FREE eBook

LEARNING web-services

#web-services

Table of Contents

About

Chapter 1: Getting started with web-services

Remarks

A web-service is a web application component that enables communication between applications for integration purposes.

Web-services follow a client-server architecture. An application "offers" a web-service (server) and other applications "consume" the web-service (client).

They are implemented over HTTP using requests and responses.

The main types of web-services are:

- SOAP (**S**imple **O**bject **A**ccess **P**rotocol)
- REST (**RE**presentational **S**tate **T**ransfer)

Examples

Installation or Setup

Server side (Host Webservices)

Web services must be installed and running (deployed) in a web server as web application components. They can be part of a bigger application, or they can be deployed alone as they may compose a complete application.

It is responsibility of the server to forward an incoming HTTP request to the corresponding deployed application, and responsibility of the application to handle the request according to:

- the HTTP verb (GET, POST, PUT, DELETE, OPTIONS, HEAD, TRACE, CONNECT)
- the request URL

The application uses the combination of these elements to locate the corresponding web-service component that should process the request.

After the web-service is located, then the request parameters are used as input data to the web-service. The web-service is responsible to convert data to the correct datatypes, and to establish a convention with the clients about transmitting different datatypes.

The web-service is processing the input data and it produces an output dataset. The output dataset is wrapped in a HTTP response and it is sent back to the sender of the request.

Client-side

A client has to prepare a HTTP request, complying to the rules of the server, and send it to the server. The response that will be received will contain the required data.

Why to use web-services

Using web-services client programs and a server programs can exchange information and collaborate to produce new services and results regardless their physical location and the technology they are built on. They only need to comply with the application level specifications.

The difference between using web-services and web-HTML-serving (browsing) is mainly that web-services are focused and specialized in processing and converting data types to produce structured results, that can be used for remote procedure calling. Web-HTML-serving is more about serving renderable/downloadable resources.

Exchanging process results using web-services is facilitating:

- integration of applications
- separation of concerns
- distributed/decentralized application architectures

Java implementations

In Java web-services are implemented as servlets. The most popular web-services frameworks are implementing a servlet that needs to be mapped with a URL. Examples of frameworks:

- Axis
- CXF
- Jersey

Web service related components

1. WSDL (Web service Description Language)
2. UDDI (Universal Description Discovery and Integration)
3. SOAP (Simple Object Access Protocol)

Read Getting started with web-services online: https://riptutorial.com/web-services/topic/5309/getting-started-with-web-services

Chapter 2: asp.net web-services

Introduction

Web Service is an application that is designed to interact directly with other applications over the internet. In simple sense, Web Services are means for interacting with objects over the Internet. The `Web serivce` consumers are able to invoke method calls on remote objects by using `SOAP` and `HTTP` over the Web. `WebService` is language independent and Web Services communicate by using standard web protocols and data formats, such as - HTTP - XML - SOAP

Syntax

1. **SOAP/WSDL**

   [Syntax: http://1111:22/HelloWorld]

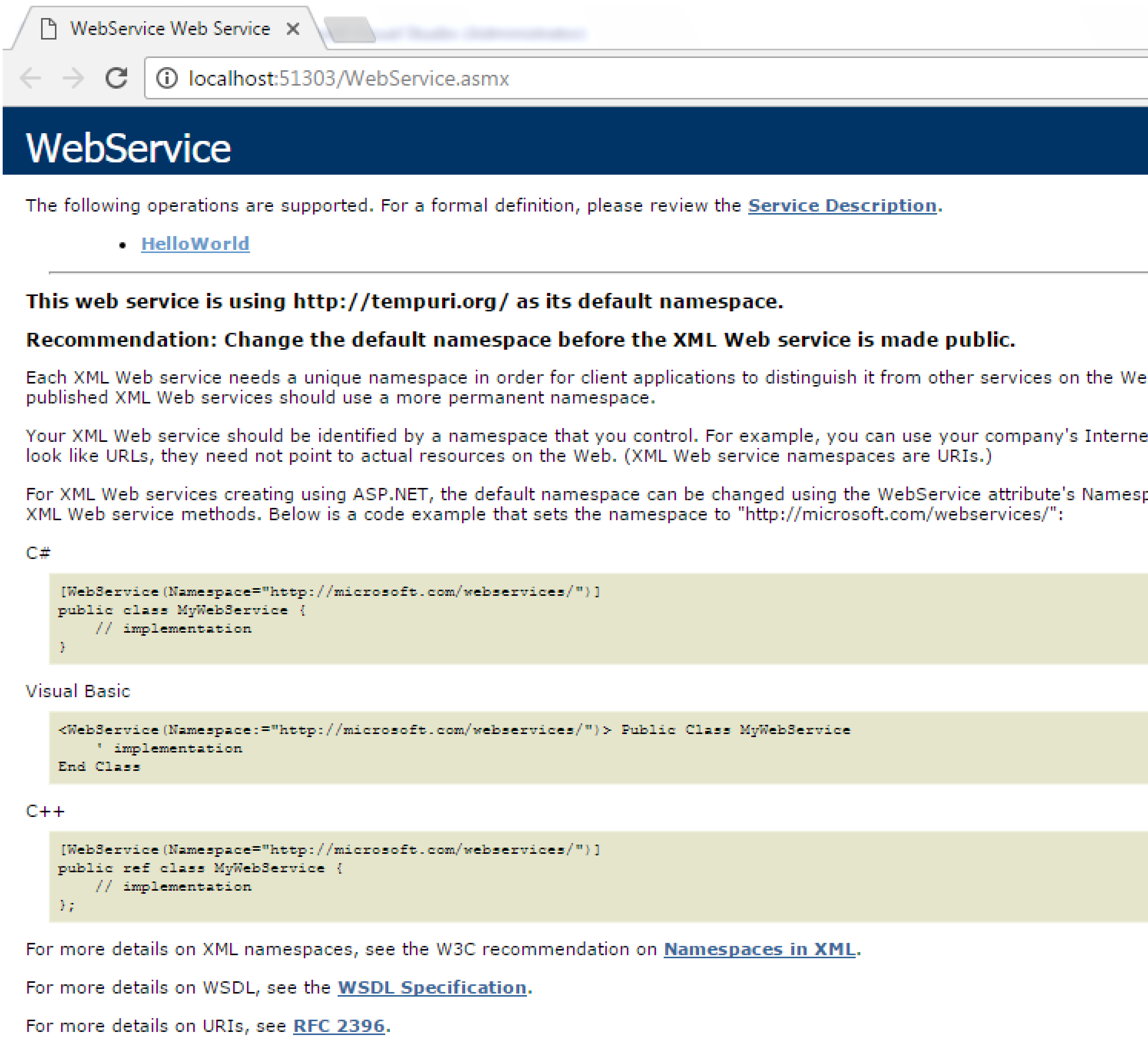
   [Response: WDSL]

Parameters

Parameters	Descriptions
PathParam	Binds the parameter passed to method to a value in path.
QueryParam	Binds the parameter passed to method to a query parameter in path.
MatrixParam	Binds the parameter passed to method to a HTTP matrix parameter in path.
HeaderParam	Binds the parameter passed to method to a HTTP header.
CookieParam	Binds the parameter passed to method to a Cookie.
FormParam	Binds the parameter passed to method to a form value.
DefaultValue	Assigns a default value to a parameter passed to method.
Context	Context of the resource for example HTTPRequest as a context.

Remarks

Now run the application that look like as follows.


WebService Web Service

localhost:51303/WebService.asmx

WebService

The following operations are supported. For a formal definition, please review the Service Description.

- HelloWorld

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the We
published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Interne
look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namesp
XML Web service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":

C#

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // implementation
}
```

Visual Basic

```
<WebService(Namespace:="http://microsoft.com/webservices/")> Public Class MyWebService
    ' implementation
End Class
```

C++

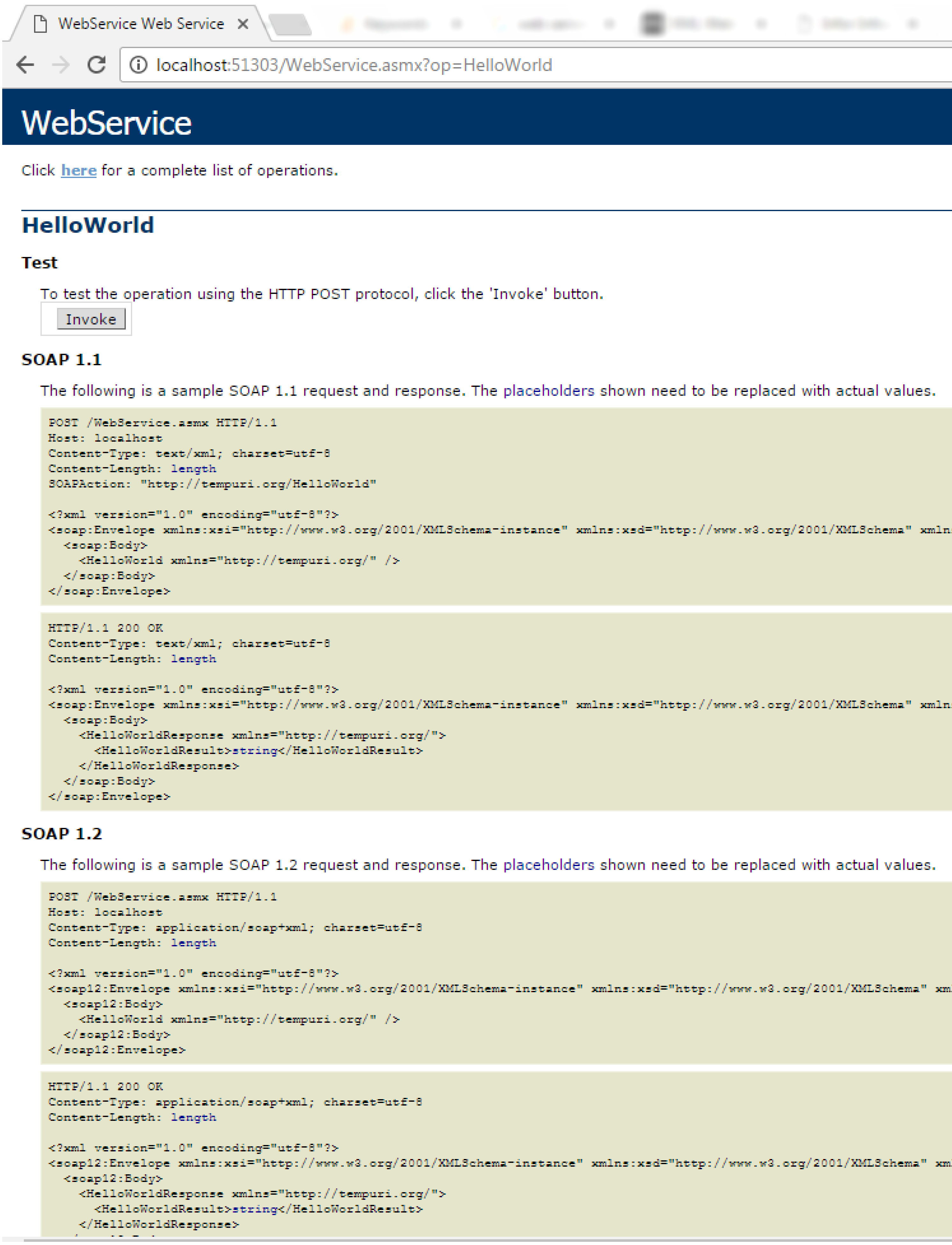
```
[WebService(Namespace="http://microsoft.com/webservices/")]
public ref class MyWebService {
    // implementation
};
```

For more details on XML namespaces, see the W3C recommendation on Namespaces in XML.

For more details on WSDL, see the WSDL Specification.

For more details on URIs, see RFC 2396.

Now in the above we see our method that we are created in the webservice.cs file, so click on that method and provide input values and click on the "invoke" link as in.



WebService

Click here for a complete list of operations.

HelloWorld

Test

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

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /WebService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/HelloWorld"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmln
  <soap:Body>
    <HelloWorld xmlns="http://tempuri.org/" />
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmln
  <soap:Body>
    <HelloWorldResponse xmlns="http://tempuri.org/">
      <HelloWorldResult>string</HelloWorldResult>
    </HelloWorldResponse>
  </soap:Body>
</soap:Envelope>
```

SOAP 1.2

The following is a sample SOAP 1.2 request and response. The placeholders shown need to be replaced with actual values.

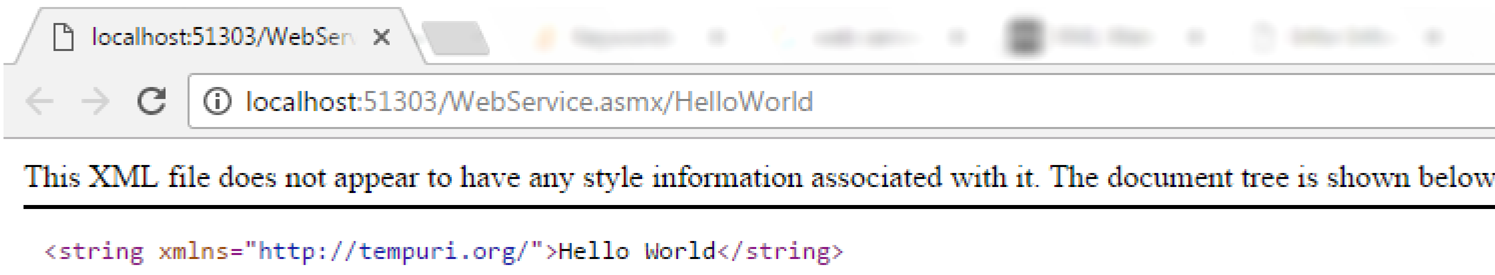
```
POST /WebService.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xm
  <soap12:Body>
    <HelloWorld xmlns="http://tempuri.org/" />
  </soap12:Body>
</soap12:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xm
  <soap12:Body>
    <HelloWorldResponse xmlns="http://tempuri.org/">
      <HelloWorldResult>string</HelloWorldResult>
    </HelloWorldResponse>
```

The output will be as follows



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

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

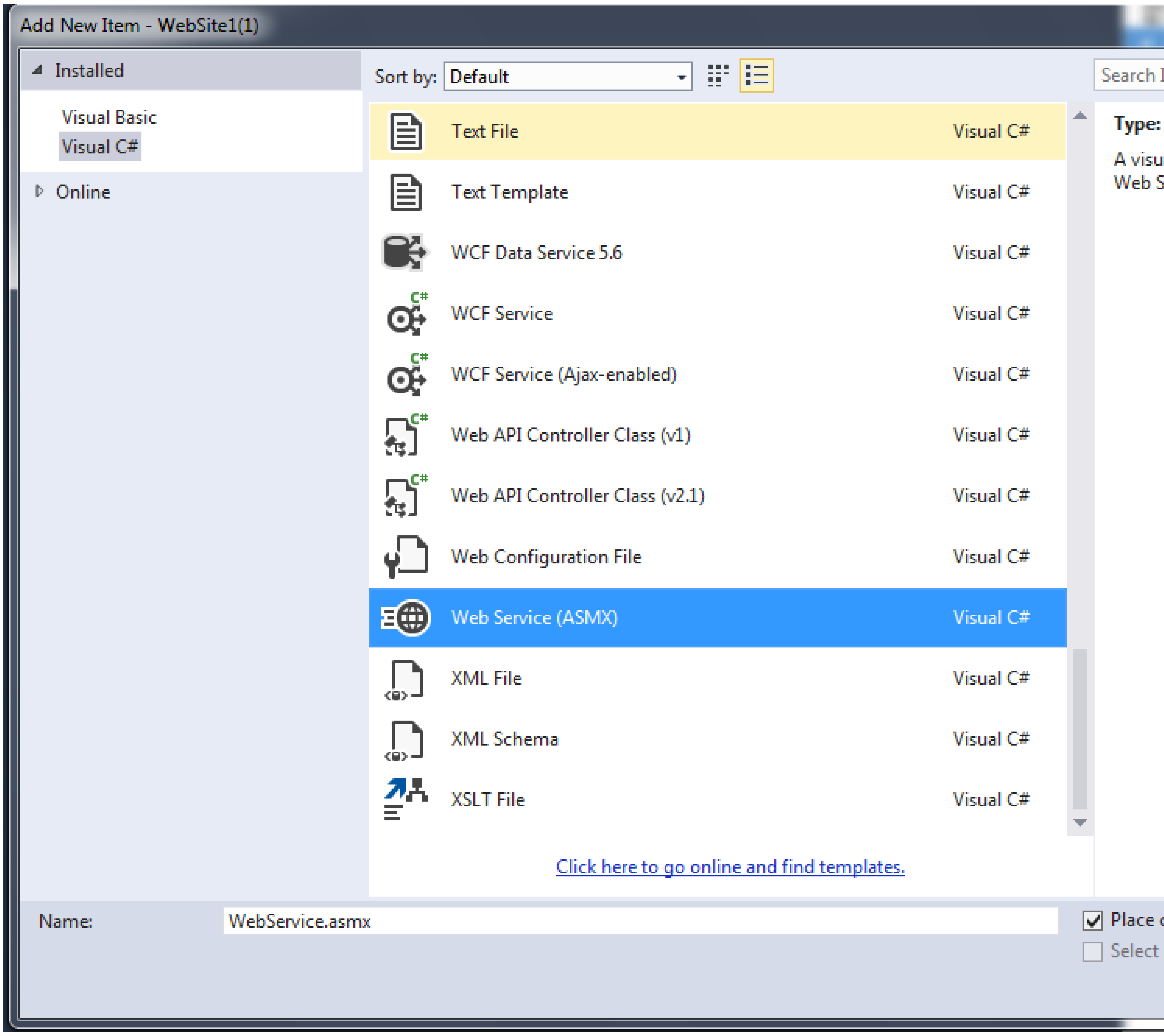
Examples

Note

If you closely observe that ,there is no separate web service template in .Framework 2010 as you see in 2008 while adding a project or web site it might be because of WCF.

So let us start using a different way to add a web service using a template

1. "Start" - "All Programs" - "Microsoft Visual Studio 2010"
2. "File" - "New Project" - "C#" - "Empty Web Application" (to avoid adding a master page)
3. Provide the web site a name such as "agetodays" or another as you wish and specify the location
4. Then right-click on Solution Explorer - "Add New Item" - you see the web service templates


Add New Item - WebSite1(1)

Installed
Visual Basic
Visual C#
Online

Sort by: Default

Search I

Text File	Visual C#
Text Template	Visual C#
WCF Data Service 5.6	Visual C#
WCF Service	Visual C#
WCF Service (Ajax-enabled)	Visual C#
Web API Controller Class (v1)	Visual C#
Web API Controller Class (v2.1)	Visual C#
Web Configuration File	Visual C#
Web Service (ASMX)	Visual C#
XML File	Visual C#
XML Schema	Visual C#
XSLT File	Visual C#

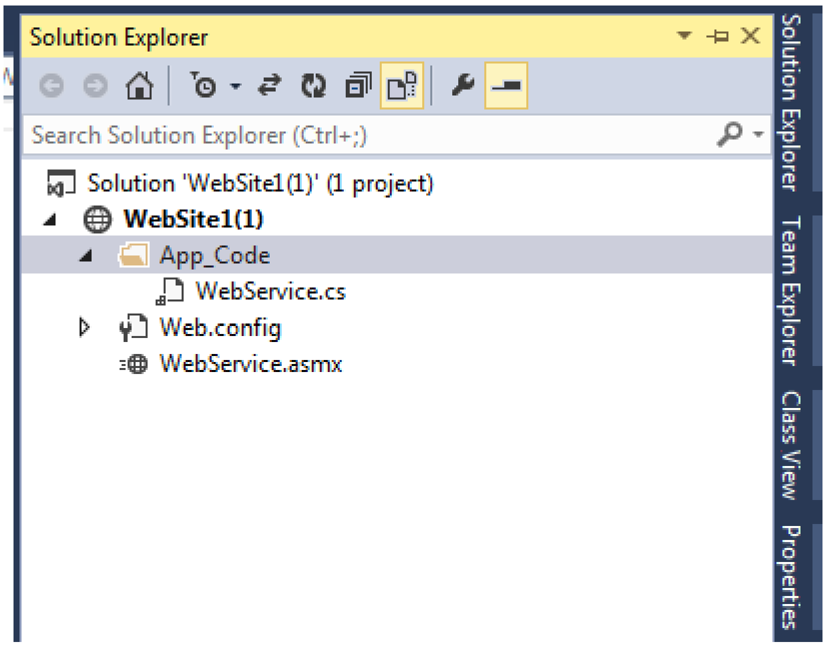
Click here to go online and find templates.

Type:
A visu
Web S

Name: WebService.asmx

[✓] Place c
[ ] Select

Select Web Service Template and click on add button. then after that the Solution Explorer look like as follows.



Solution 'WebSite1(1)' (1 project)
WebSite1(1)
App_Code
WebService.cs
Web.config
WebService.asmx

Then open the Webservice.cs class and write the following method followed by [webMethod] attribute as in.

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

Read asp.net web-services online: https://riptutorial.com/web-services/topic/8341/asp-net-web-services

Chapter 3: Calling Web Services programmatically using C#.net and Xamarin app development

Introduction

Here we will see Pro-grammatically calling and using web services in ASP.Net C# . For the purpose you will required to download following ddl which provides you many functions. Download ImportJson from https://drive.google.com/open?id=0B-2bGoHKJvnOckdPUHVjdFZTcFU

This article is very useful for those of you who are going to develop a project using ASP.NET C# Web services/ Web API Services. This article is also useful for those who are developing a project using Xamarin: Mobile App Development

Remarks

You required to give reference of ImportJson dll and restsharp ddl. ImportJson can be downloaded from here https://drive.google.com/open?id=0B-2bGoHKJvnOckdPUHVjdFZTcFU And restsharp.dll will get from internet

Any suggestion/contact, please note **akhandagale65@gmail.com**

Examples

Calling Simple GET Method

```
/// <summary>
    /// Simple Get method
    /// </summary>
    /// <returns> Json formated data </returns>
    public string GetJsonData1()
    {
        IOperations _Obj = ClsOperations.GetOperations();
        string url = "http://1.2.3.4:1234/Services/rest/CallService/WebRequest/";
        string jsonResult = _Obj.GetJsonResult(url);
        return jsonResult;
    }
```

Calling Web Service with Data POST/ POST Method

```
        /// <summary>
    /// Post Method with input parameter
    /// </summary>
    /// <returns> Json formated data </returns>
    public string GetJsonData2()
    {
        IOperations _Obj = ClsOperations.GetOperations();
        string url = "http://1.2.3.4:1234/Services/rest/CallService/WebRequest/";
        Dictionary<string, object> objDec = new Dictionary<string, object>();
        objDec.Add("@FirstParameter", "Value1");
        objDec.Add("@SecondParameter", "Value2");
        objDec.Add("@ThirdParameter", "Value3");
        string jsonResult = _Obj.GetJsonResult(url, objDec);
        return jsonResult;
    }
```

Calling Web Service with Data POST/ POST Method (Posted Data in JSON Format)

```
/// <summary>
    /// Post Method with Input/ data to post in JSON format
    /// </summary>
    /// <returns> Json formated data </returns>
    public string GetJsonData3()
    {
        IOperations _Obj = ClsOperations.GetOperations();
        string url = "http://1.2.3.4:1234/Services/rest/CallService/WebRequest/";
        string inputjson = "{\"@FirstParameter\": \"Value1\",\"@SecondParameter\":
\"Value2\",\"@ThirdParameter\": \"Value3\"}";
        string jsonResult = _Obj.GetJsonResult(url, null,inputjson );
        return jsonResult;
    }
```

Web Service call with output As IEnumerator object

```
/// <summary>
    /// Post Method with Input/ data to post in JSON format Or you can send dictionary as
shown in previous methods
    /// </summary>
    /// <returns> Json formated data </returns>
    public void  GetJsonData4()
    {
        IOperations _Obj = ClsOperations.GetOperations();
        string url = "http://1.2.3.4:1234/Services/rest/CallService/WebRequest/";
        string inputjson = "{\"@FirstParameter\": \"Value1\",\"@SecondParameter\":
\"Value2\",\"@ThirdParameter\": \"Value3\"}";
        string jsonResult = _Obj.GetJsonResult(url, null, inputjson);
        IEnumerator objIEnumerator = _Obj.GetJsonEnumerableResult(jsonResult);
        // you can perform further operations on it

    }
```

Web Service Output in List format or DataTable Format

```
    /// <summary>
        /// Post Method with Input/ data to post in JSON format Or you can send dictionary as
shown in previous methods
        /// </summary>
        /// <returns> Json formated data </returns>
        public DataTable  GetJsonData6()
        {
            IOperations _Obj = ClsOperations.GetOperations();
            string url = "http://1.2.3.4:1234/Services/rest/CallService/WebRequest/";
            string inputjson = "{\"@FirstParameter\": \"Value1\",\"@SecondParameter\":
\"Value2\",\"@ThirdParameter\": \"Value3\"}";
            IEnumerator objIEnumerator = _Obj.GetJsonEnumerableResult(url, null, inputjson);
            // you can perform further operations on it

            // If you want to convert it in Datatable / List

            List<ClsMyPropertyClass> lst = new List<ClsMyPropertyClass>();
            while (objIEnumerator.MoveNext())
            {

lst.Add(Newtonsoft.Json.JsonConvert.DeserializeObject<ClsLineEDoDetails>(objIEnumerator.Current.ToStri

            }
// Upto this you will get List , and you can perform operations on it

            // Now if youu want result in datatable, here i written function for List to
datatable conversion

            return CommonServiceCall.ToDataTable(lst);

        }
```

Forcefully make method GET OR POST

```
/* By Default if you send only url then automatically it will recognize as GET Method and if
service having parameters with, Then automatically will convert to POST Method. But I observed
some of the services having only URL but are POST Type. For the purpose you can forcefully
make the method as you want. As bellow:  */
        /// <summary>
        /// If you want make the service call GET OR POST forcefully then
        /// </summary>
        /// <returns> Json formated data </returns>
        public void GetJsonData5()
        {
            IOperations _Obj = ClsOperations.GetOperations();
            string url = "http://1.2.3.4:1234/Services/rest/CallService/WebRequest/";
            string inputjson = "{\"@FirstParameter\": \"Value1\",\"@SecondParameter\":
\"Value2\",\"@ThirdParameter\": \"Value3\"}";
string  _result =   _ Obj.GetJsonResult(url, null, inputjson, ServiceType.POST);;
                     }
```

Read Calling Web Services programmatically using C#.net and Xamarin app development online: https://riptutorial.com/web-services/topic/9689/calling-web-services-programmatically-using-csharp-net-and-xamarin-app-development

Credits

S. No	Chapters	Contributors
1	Getting started with web-services	Community, Naga G, sanastasiadis
2	asp.net web-services	Denish Parvadia
3	Calling Web Services programmatically using C#.net and Xamarin app development	Amol Khandagale