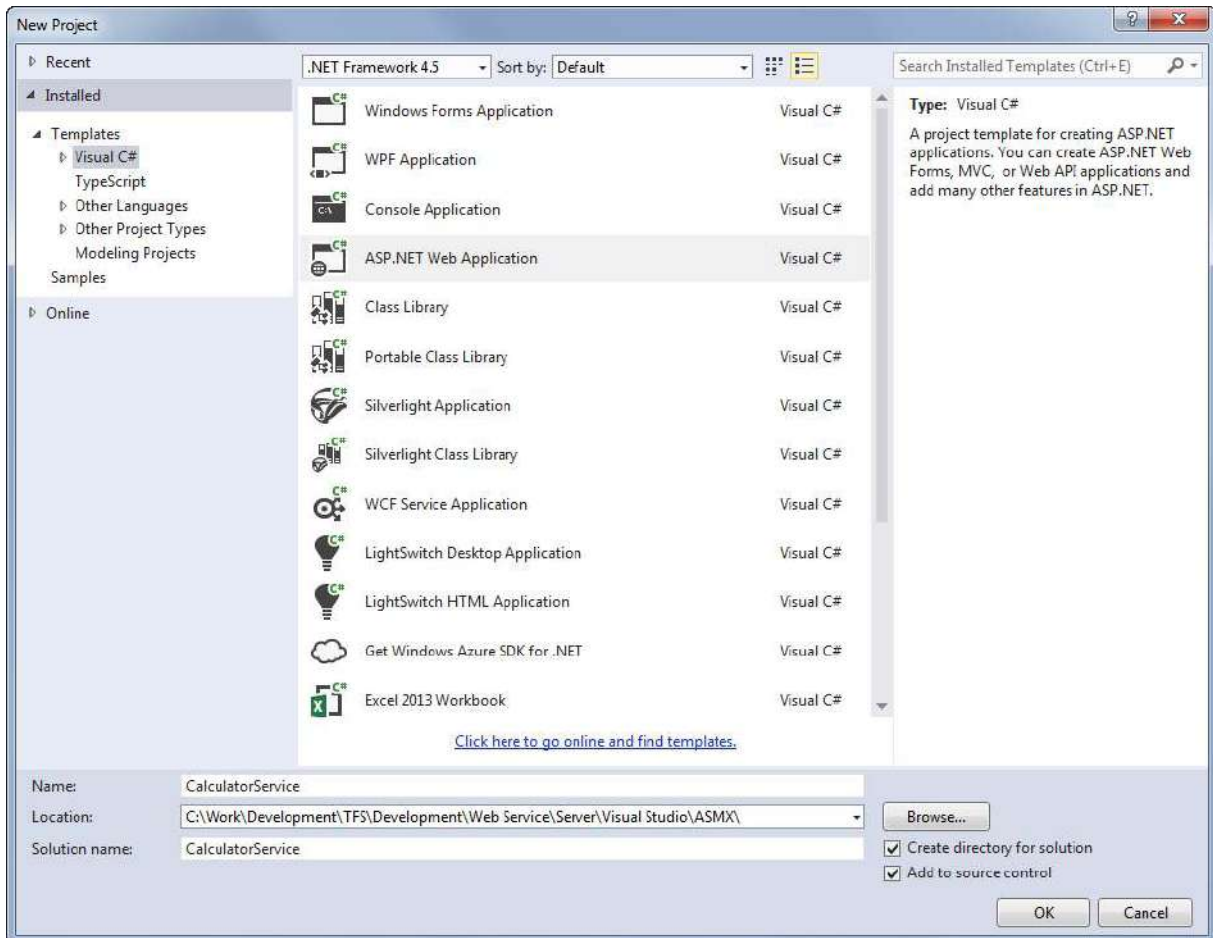
- ASMX Web Service (Traditional Web Service using SOAP)
- WCF Service: A general approach used to create all kind of communication including web services, both SOAP and REST
- ASP.NET Web API (The modern Web Service using REST, Web 2.0)



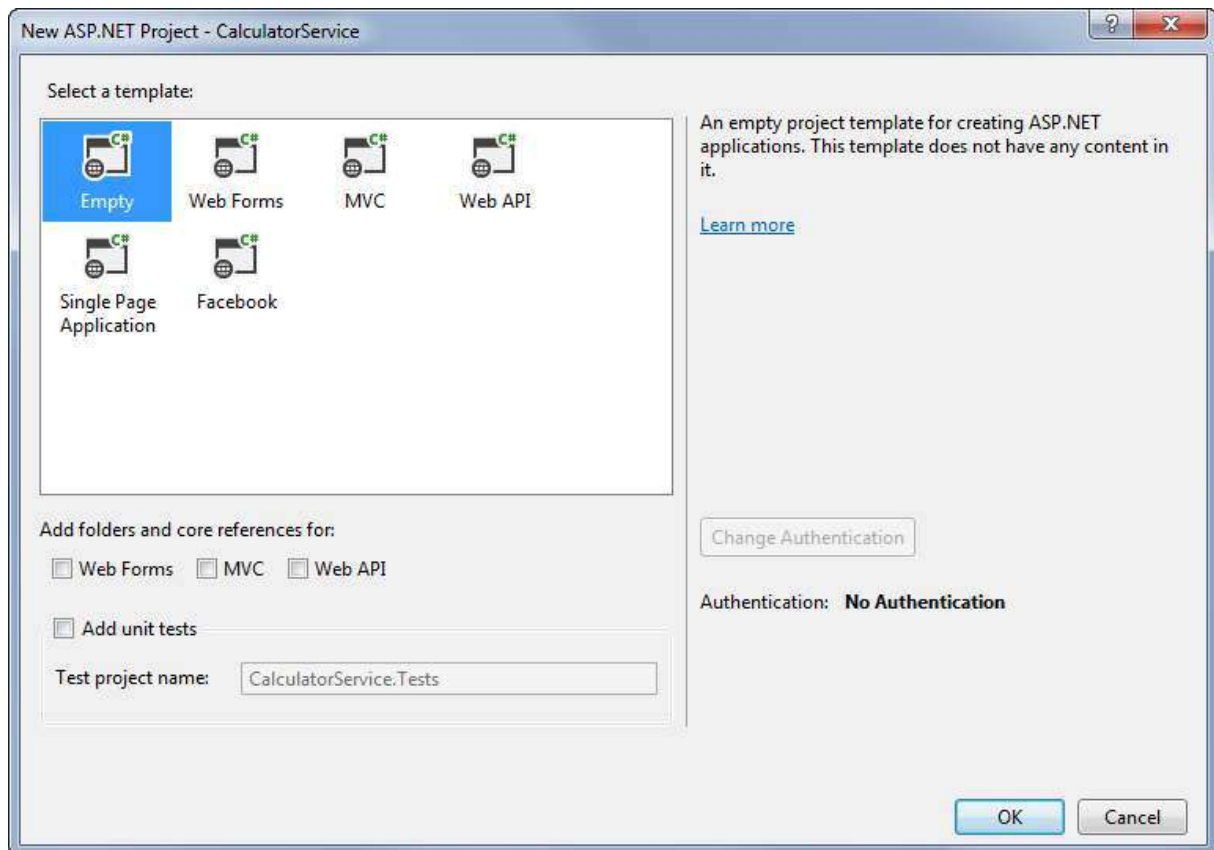
# 3. ASMX Web Service

In this chapter we will go through the steps in order to create a ASMX/SOAP Web Service in Visual Studio.

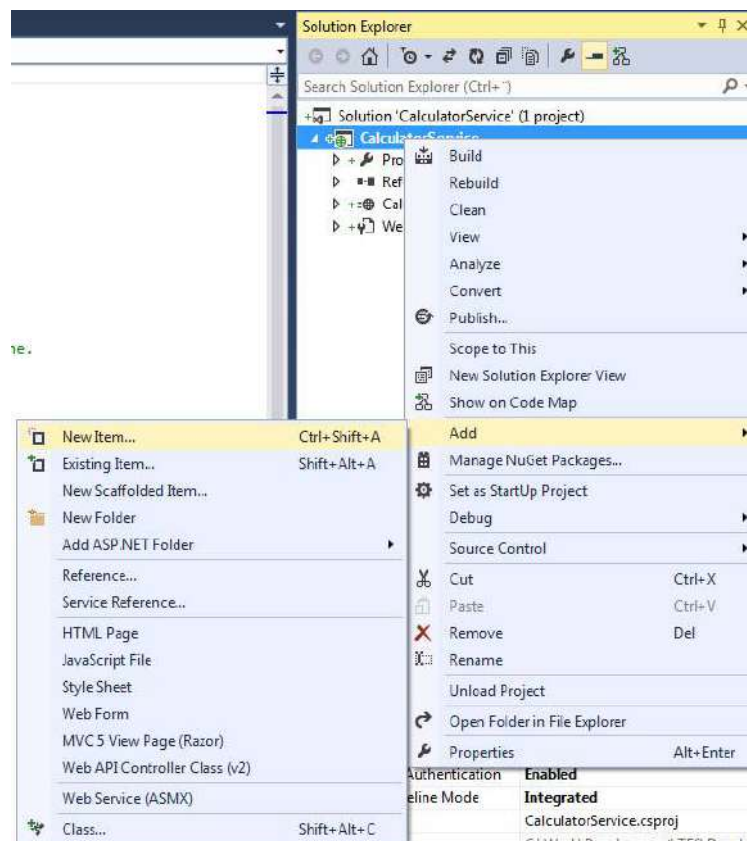
New Project:



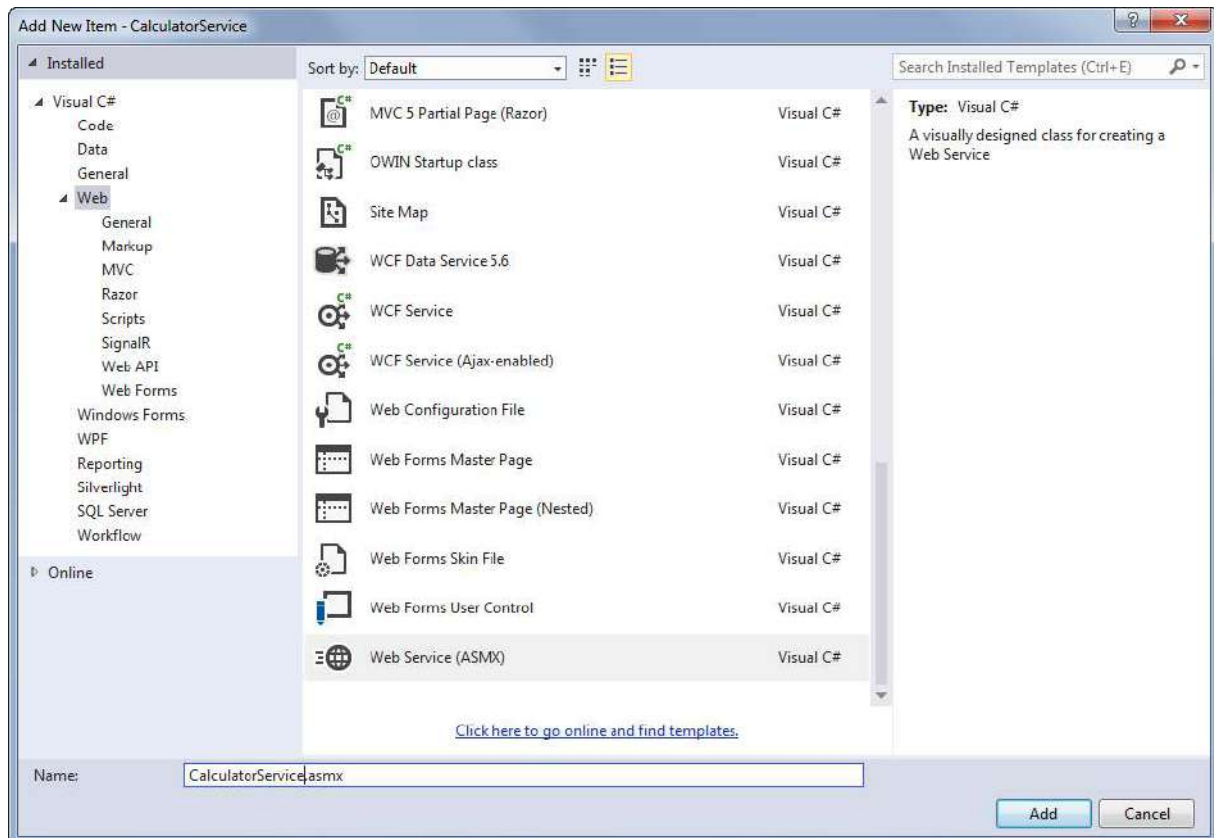
Select the "Empty" template:



Select "Add New Item":



Select the “Web Service (ASMX)” item:



The default template looks like this:

```

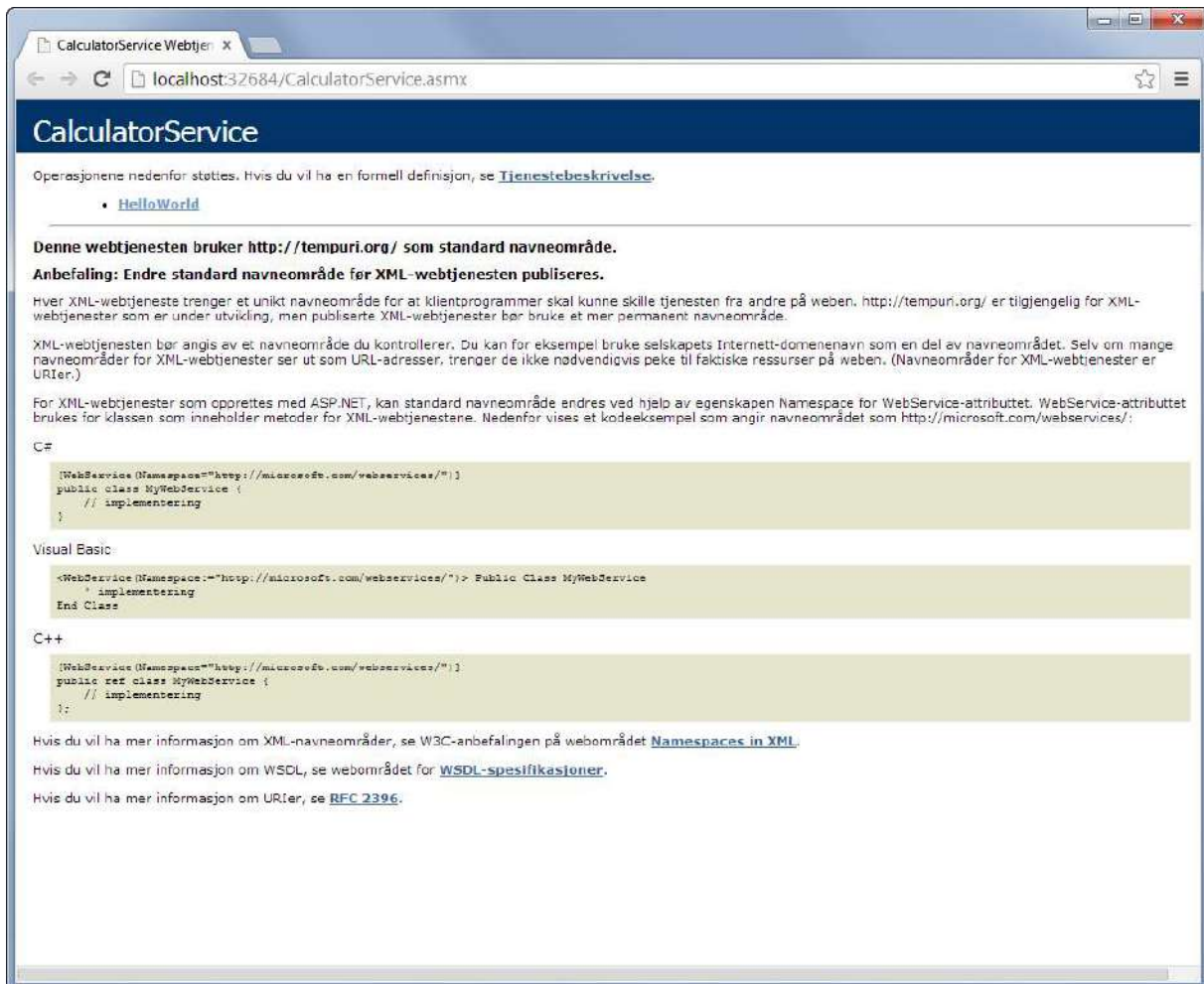
CalculatorService.asmx.cs
CalculatorService.CalculatorService
HelloWorld()

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

namespace CalculatorService
{
    /// <summary>
    /// Summary description for CalculatorService
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
    // [System.Web.Script.Services.ScriptService]
    public class CalculatorService : System.Web.Services.WebService
    {
        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }
    }
}

```

## Test the built-in HelloWorld() method – F5



### 3.1. Create a Web Method

Create a method that gets the current date and time:

```
[WebMethod]
public string GetTime()
{
    DateTime time = DateTime.Now;
    string dateformat = "yyyy.MM.dd HH:MM:ss";

    return time.ToString(dateformat);
}
```

...

# 4. IIS Web Server

IIS – Internet Information Server

# 5. Web Service Client in Visual Studio

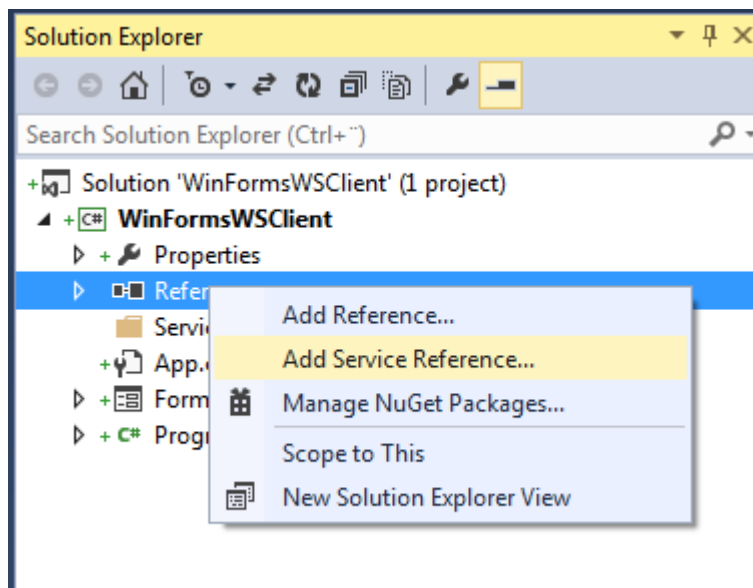
Walkthrough: Connecting to Data in a Web Service (Windows Forms):

<http://msdn.microsoft.com/en-us/library/ms171891.aspx>

## 5.1. Windows Forms

Create a New WinForm Project.

Select “Add Service Reference”





# 6. Get Data from a Temperature Device

In this chapter we will publish temperature data from a temperature device via a Web Service.

## 6.1. TC-01 Thermocouple Device

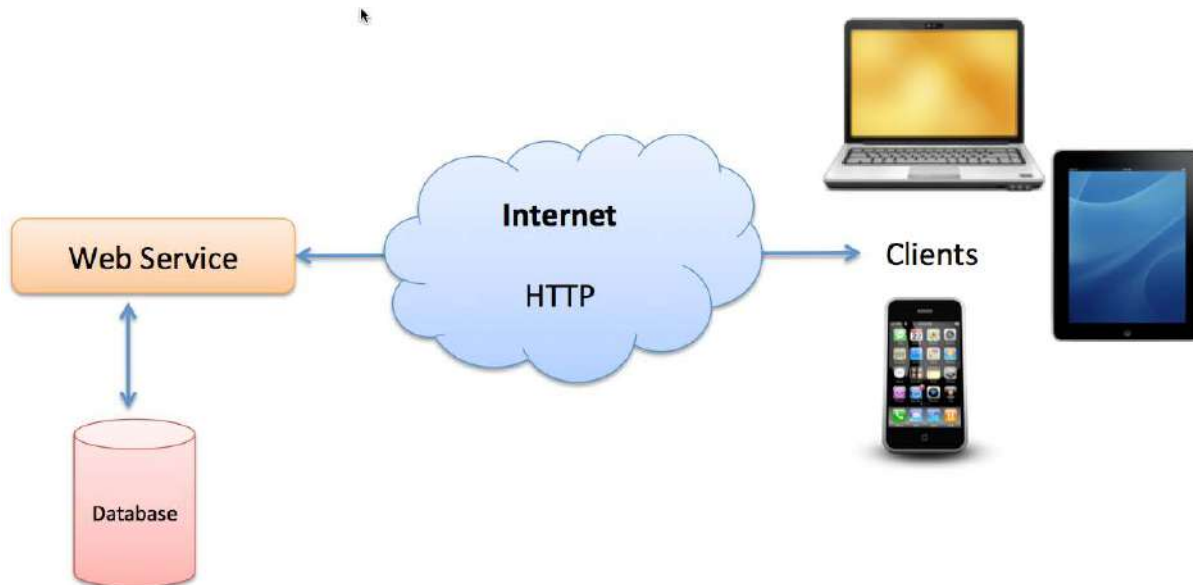
Below we see the TC-01 Thermocouple device from National Instruments:



TC-01 Web Site: <http://sine.ni.com/nips/cds/view/p/lang/no/nid/208177>

# 7. Web Service with Data from a Database

The main purpose with a Web Service is to share data from a database between devices in a network.



Direct Connection between the Database and the Clients that need the Data is normally not possible, due to security, compatibility issues, etc. (Firewalls, Hacker Attacks, etc.).

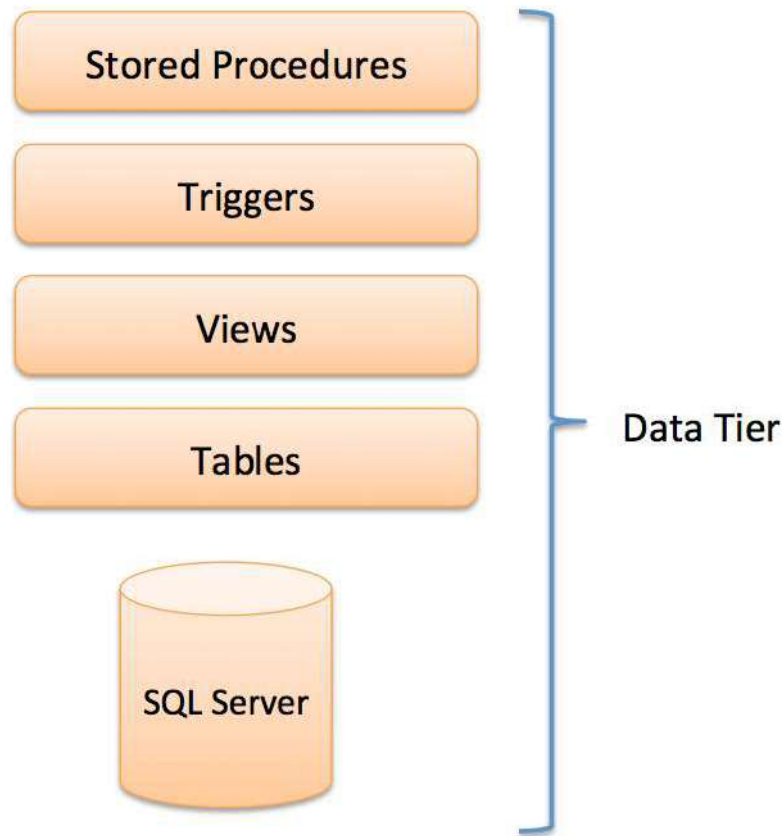
Direct Connection in a Local Network (behind the Firewall) is normally OK – but not over the Internet.

## 7.1. Create Database

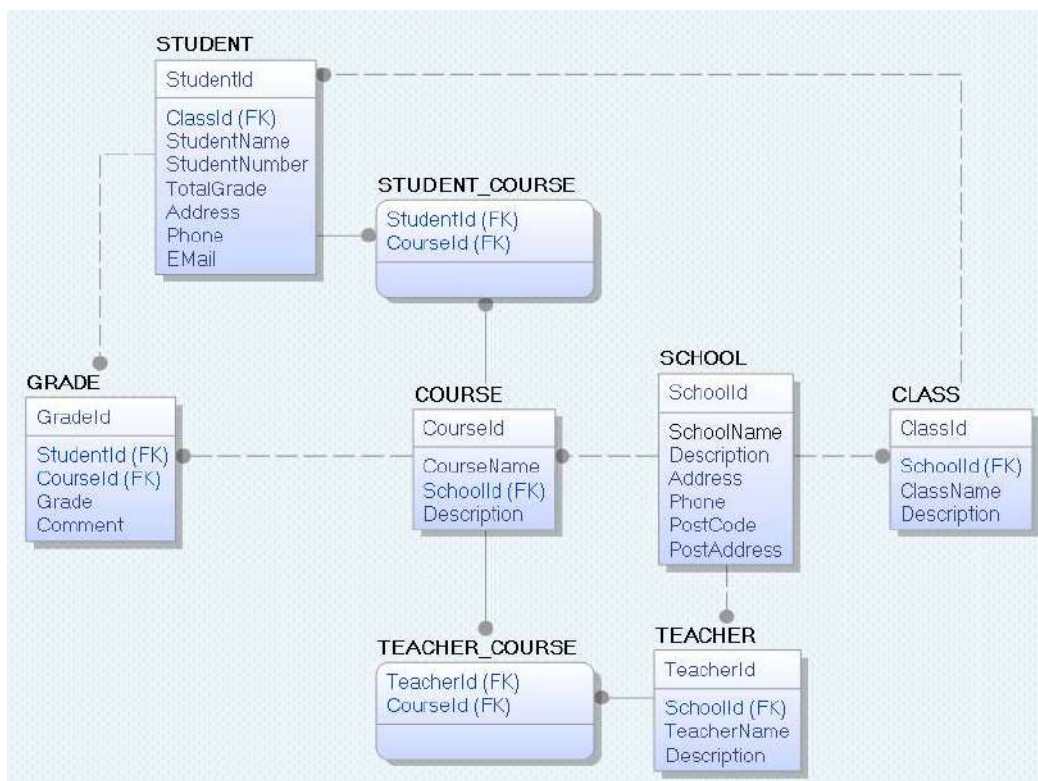
We are going to create the Database / Data Tier, including:

- Tables
- Views
- Stored Procedures

- etc.

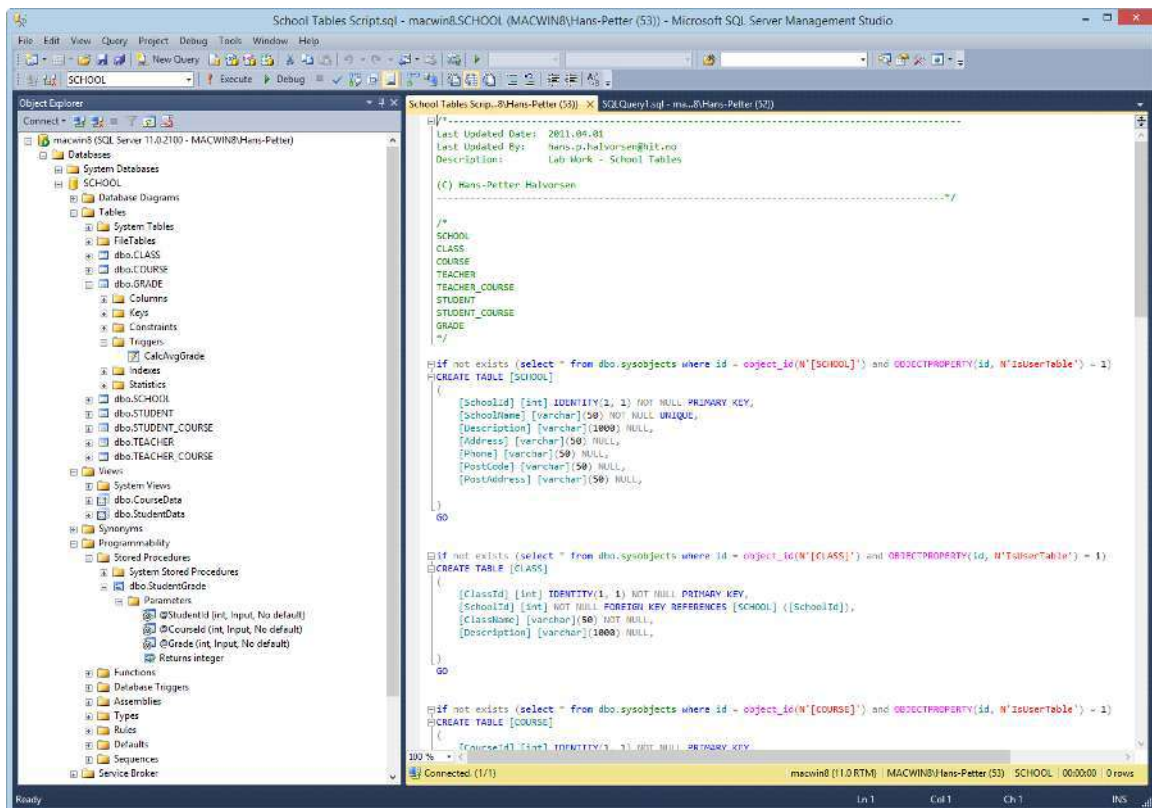


We are going to create the following example database:



The database is designed using Erwin.

The Database is created and implemented using SQL Server Management Studio



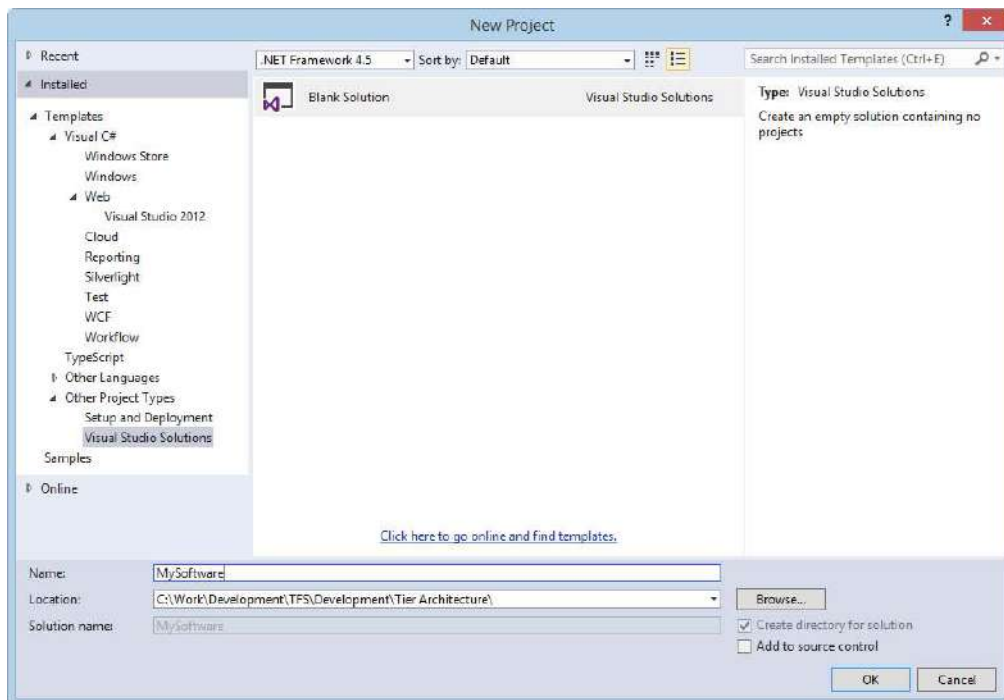
You may download a Zip File with Tables, Views, Stored Procedures, etc. in order to create the Data Tier in SQL Server.

<http://home.hit.no/~hansha/?tutorial=sq>

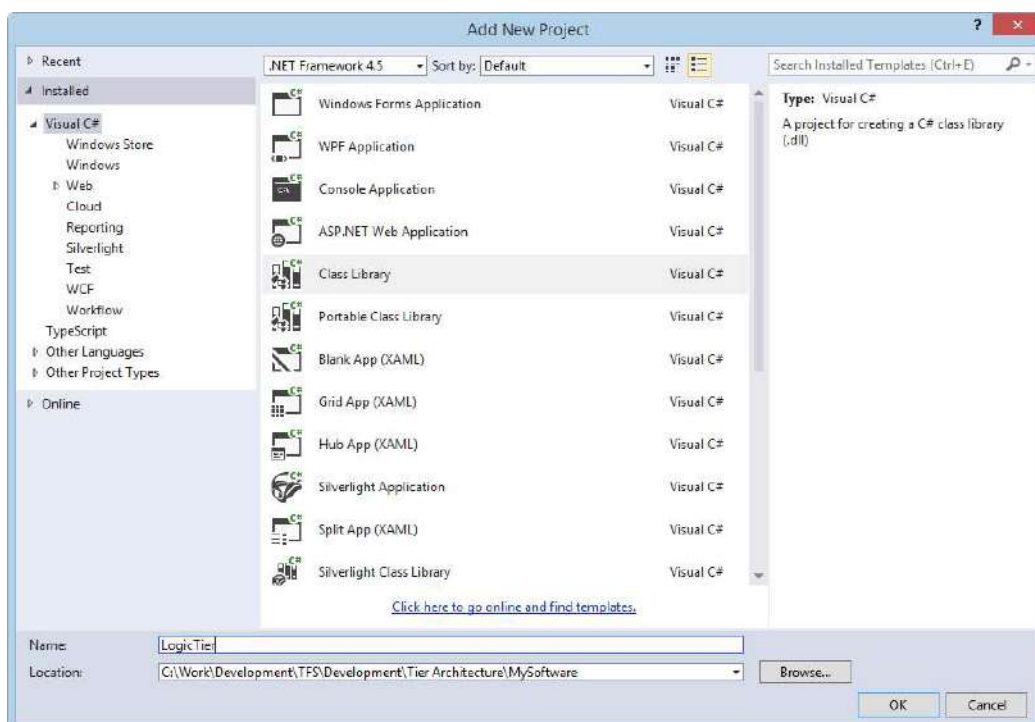
## 7.2. Data Access Tier

We will create a project in Visual Studio where we create alle the logic/code that deal with the communication with the database.

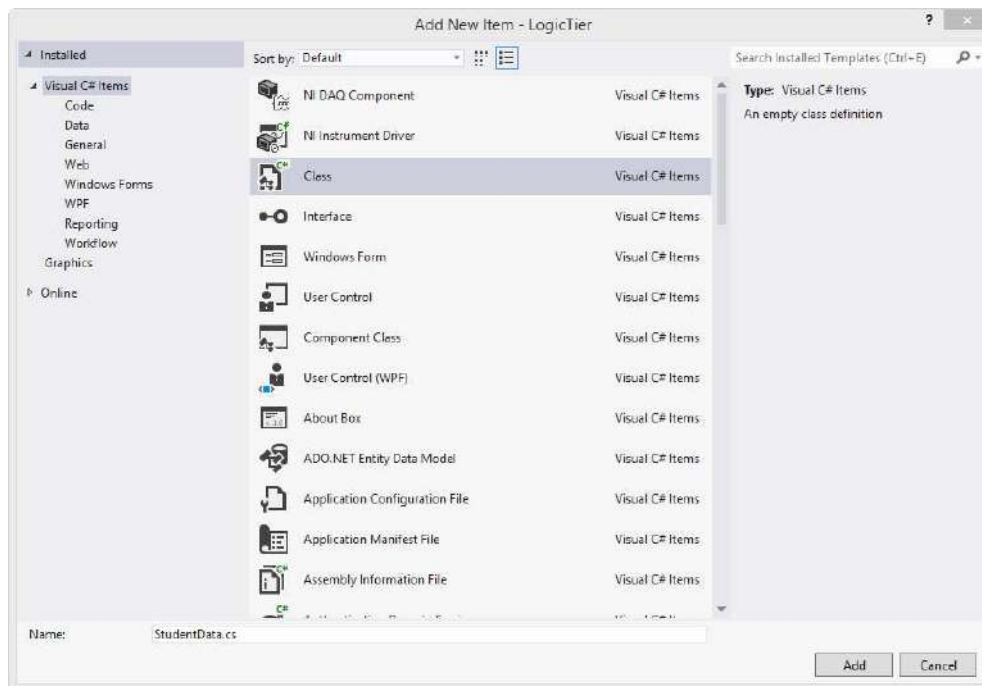
Create an empty Solution in Visual Studio:



Add Project for Data Access Tier:



Add a New Class to the Project:



Create the Code, e.g., like this:

```

StudentData.cs  + X
Tuc.School.LogicTier.StudentData  -  GetStudentDB(string connectionString)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Data;

namespace Tuc.School.LogicTier
{
    public class StudentData
    {
        public DataSet GetStudentDB(string connectionString)
        {
            string selectSQL = "select StudentName, StudentNumber, SchoolName, ClassName, Grade from StudentData order by StudentName";

            // Define the ADO.NET objects.
            SqlConnection con = new SqlConnection(connectionString);
            SqlDataAdapter da = new SqlDataAdapter(selectSQL, con);

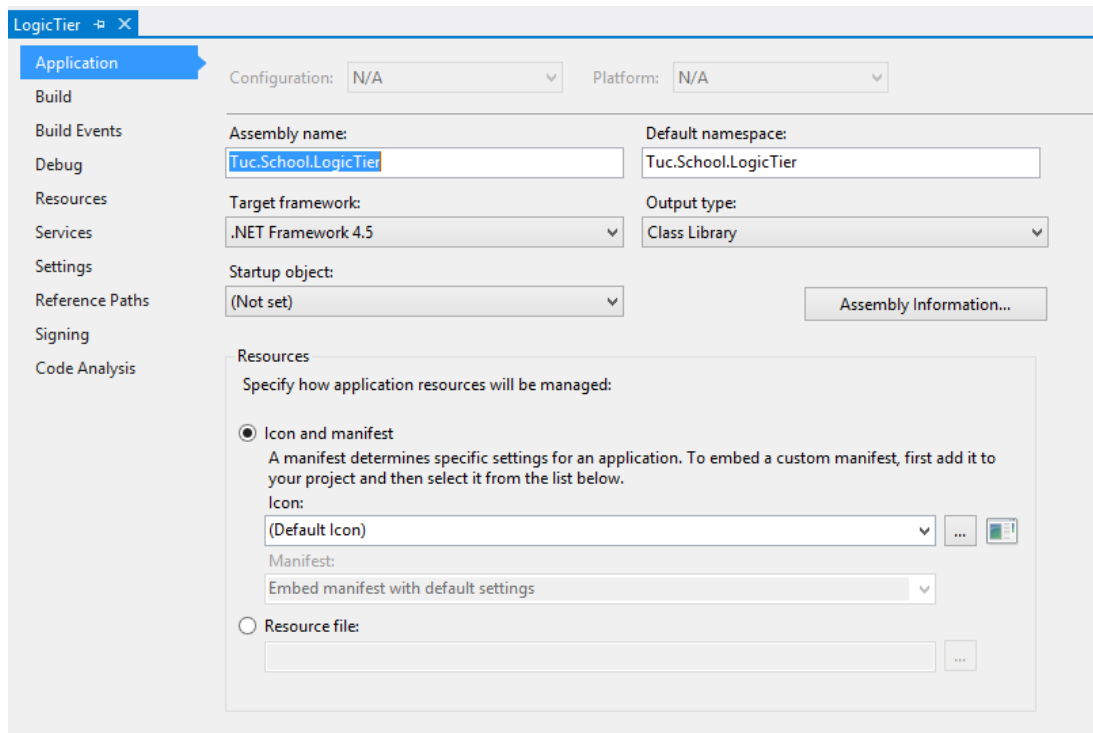
            DataSet ds = new DataSet();
            da.Fill(ds);

            return ds;
        }
    }
}

```

Make sure to import the necessary Name Spaces. You should also create a proper Name space for the Class file.

Create a proper name for the Assembly (.dll File):

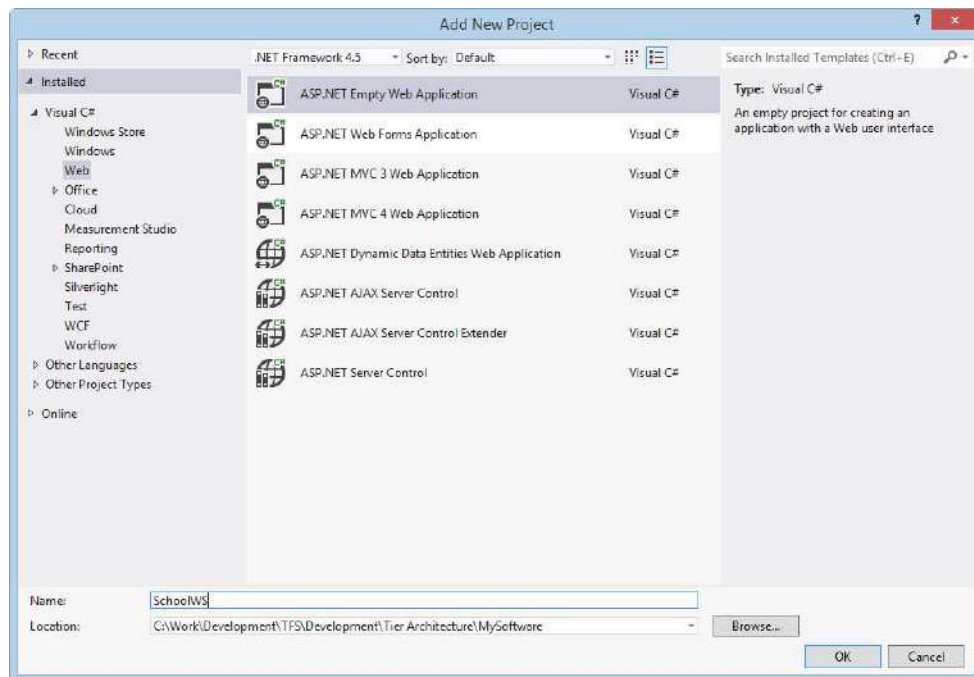


Then Build your Project (hopefully with no errors). This will be the Assembly for your Data Access (Logic) Tier, that can be imported and used in other projects. Create once – use it many times!!

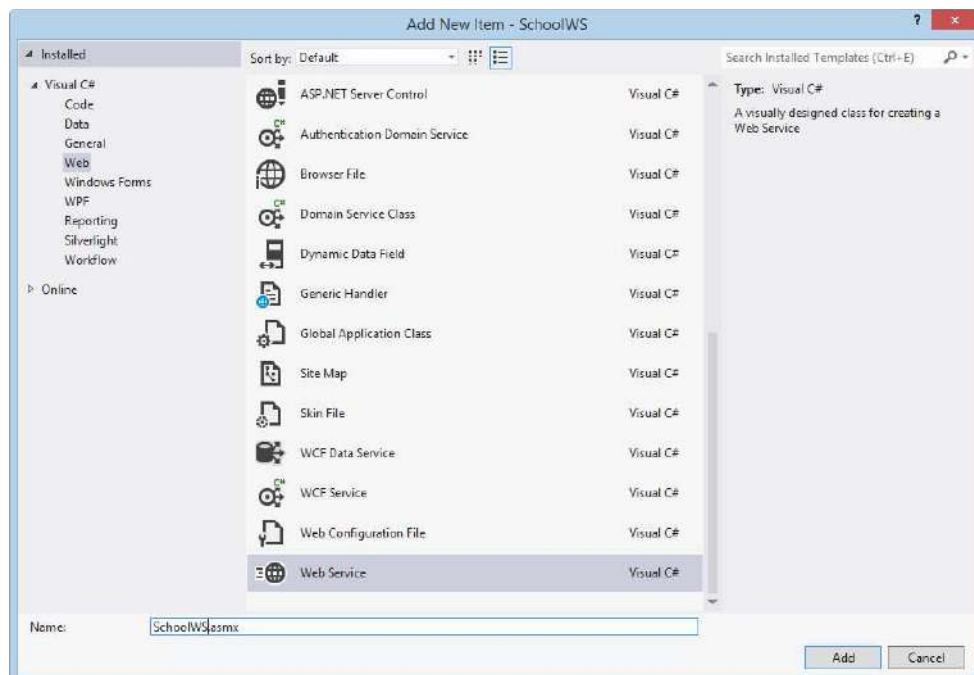
## 7.3. Create the Web Service

Now we will create the Web Service that is the connection between the Data Access (Logic) Tier and the Device that is going to communicate with the database.

Create an ASP.NET Project:



Add Web Service:



Web Service Code:



```

SchoolWS.asmx.cs
SchoolWS.SchoolWS
connectionString

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

using System.Data;
using System.Web.Configuration;
using Tuc.School.LogicTier;

namespace SchoolWS
{
    /// <summary>
    /// Summary description for SchoolWS
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
    // [System.Web.Script.Services.ScriptService]
    public class SchoolWS : System.Web.Services.WebService
    {
        private string connectionString = WebConfigurationManager.ConnectionStrings["SCHOOLConnectionString"].ConnectionString;

        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }

        [WebMethod]
        public DataSet GetStudent()
        {
            StudentData studentData = new StudentData();

            return studentData.GetStudentDB(connectionString);
        }
    }
}

```

The Database ConnectionString is located in Web.config:

```

Web.config
SchoolWS.asmx.cs
StudentInformation

<?xml version="1.0"?>

<!--
For more information on how to configure your ASP.NET application, please visit
http://go.microsoft.com/fwlink/?LinkID=169433
-->

<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.5" />
    <httpRuntime targetFramework="4.5" />
  </system.web>

  <connectionStrings>
    <add name="SCHOOLConnectionString" connectionString="Data Source=macwin8;Initial Catalog=SCHOOL;Persist Security Info=True;User ID=sa;Password="
      providerName="System.Data.SqlClient" />
  </connectionStrings>

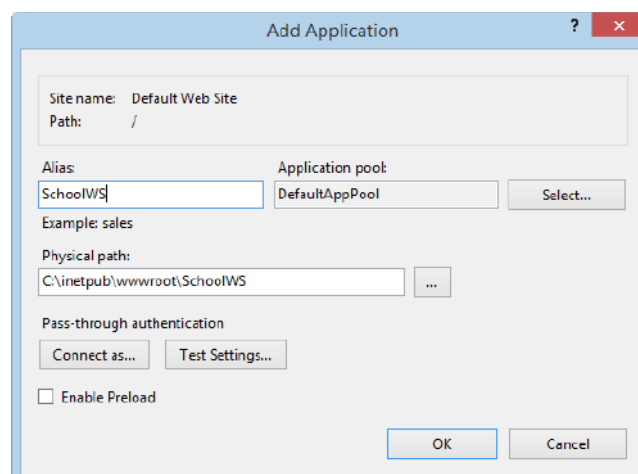
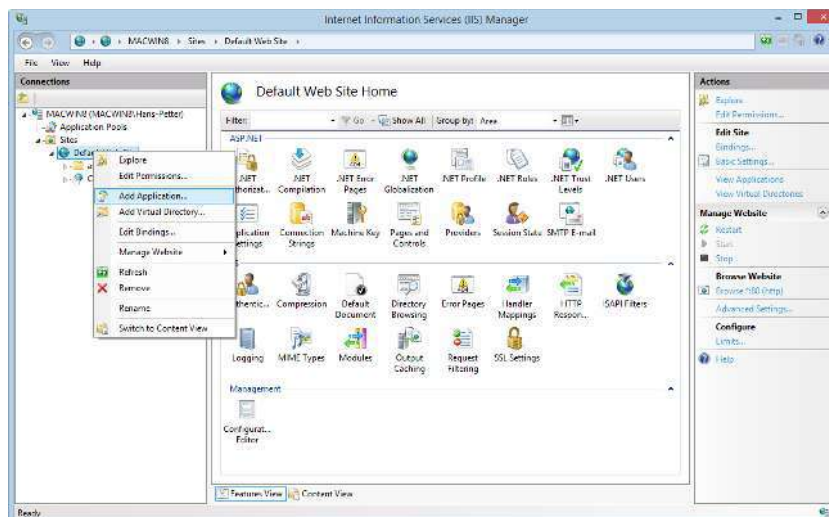
</configuration>

```

Test the Web Service:



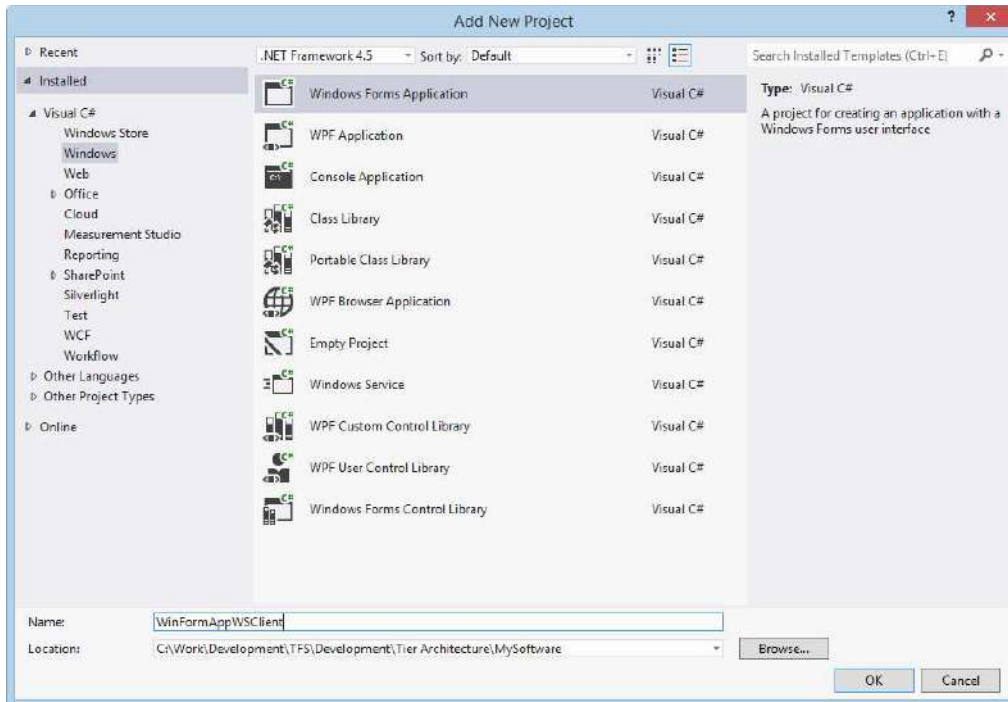
Deploy/Publish Web Service to IIS. Copy Web Service Files to default IIS Directory:  
C:\inetpub\wwwroot.



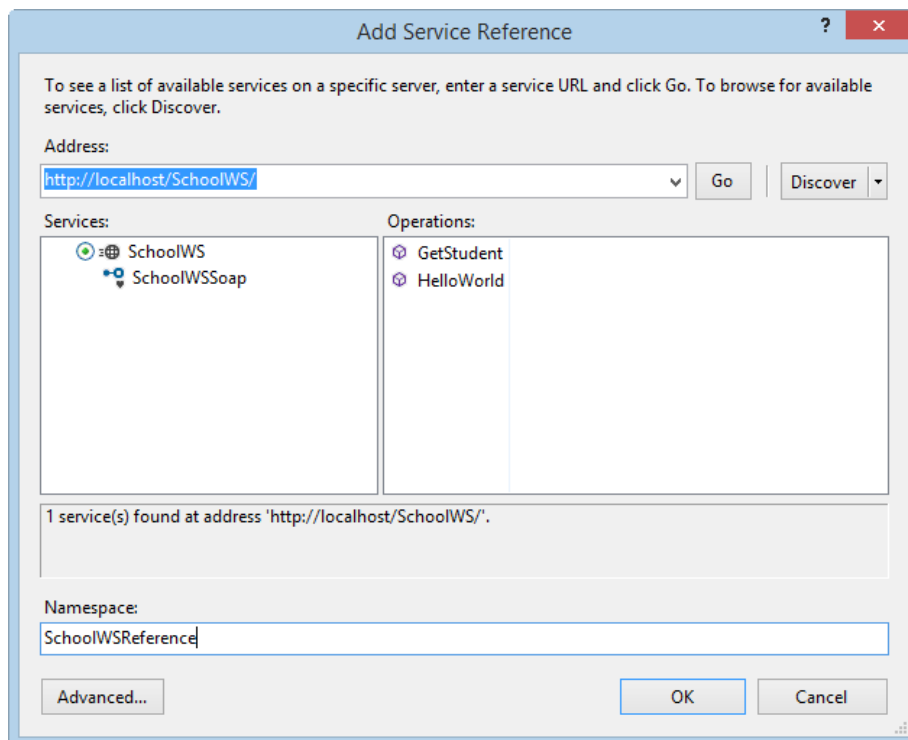
## 7.4. Using the Web Service

We will use Web Service in a WinForm Application.

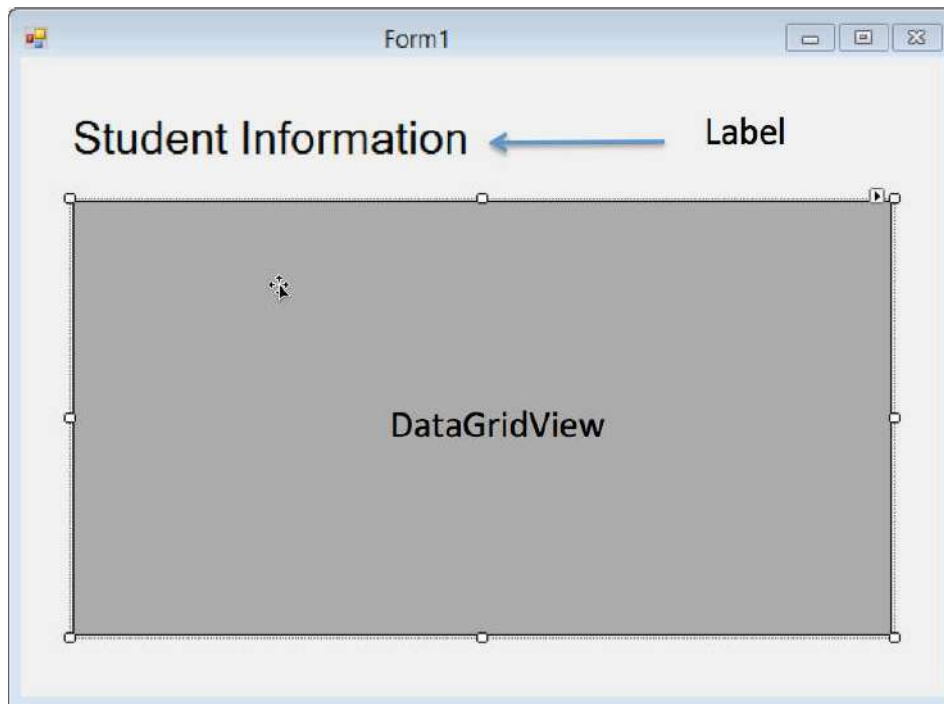
Create New WinForm Project:



Add Web Service Reference:



Create GUI:



Create Code:

```
FormWSClient.cs -> FormWSClient.cs [Design]
WinFormAppWSClient.FormWSClient -> FormWSClient_Load(object sender, EventArgs e)

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WinFormAppWSClient
{
    public partial class FormWSClient : Form
    {
        public FormWSClient()
        {
            InitializeComponent();
        }

        private void FormWSClient_Load(object sender, EventArgs e)
        {
            FillStudentGrid();
        }

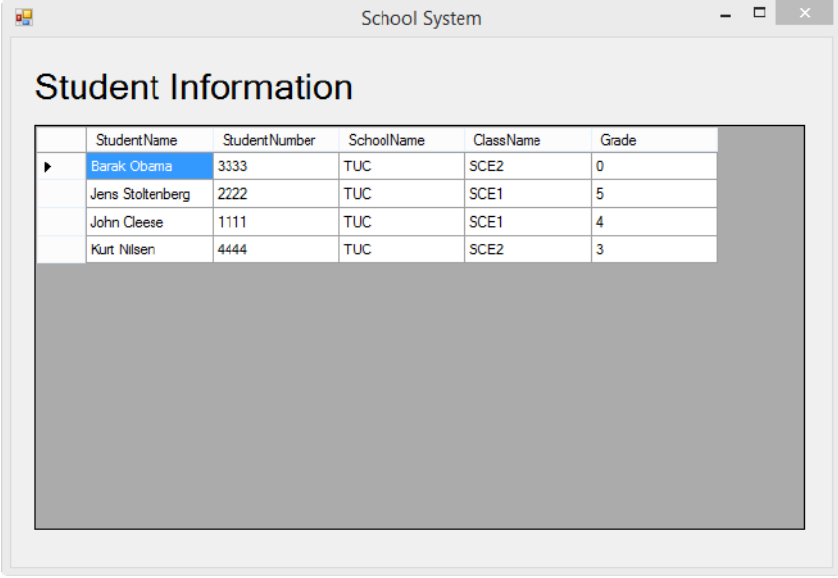
        private void FillStudentGrid()
        {
            DataSet ds = new DataSet();

            SchoolWSReference.SchoolWSSoapClient schoolWs = new SchoolWSReference.SchoolWSSoapClient();

            ds = schoolWs.GetStudent();

            dataGridViewStudentInformation.DataSource = ds.Tables[0];
        }
    }
}
```

Test it:



The screenshot shows a web application window titled "School System". The main content area is titled "Student Information" and displays a table with the following data:

	StudentName	StudentNumber	SchoolName	ClassName	Grade
▶	Barak Obama	3333	TUC	SCE2	0
	Jens Stoltenberg	2222	TUC	SCE1	5
	John Cleese	1111	TUC	SCE1	4
	Kurt Nilsen	4444	TUC	SCE2	3

It works!

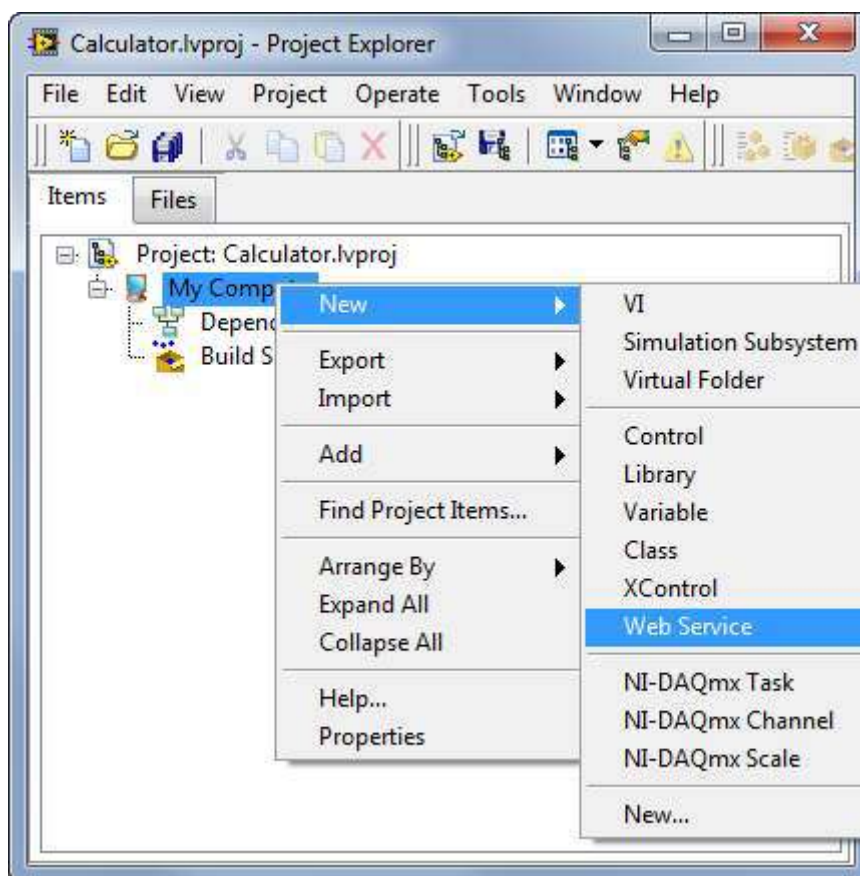
# 8. Web Services in LabVIEW

## 8.1. Web Service

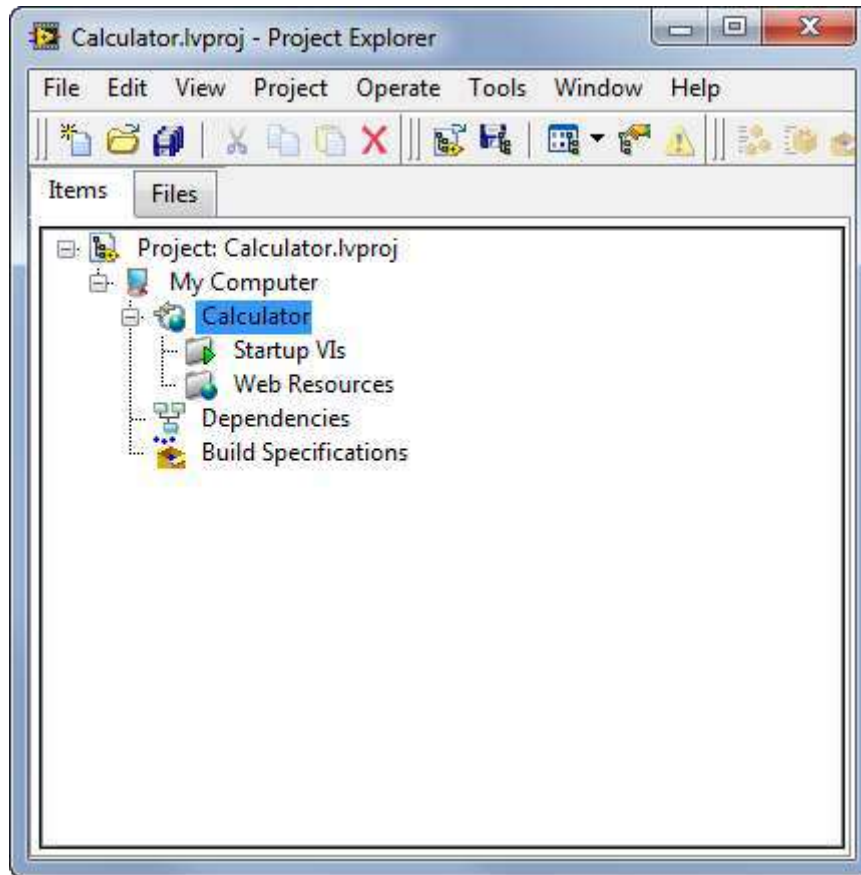
Tutorial: Creating and Accessing a LabVIEW Web Service:

[http://zone.ni.com/reference/en-XX/help/371361K-01/lvhowto/build\\_web\\_service](http://zone.ni.com/reference/en-XX/help/371361K-01/lvhowto/build_web_service)

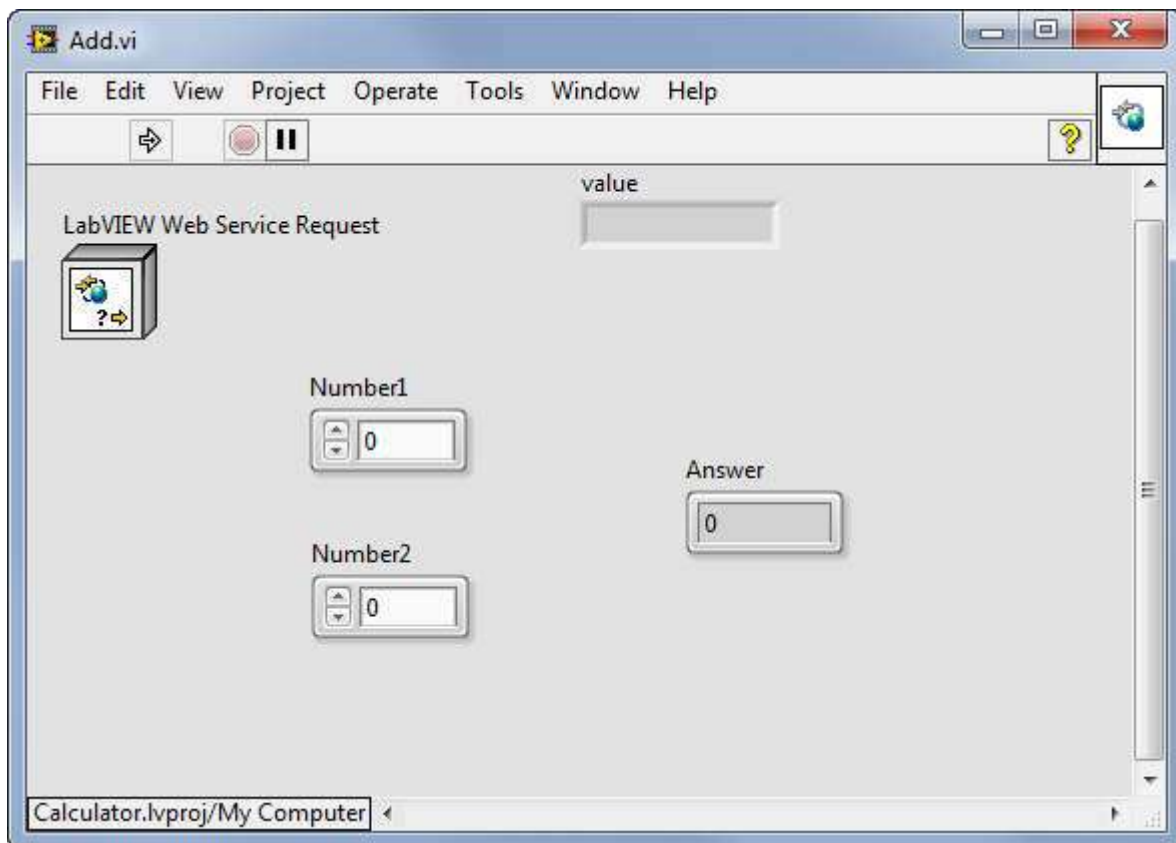
Create a new LabVIEW project:



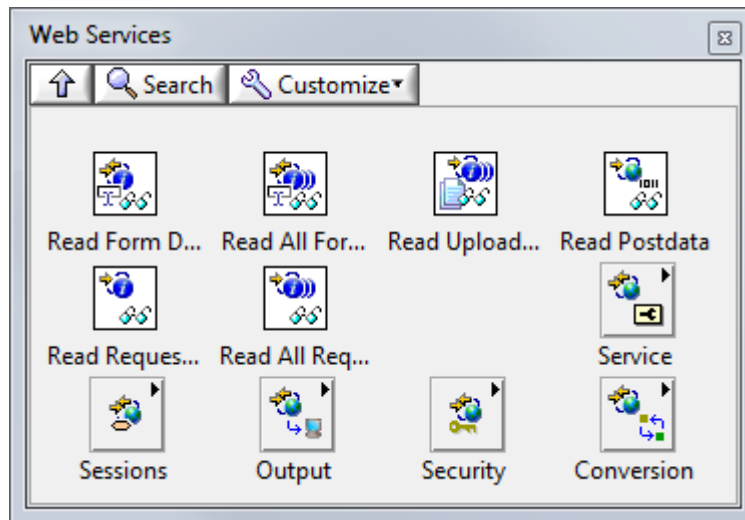
The Project Explorer should now look like this:



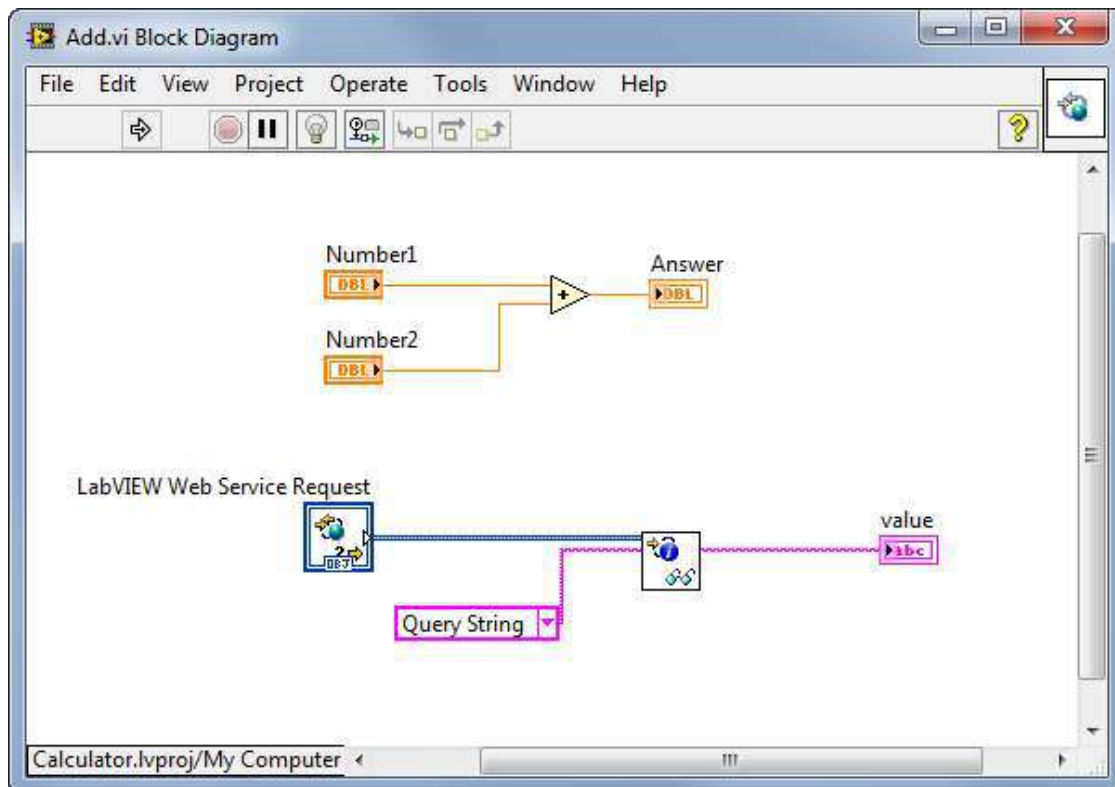
Create a Web Service Method (SubVI):



Below we see the Web Services palette in LabVIEW

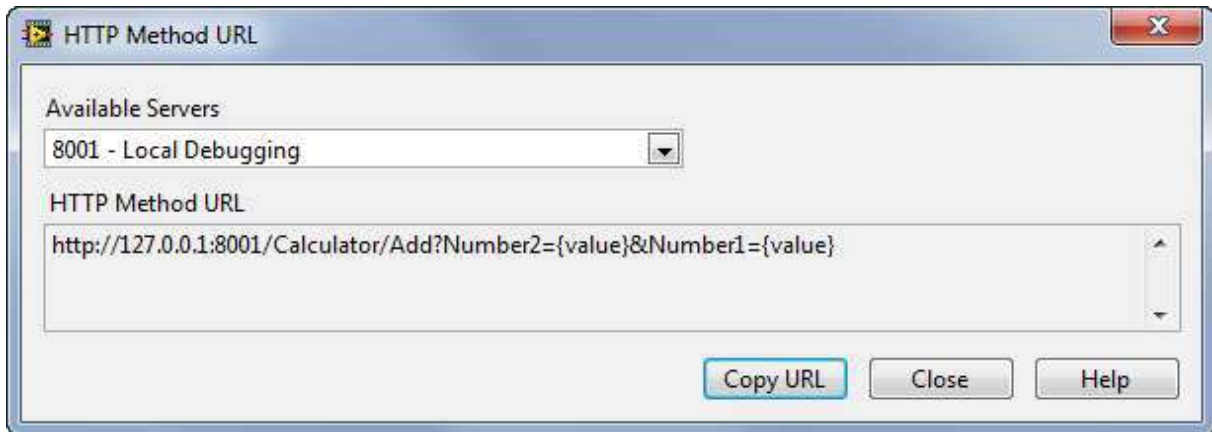


Create the following block diagram:

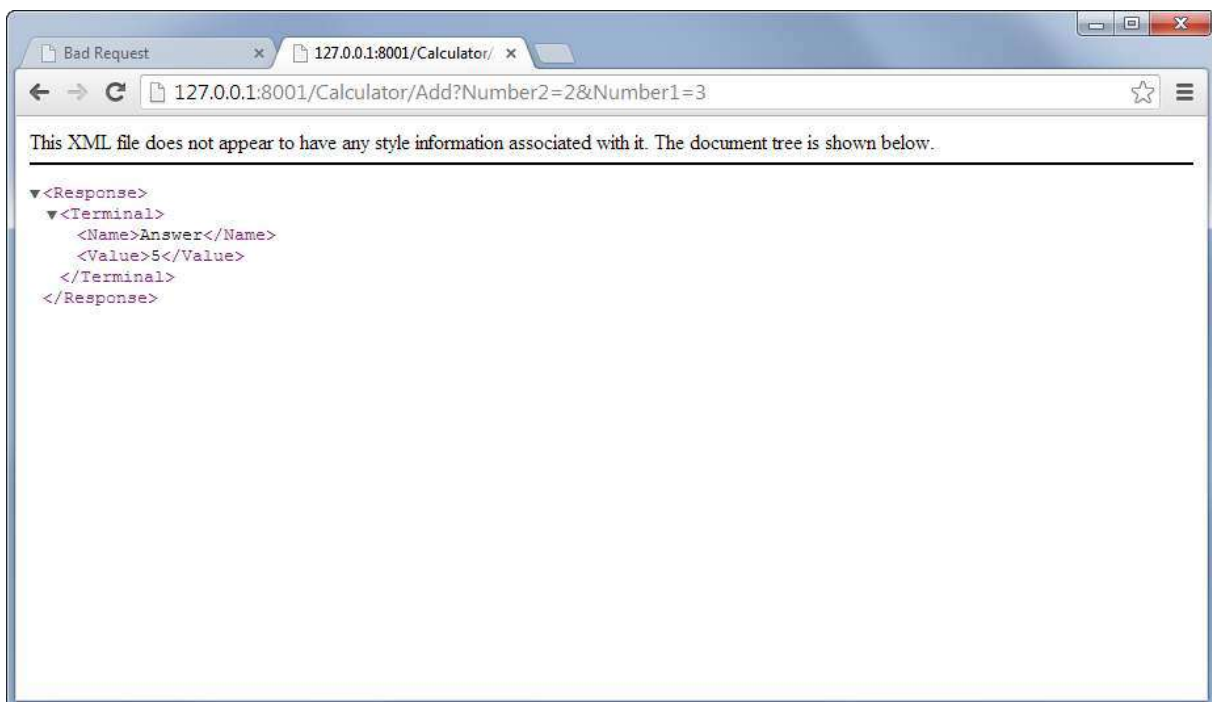


Test it:





In the Web browser it should look like this:



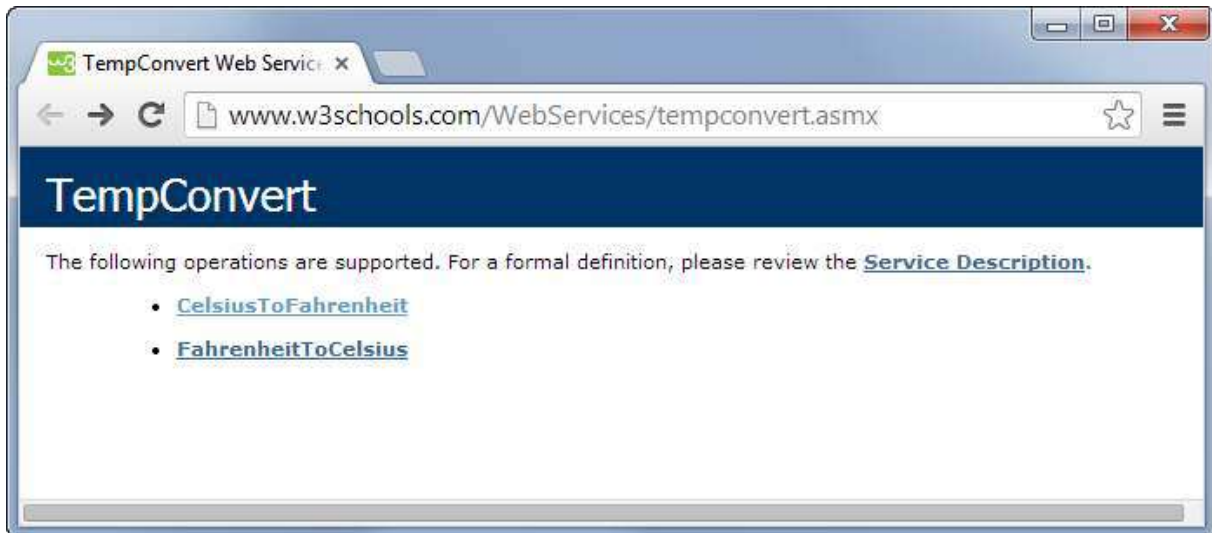
## 8.2. LabVIEW Client

### 8.2.1. Temperature Conversion

We will get data from a Web Service using LabVIEW.

We will use the TUC Weather station Web Service as example:

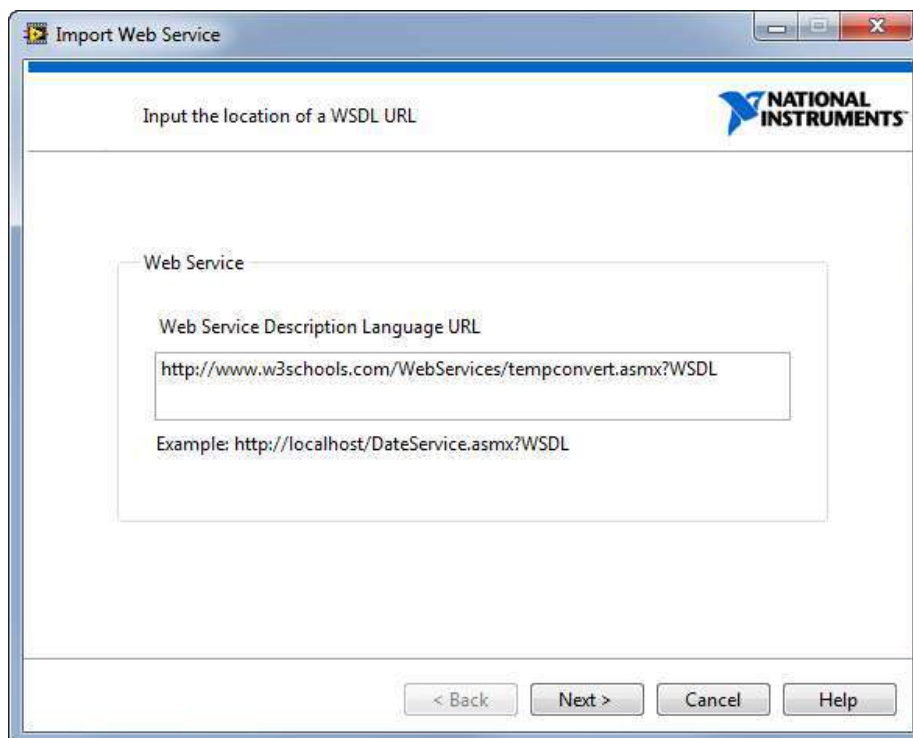
<http://www.w3schools.com/WebServices/tempconvert.asmx>



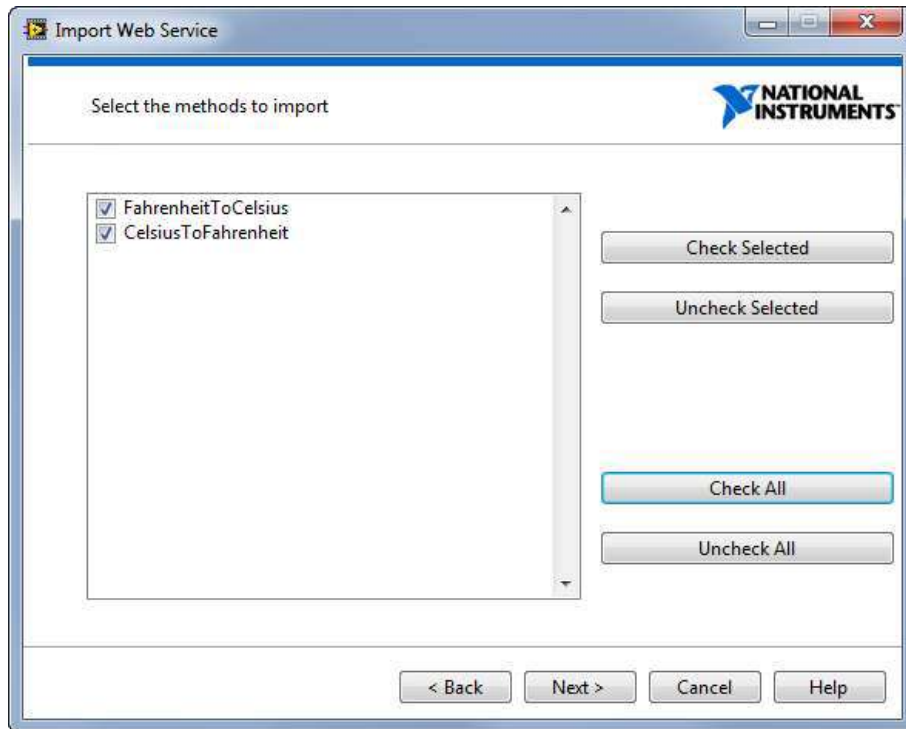
WDSL:

<http://www.w3schools.com/WebServices/tempconvert.asmx?WSDL>

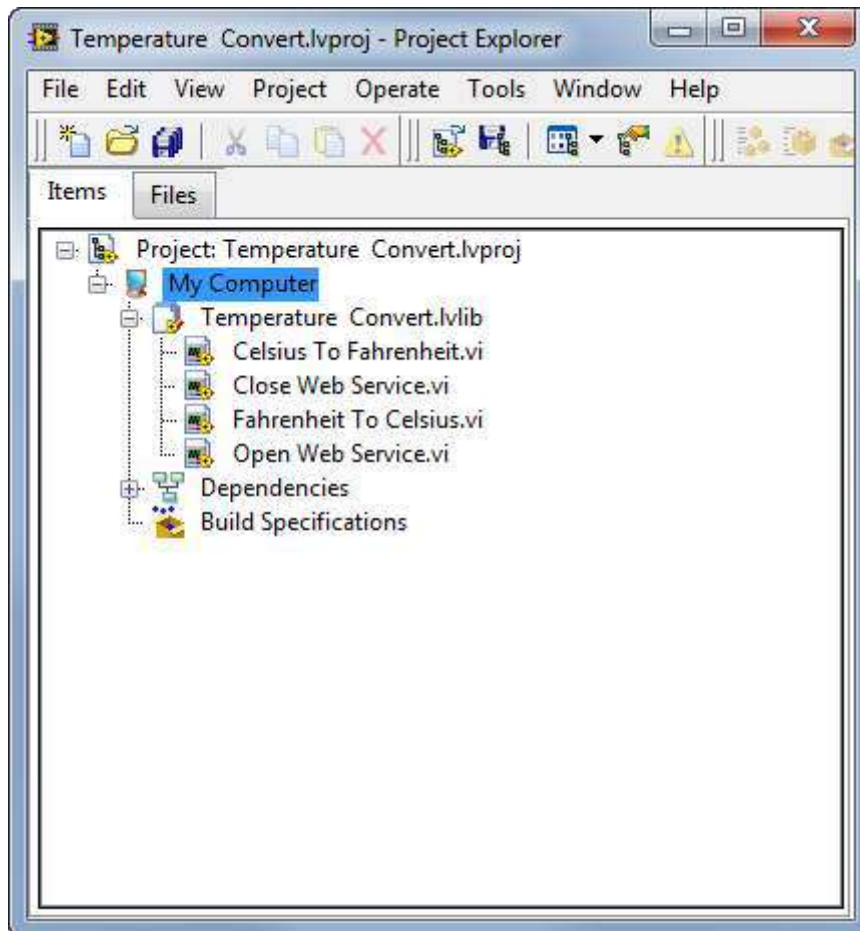
LabVIEW:



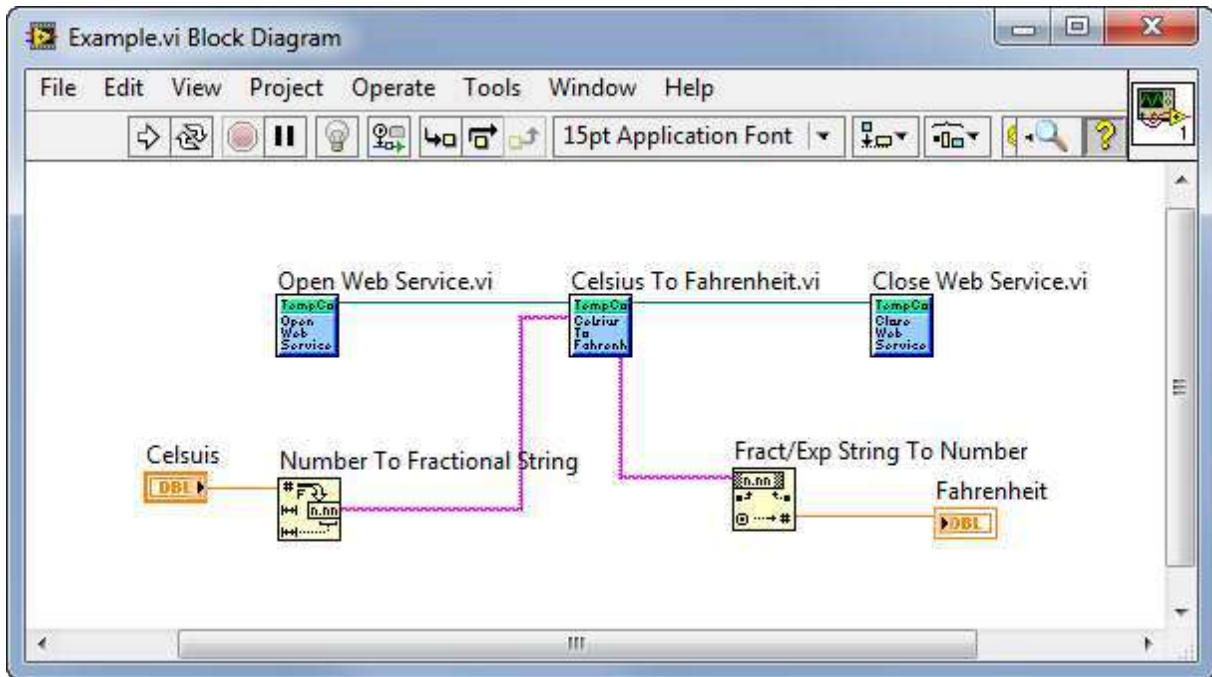
Select which of the available Web Methods you want to import:



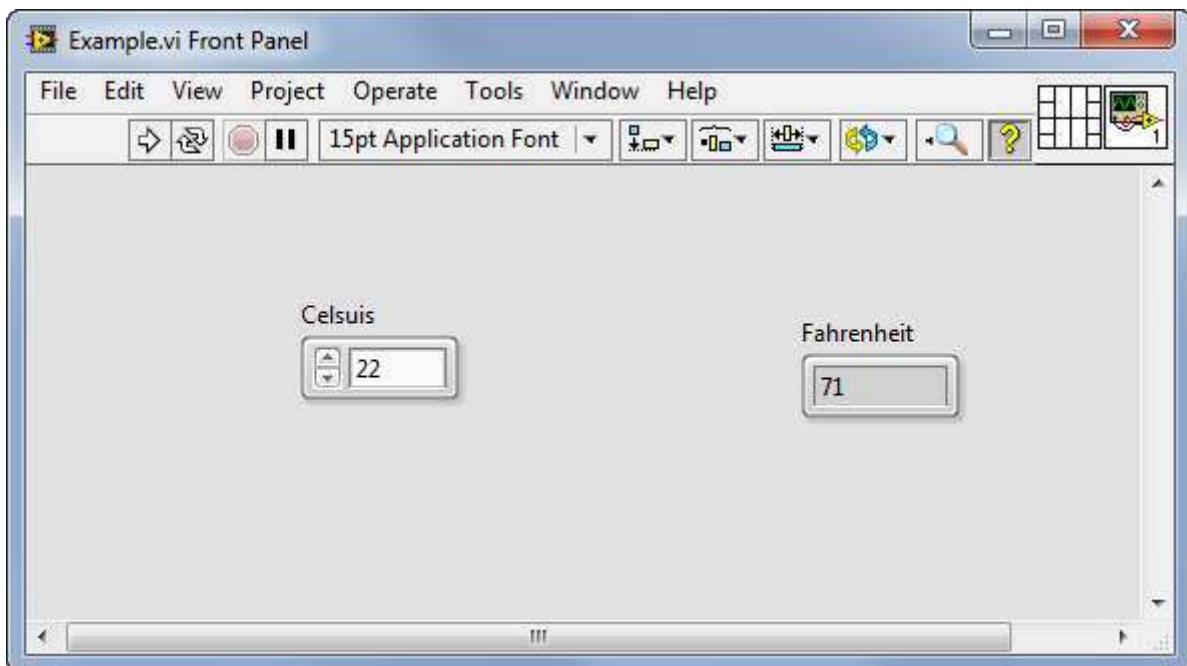
The Project Explorer should look like this:



Create a simple example:



Front Panel:

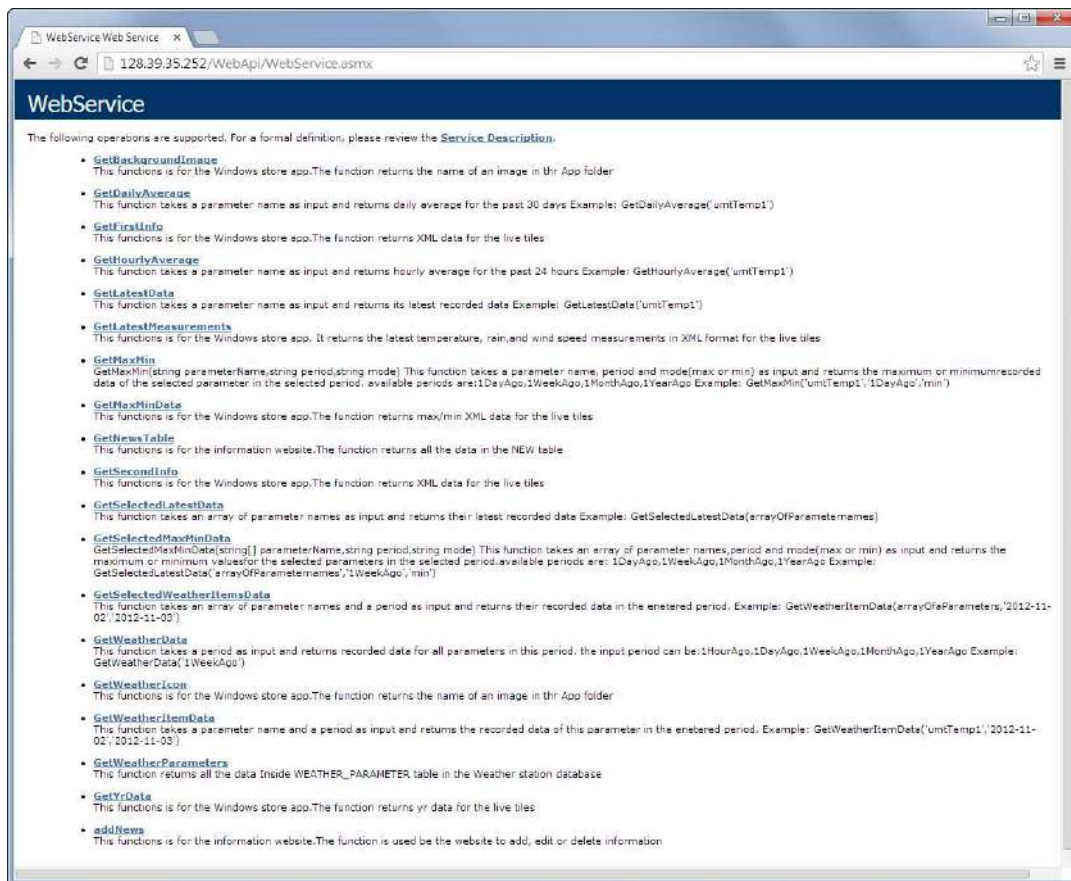


## 8.2.2. Weather Station Example

We will get data from a Web Service using LabVIEW.

We will use the TUC Weather station Web Service as example:

<http://128.39.35.252/WebApi/WebService.asmx>



The WSDL:

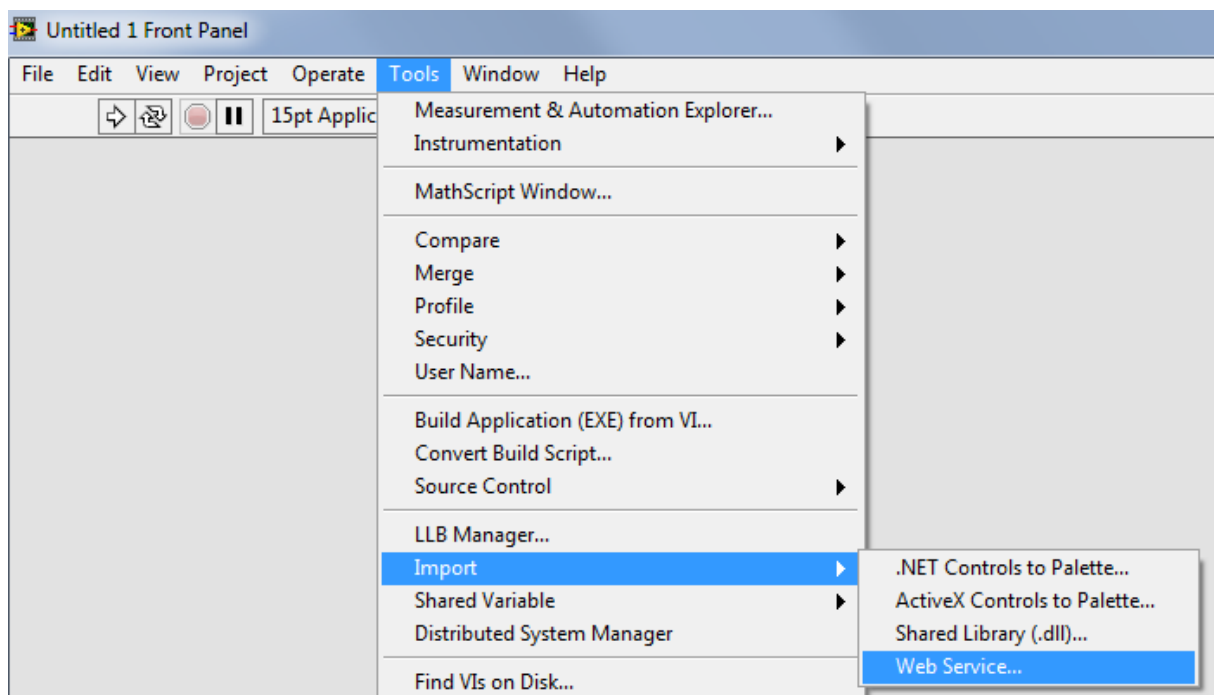
<http://128.39.35.252/WebApi/WebService.asmx?WSDL>

```

This XML file does not appear to have any style information associated with it. The document tree is shown below.
<?xml version='1.0' encoding='utf-8'>
<definitions xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:tns="http://schemas.xmlsoap.org/wsdl/tms/" xmlns:tm="http://schemas.xmlsoap.org/wsdl/tns/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tm="http://schemas.xmlsoap.org/wsdl/tns/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://TUOWeather.org/">
<wsdl:types>
<xs:schema elementFormDefault="qualified" targetNamespace="http://TUOWeather.org/">
<xs:element name="GetMaxMin">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="0" maxOccurs="1" name="parameterName" type="xsd:string"/>
<xs:element minOccurs="0" maxOccurs="1" name="period" type="xsd:string"/>
<xs:element minOccurs="0" maxOccurs="1" name="mode" type="xsd:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GetMaxMinResponse">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="1" maxOccurs="1" name="GetMaxMinResult" type="tns:CustomData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:complexType name="CustomData">
<xs:sequence>
<xs:element minOccurs="1" maxOccurs="1" name="Value" type="xsd:double"/>
<xs:element minOccurs="0" maxOccurs="1" name="TimeStamp" type="xsd:string"/>
<xs:element minOccurs="0" maxOccurs="1" name="Unit" type="xsd:string"/>
<xs:element minOccurs="0" maxOccurs="1" name="Description" type="xsd:string"/>
</xs:sequence>
</xs:complexType>
<xs:element name="GetLatestData">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="0" maxOccurs="1" name="parameterName" type="xsd:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GetLatestDataResponse">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="1" maxOccurs="1" name="GetLatestDataResult" type="tns:CustomData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GetSelectedMaxMinData">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="0" maxOccurs="1" name="parameterName" type="tns:ArrayOfString"/>
<xs:element minOccurs="0" maxOccurs="1" name="period" type="xsd:string"/>
<xs:element minOccurs="0" maxOccurs="1" name="mode" type="xsd:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>

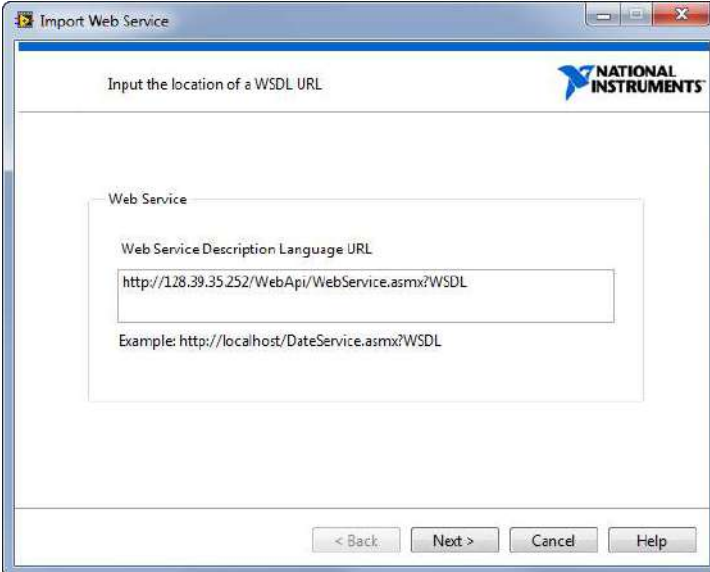
```

Lets start creating the the Web Service client in LabVIEW:



Type the Web Service path/URL:





Import Web Service

Input the location of a WSDL URL

NATIONAL INSTRUMENTS

Web Service

Web Service Description Language URL

Example: <http://localhost/DateService.asmx?WSDL>

< Back   Next >   Cancel   Help

...



Import Web Service

Input the authentication information

NATIONAL INSTRUMENTS

Authentication

Web service requires authentication

User name\*  Password

Domain

Proxy Server

Use a proxy server for your LAN

Name/Address\*  Port

Proxy service requires authentication

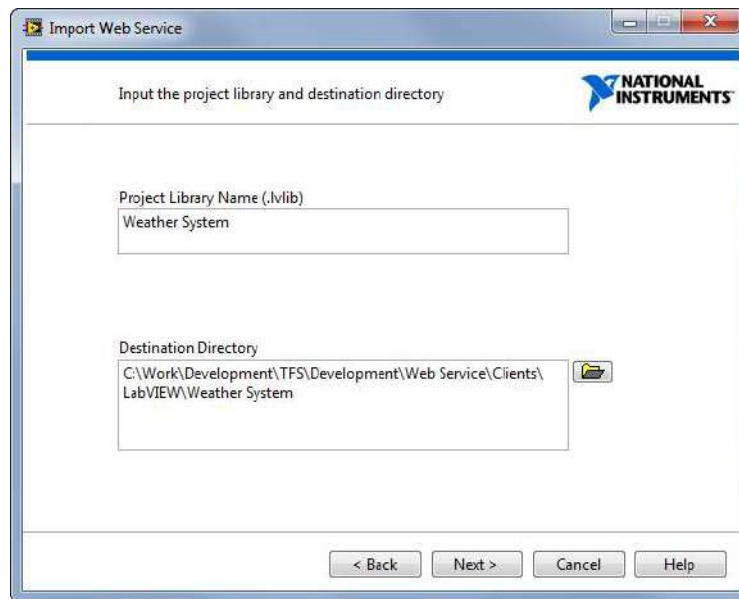
User name\*  Password

Domain

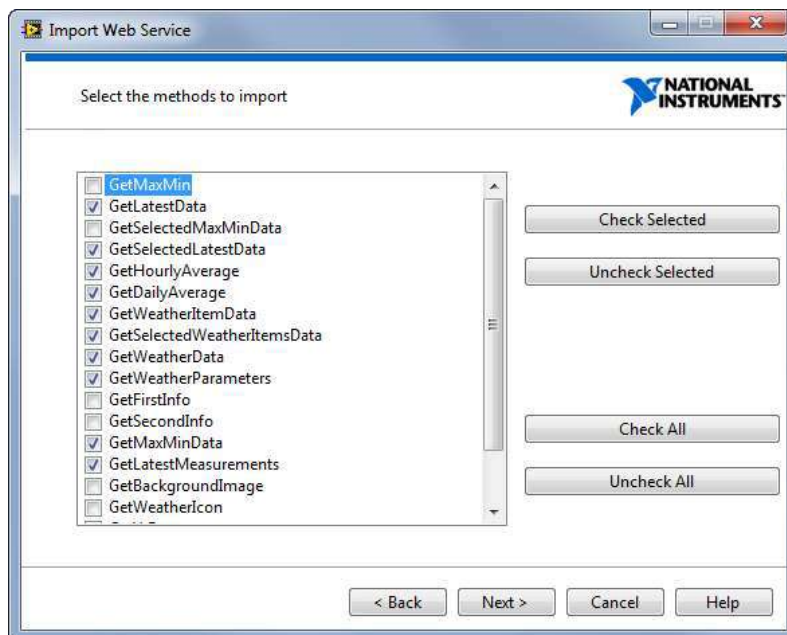
\* Marked items are required when the option is enabled.

< Back   Next >   Cancel   Help

...

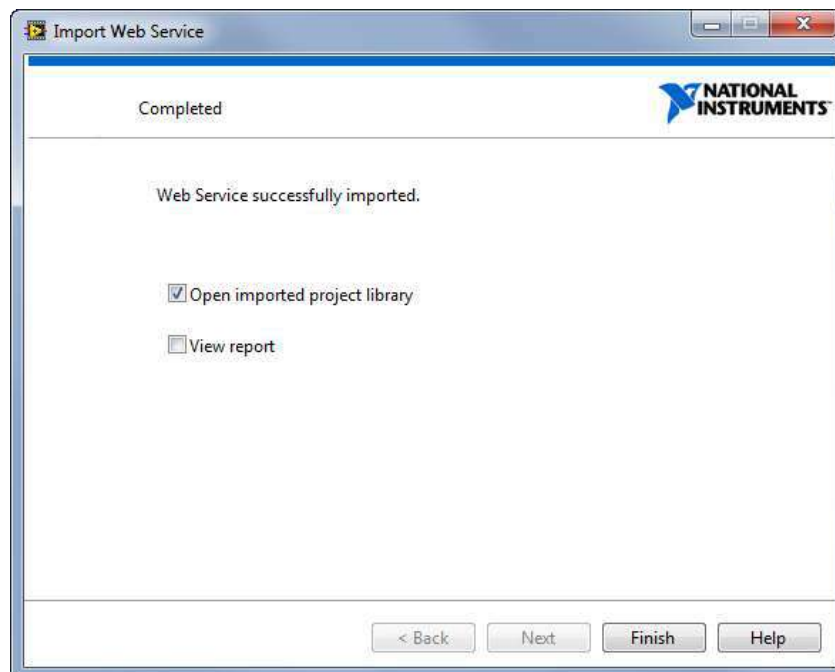
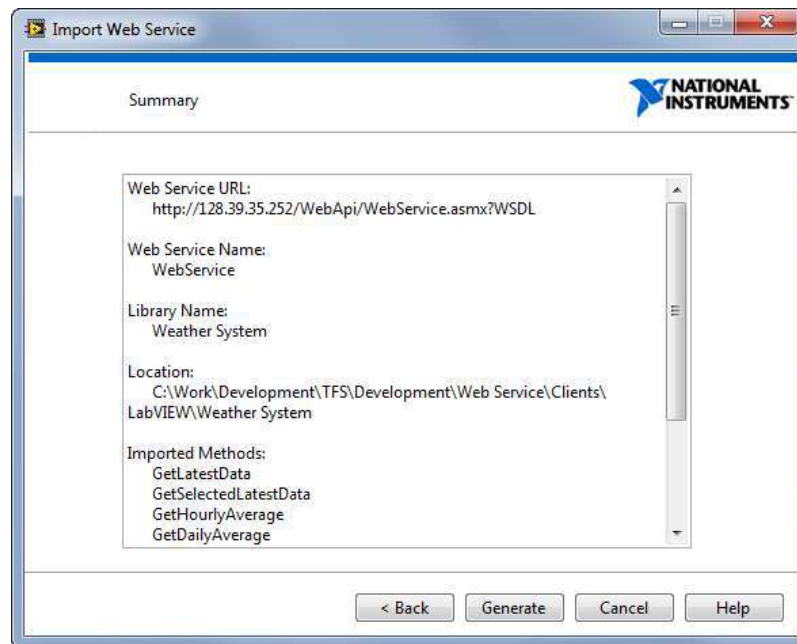


...

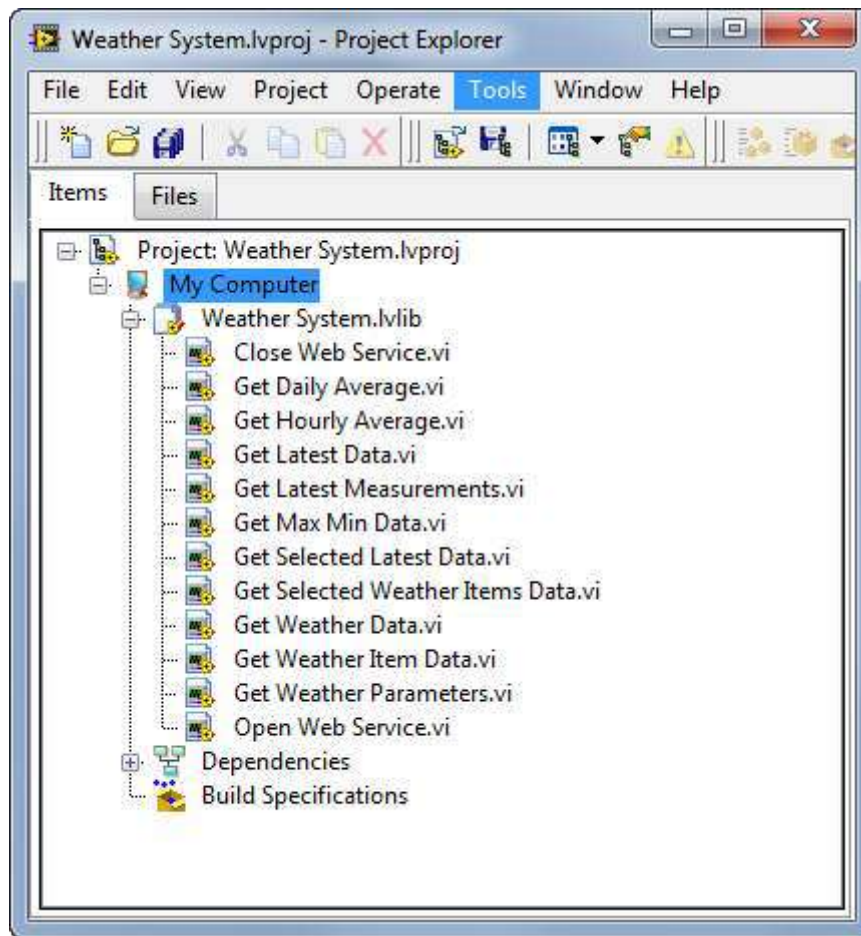


...

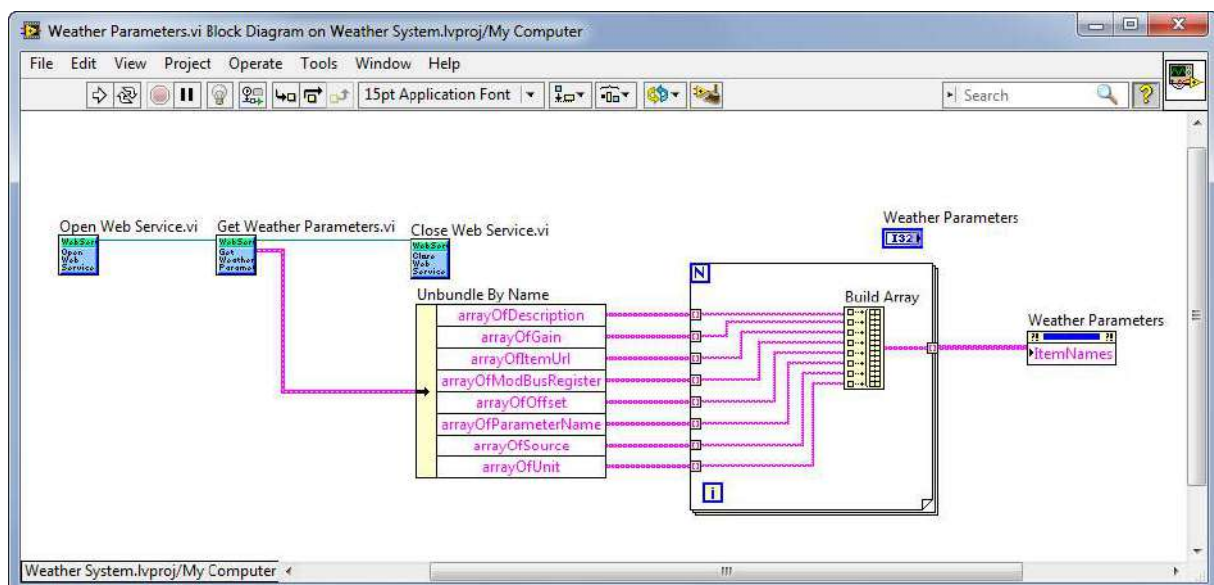




...



...



Final results:

Weather Parameters.vi Front Panel on Weather System.lvproj/My Computer

File Edit View Project Operate Tools Window Help

15pt.Application Font

Search

Weather Parameters

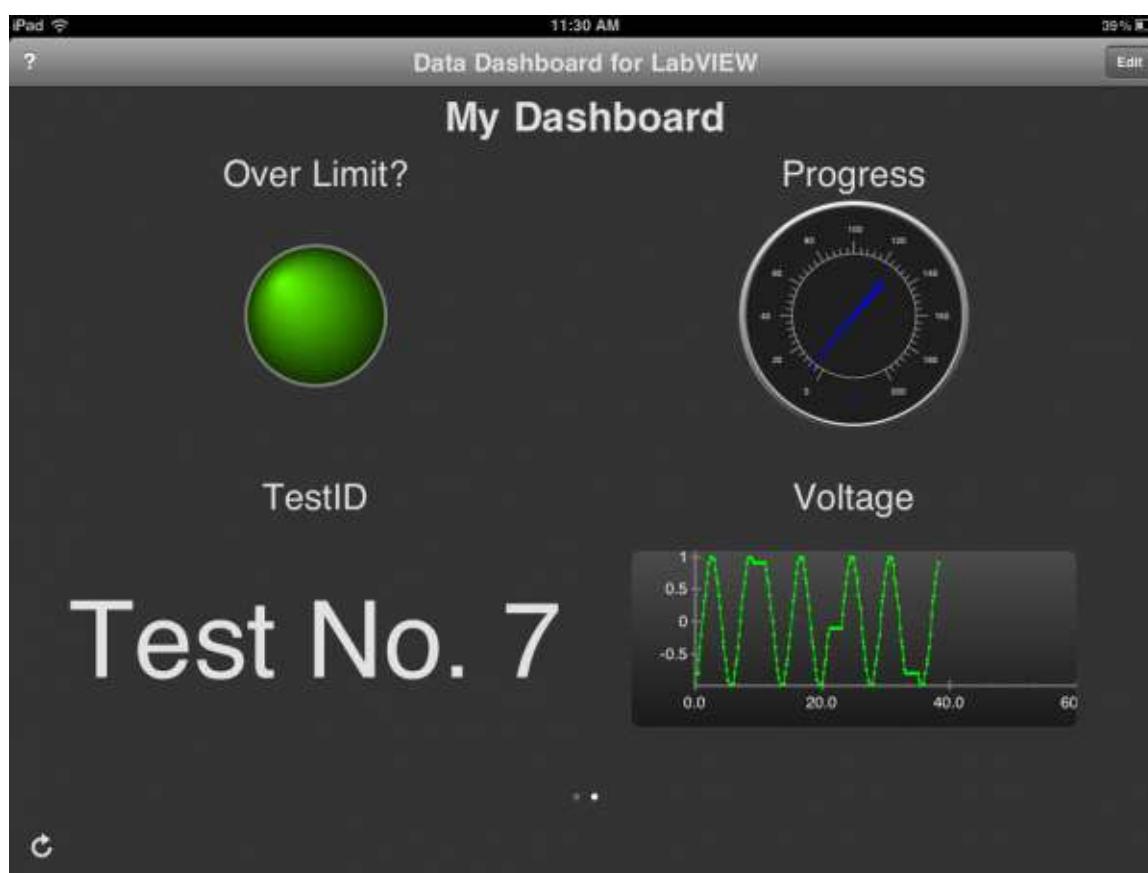
Parameter	Gain	OPC Tag	Modbus Register	Offset	ParameterName	Source	Unit
Time of last measurement	1	opc://localhost/Kepwi	1	0	umtLastMeasTime	Modbus	Timestamp
Wind speed	0,448	opc://localhost/Kepwi	5	0	umtWindSpeed	Modbus	m/s
Wind direction- unadjusted	1	opc://localhost/Kepwi	9	0	umtRawWindDir	Modbus	°
Wind direction	1	opc://localhost/Kepwi	13	0	umtAdjWindDir	Modbus	°
Relative humidity	1	opc://localhost/Kepwi	17	0	umtRelHumidity	Modbus	%
Barometric pressure - unadj	33,86	opc://localhost/Kepwi	21	0	umtRawBaromPress	Modbus	hPa
Barometric pressure	33,86	opc://localhost/Kepwi	25	0	umtAdjBaromPress	Modbus	hPa
Electronic compass wind dir	1	opc://localhost/Kepwi	29	0	umtTrueNorthOffset	Modbus	°
Last reported time of rainfall	1	opc://localhost/Kepwi	33	0	umtLastRain	Modbus	Timestamp
Air temperature	0,5555556	opc://localhost/Kepwi	37	-17,7777	umtTemp1	Modbus	°C
Wind chill	0,555556	opc://localhost/Kepwi	41	-17,7777	umtWindChill	Modbus	°C
Heat index	0,555556	opc://localhost/Kepwi	45	-17,7777	umtHeatIndex	Modbus	°C
Dew Point	0,555556	opc://localhost/Kepwi	49	-17,7777	umtDewPoint	Modbus	°C
Density Altitude	0,3048	opc://localhost/Kepwi	53	0	umtDensityAltitude	Modbus	m
3-sec avg of wind speed	0,448	opc://localhost/Kepwi	57	0	umt3SecRollAvgWindSpeed	Modbus	m/s
3-sec avg of wind direction	1	opc://localhost/Kepwi	61	0	umt3SecRollAvgWindDir	Modbus	°
2-min avg of wind speed	0,448	opc://localhost/Kepwi	65	0	umt2MinRollAvgWindSpeed	Modbus	m/s
2-min avg of adjusted wind d	1	opc://localhost/Kepwi	69	0	umt2MinRollAverageWindDir	Modbus	°
10-min avg of wind speed	0,448	opc://localhost/Kepwi	73	0	umt10MinAverageWindSpeed	Modbus	m/s
10-min avg of adjusted wind	1	opc://localhost/Kepwi	77	0	umt10MinRollAvgWindDir	Modbus	°
10-min wind direction at ma	1	opc://localhost/Kepwi	81	0	umt10MinWindGustDir	Modbus	°
10-min max wind speed	0,448	opc://localhost/Kepwi	85	0	umt10MinWindGustSpeed	Modbus	m/s
Time of 10-min max wind sp	1	opc://localhost/Kepwi	89	0	umt10MinWindGustTime	Modbus	Timestamp
60-min wind direction at ma	1	opc://localhost/Kepwi	93	0	umt60MinWindGustDir	Modbus	°

Weather System.lvproj/My Computer

# 9. Data Dashboard for LabVIEW

The Data Dashboard for LabVIEW app lets you create a custom dashboard that can remotely control and monitor running NI LabVIEW applications using Web Services.

Web Site: [www.ni.com/mobile](http://www.ni.com/mobile)



The App is available for IOS, Android and Windows 8 (Windows Store App).

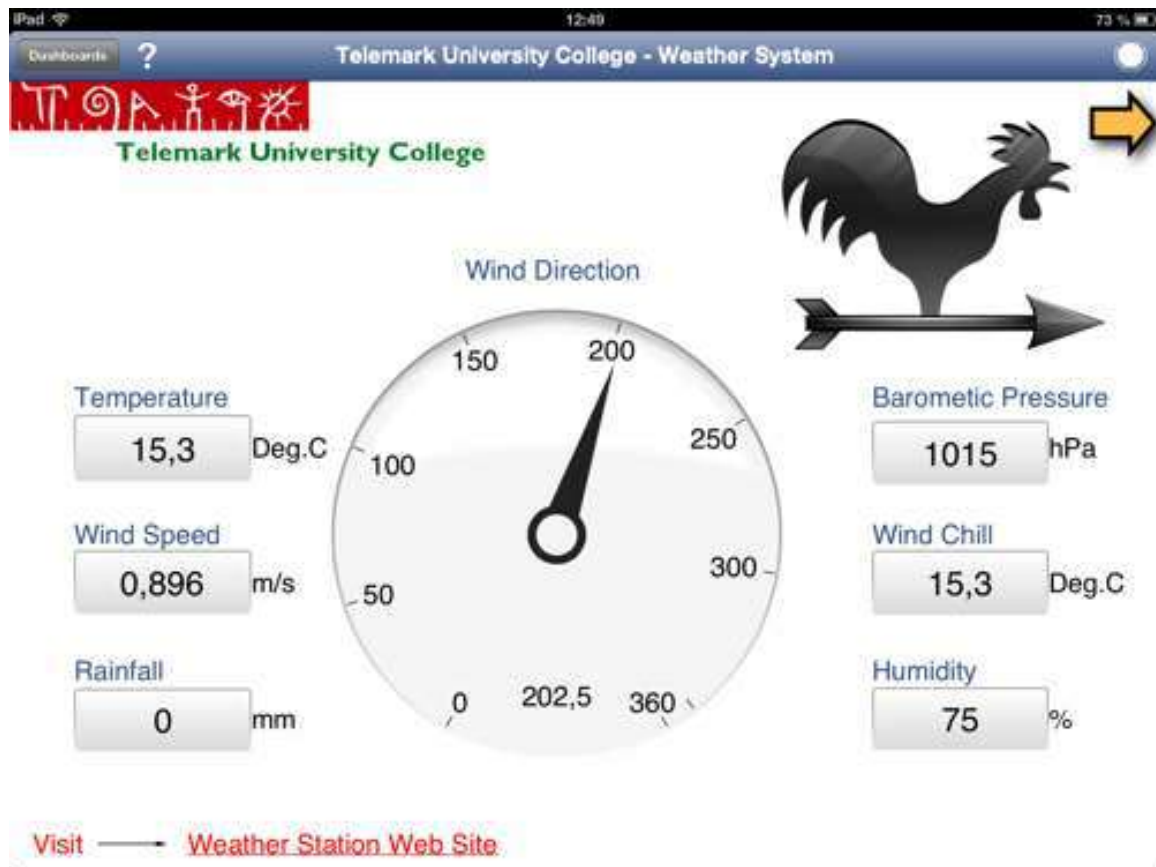
Download Dashboard App for iPad here:

<https://itunes.apple.com/us/app/data-dashboard-for-labview/id481303987?mt=8>

Getting Started with the Data Dashboard for LabVIEW:

<http://www.ni.com/white-paper/13757/en/>

A premaid Data Dashboard for the Weather system at telemark University College is shown below:



Download premade Dashboard here: <http://home.hit.no/~hansha/?page=weather>

The Data Dashboard for the Weather system is using the following web service:

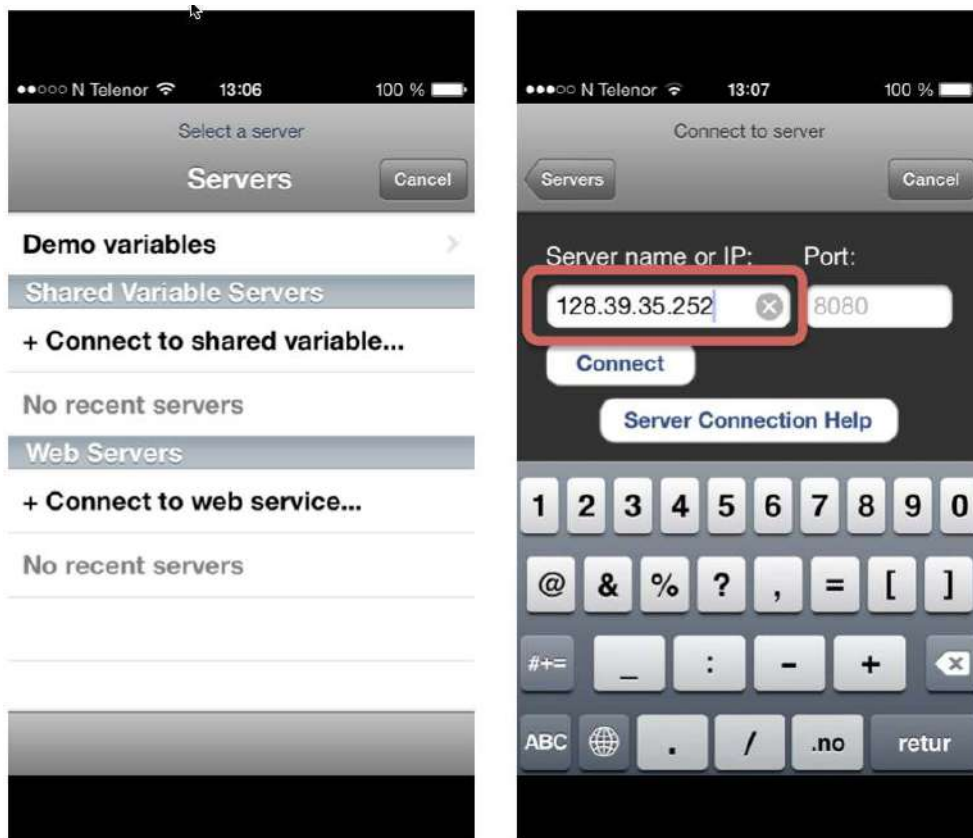
<http://128.39.35.252:8080/TUCweather/WebServices>

We will use this Web Service in order to create a simple example (using an iPhone).

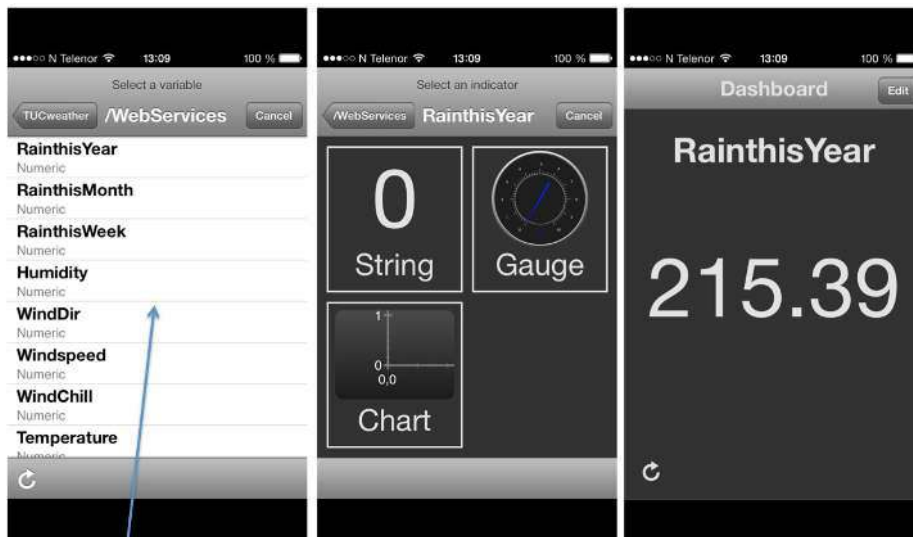
Download and open the LabVIEW for Data Dashboard App:



Connect to the Web Service:



Select one of the available web methods:



Available Methods in the Web Service

Try some of the other available Methods as well

# 10. Python

...

# 11. MATLAB

....





**Høgskolen i Telemark**

**Telemark University College**

**Faculty of Technology**

**Kjølnes Ring 56**

**N-3914 Porsgrunn, Norway**

**[www.hit.no](http://www.hit.no)**

---

**Hans-Petter Halvorsen, M.Sc.**

**Telemark University College**

**Faculty of Technology**

**Department of Electrical Engineering, Information Technology and Cybernetics**

**E-mail: [hans.p.halvorsen@hit.no](mailto:hans.p.halvorsen@hit.no)**

**Blog: <http://home.hit.no/~hansha/>**

---

