

C# & .NET Framework

Chapter 25: XML Web Services

Chapter 25: Objectives

- XML Web Services introduction
- The .NET XML Web Service Namespaces
- Building an XML Web Service by Hand
- Building an XML Web Service Using Visual Studio 2005
- The Role of the WebService Base Class
- Understanding the [WebService] Attribute
- Understanding the [WebServiceBinding] Attribute
- Understanding the [WebMethod] Attribute
- Exploring the Web Service Description Language (WSDL)
- XML Web Service Wire Protocols
- Generating Proxy Code Using Visual Studio 2005

XML Web Services Introduction

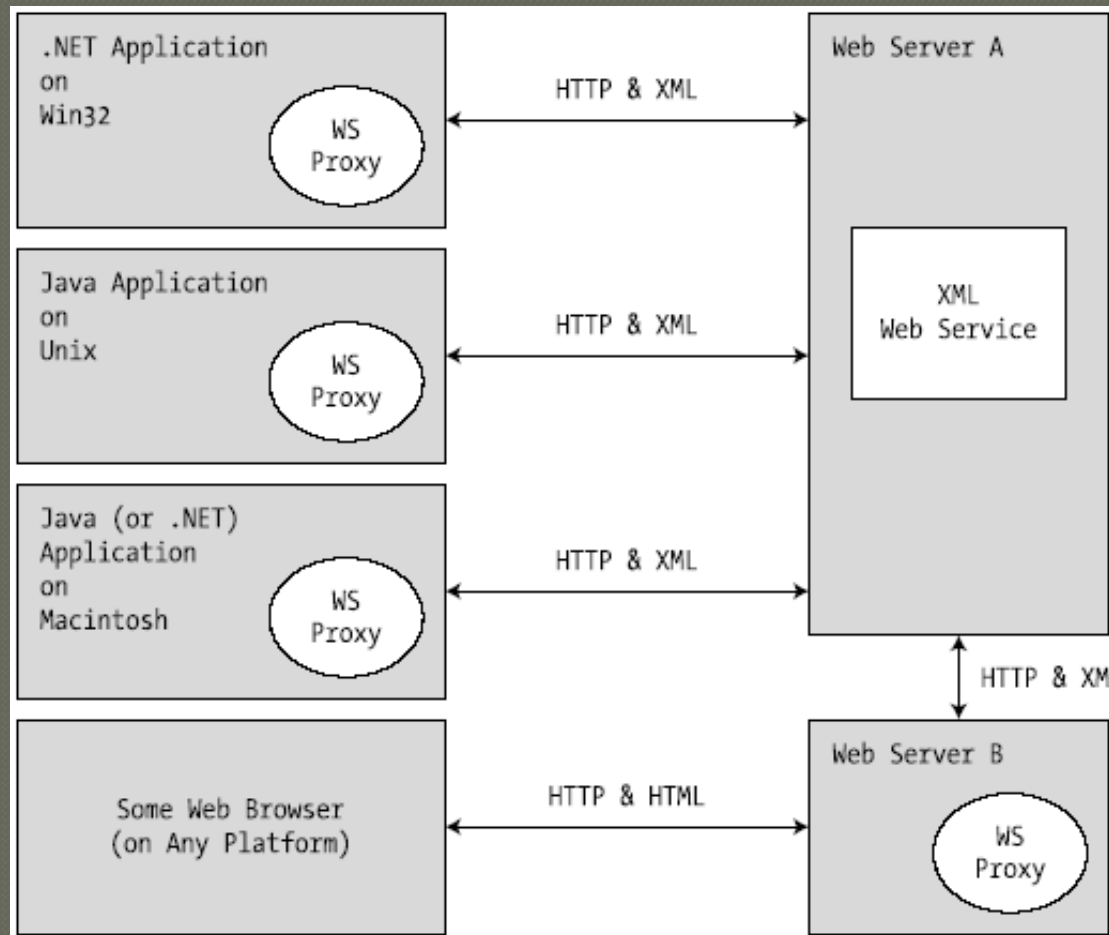
- What is Web service?
 - Web services are typically application programming interfaces (API) or web APIs that are accessed via Hypertext Transfer Protocol (HTTP) and executed on a remote system
- Benefits of XML Web Services
 - XML web services allow you to invoke methods and properties of a remote object using standard HTTP requests.
 - XML web services provide a way for unrelated platforms, operating systems, and programming languages to exchange information in harmony.

XML Web Services Introduction

● XML Web Service Clients

- XML web service client is not limited to a web page
- The proxy's implementation code forwards requests to the XML web service using standard HTTP.
- The proxy also maps the incoming stream of XML back into .NET-specific data types (or whatever type system is required by the consumer application).

XML Web Services Introduction



XML Web Services Introduction

● XML Web Service Components

- An XML web service involves the following core technologies:
 - A discovery service: helps clients resolve the location of the XML web service.
 - A description service: helps clients know what the XML web service can do.
 - A transport protocol: pass the information between the client and the XML web service.

XML Web Services Introduction

● XML Web Service Discovery

- In order to allow client to look up a web service, it must be registered with a Universal Description, Discovery, and Integration (UDDI) server.
- Clients may submit request to a UDDI catalog to find a list of all web services that match some search criteria.
- In addition to UDDI catalog, Client can use *Discovery of Web Services* (DISCO) to locate Web services.

XML Web Services Introduction

- XML Web Service Description

- The client must fully understand the exposed functionality before invoking methods.
- A XML web service is described using *Web Service Description Language (WSDL)*.
 - WSDL will be automatically generated by Microsoft IIS when the incoming request has a ?wsdl suffix.
- Use wsdl.exe to generate proxy type for an XML web service based on a WSDL definition.

XML Web Services Introduction

● Transport Protocol.

- HTTP GET, HTTP POST, or SOAP can be used to move information between consumers and web services.
- SOAP messages can contain XML descriptions of complex types (including your custom types as well as types within the .NET base class libraries).
- HTTP GET or HTTP POST protocols is restricted to a more limited set of core data XML schema types.

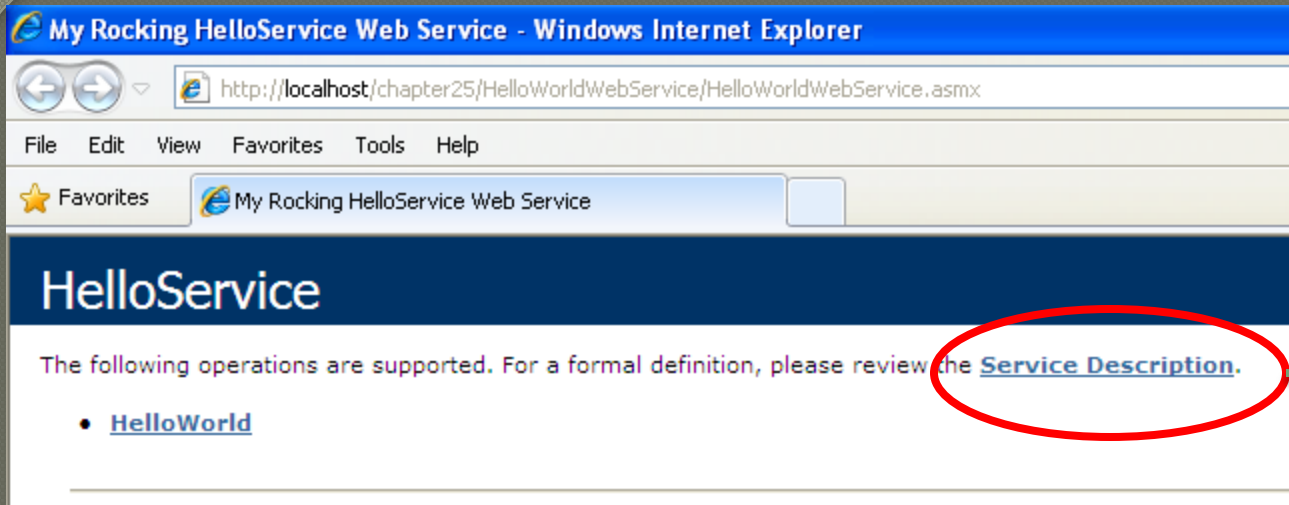
The first web service

HelloWorldWebService.asmx

HelloWorldWebService.asmx

```
<%@ WebService Language="C#" Class="HelloWebService.HelloService" %>
using System;
using System.Web.Services;

namespace HelloWorldService
{
    public class HelloService
    {
        [WebMethod]
        public string HelloWorld()
        {
            return "Hello!";
        }
    }
}
```



My Rocking HelloService Web Service - Windows Internet Explorer

http://localhost/chapter25/HelloWorldWebService/HelloWorldWebService.asmx

File Edit View Favorites Tools Help

★ Favorites My Rocking HelloService Web Service

HelloService

The following operations are supported. For more information, click [here](#).

- [HelloWorld](#)

My Rocking HelloService Web Service - Windows Internet Explorer

http://localhost/chapter25/HelloWorldWebService/HelloWorldWebService.asmx?op=HelloWorld

File Edit View Favorites Tools Help

★ Favorites My Rocking HelloService Web Service

HelloService

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

HelloWorld

Test

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

Invoke

http://localhost/chapter25/HelloWorldWebService/HelloWorldWebService.asmx/HelloWorld

http://localhost/chapter25/HelloWorldWebService/HelloWorldWebService.asmx/HelloWorld

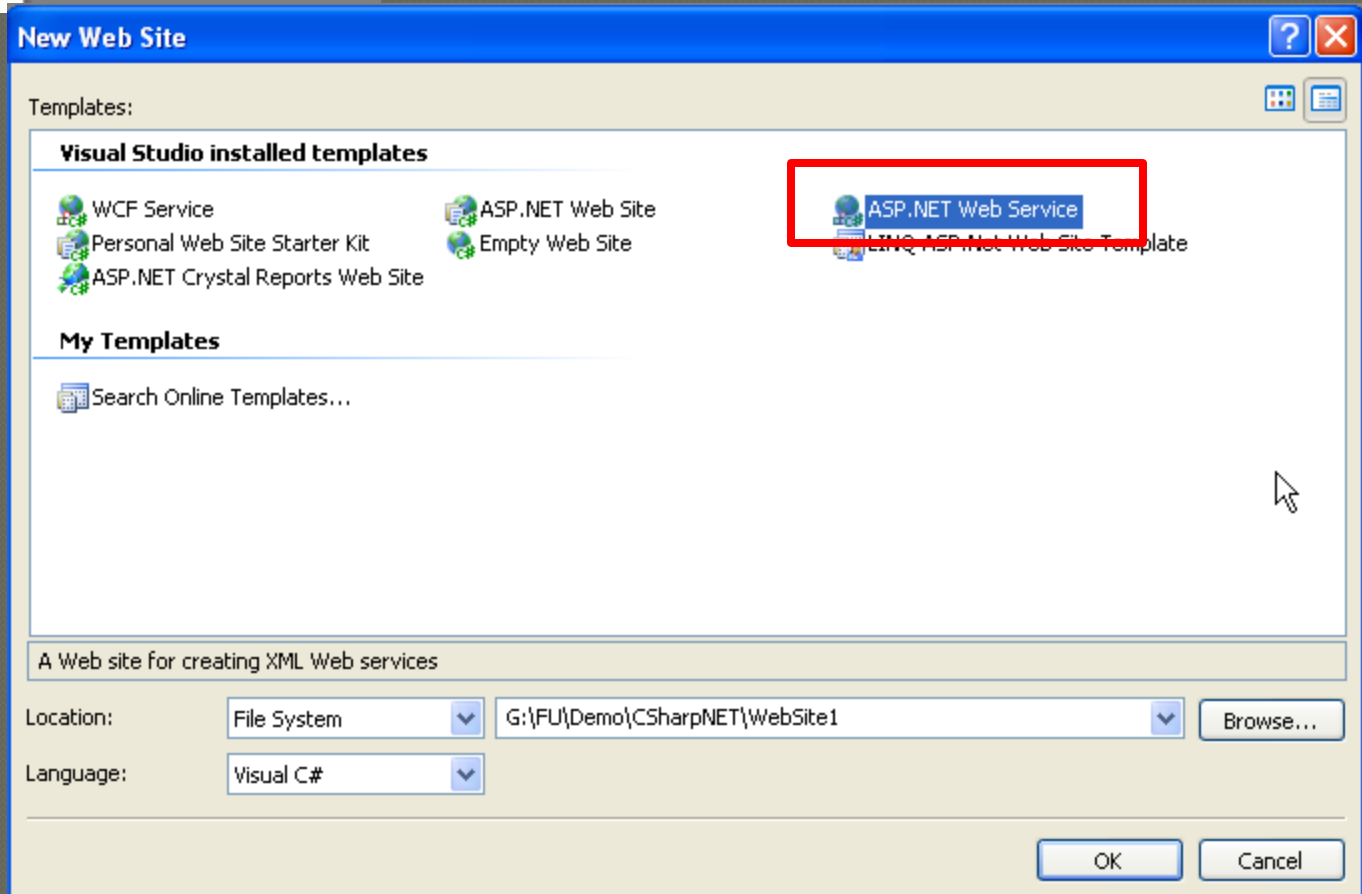
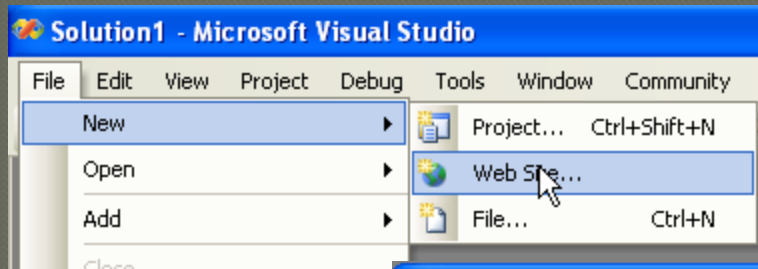
File Edit View Favorites Tools Help

★ Favorites My Rocking HelloService Web...

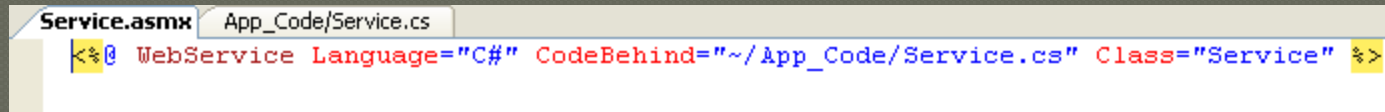
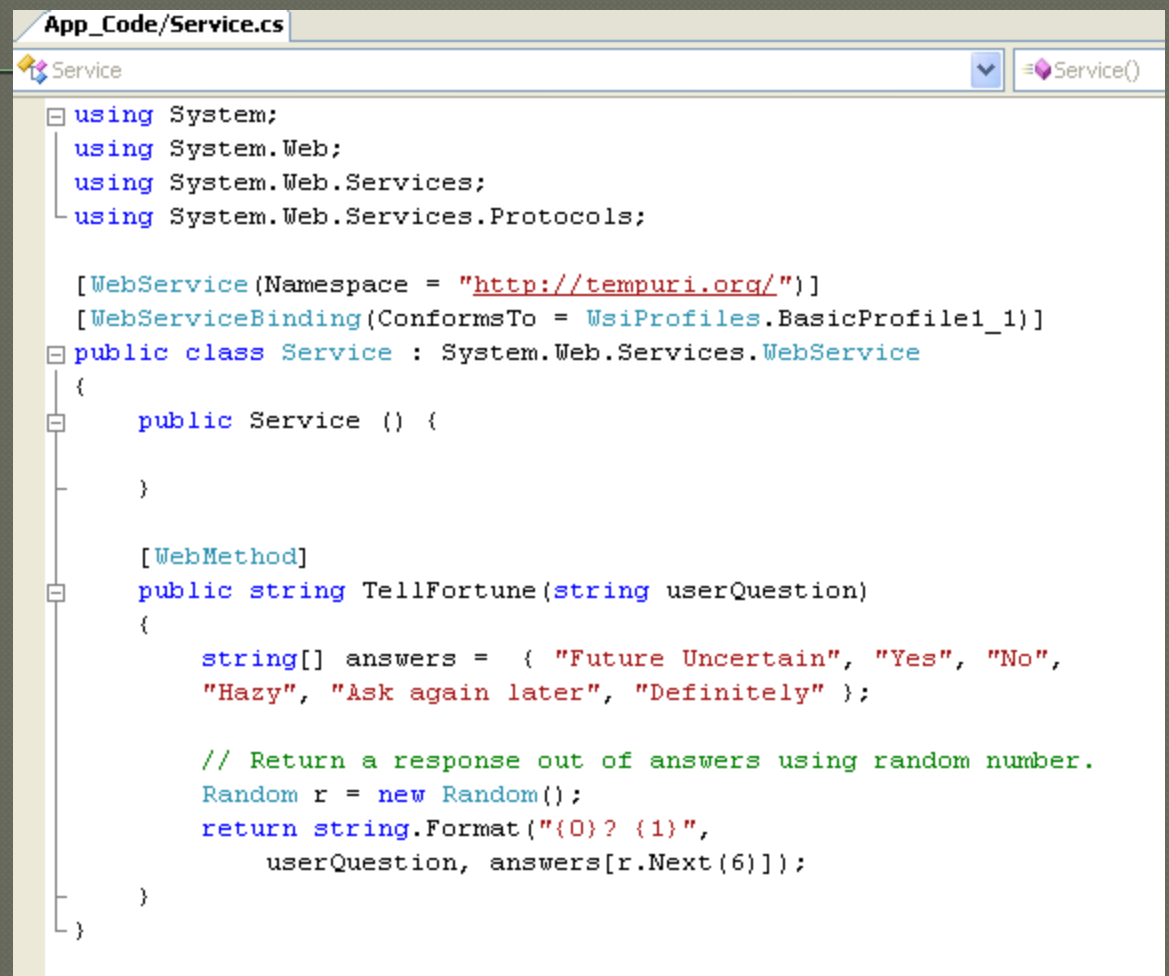
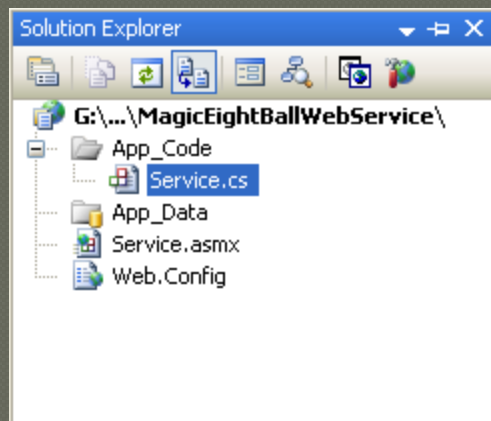
http://localhost/chapter2...

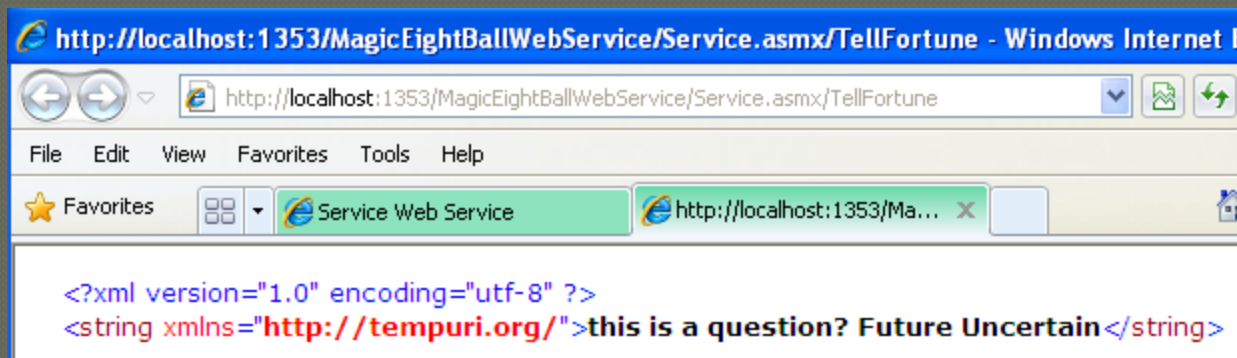
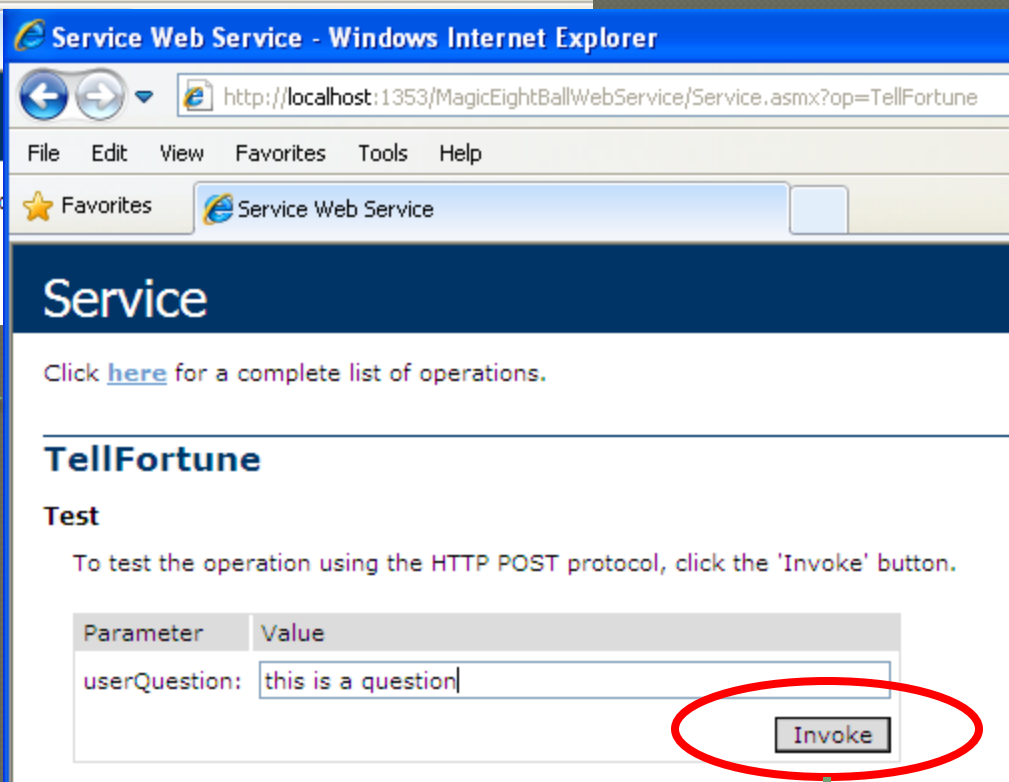
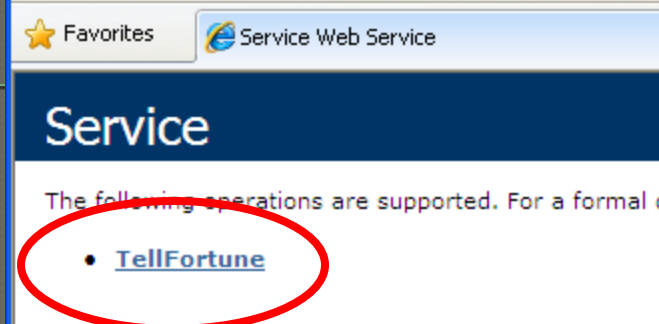
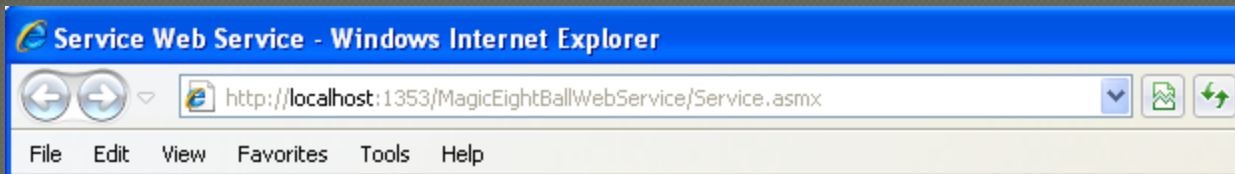
```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://tempuri.org/">Hello!</string>
```

Building an XML Web Service Using VisualStudio 2005



Ex: Ch_25 Code\MagicEightBallWebService





The Role of the WebService Base Class

- A web service can derive directly from System.Object.
- By default, web services developed using Visual Studio 2005 automatically derive from the System.Web.Service.WebService base class.
- Key Members of the **System.Web.Services.WebService** Type

Property	Description
Application	Provides access to the HttpApplicationState object for the current HTTP request
Context	Provides access to the HttpContext type that encapsulates all HTTP-specific context used by the HTTP server to process web requests
Server	Provides access to the HttpServerUtility object for the current request
Session	Provides access to the HttpSessionState type for the current request
SoapVersion	Retrieves the version of the SOAP protocol used to make the SOAP request to the XML web service

The Role of the WebService Base Class

- if you wish to build a stateful web service using **Application** and **Session** variables, you are required to derive from **WebService**.
- if you are building an XML web service that does not require the ability to “remember” information about the external users, **extending** **WebService** is not required.

The [WebService] Attribute

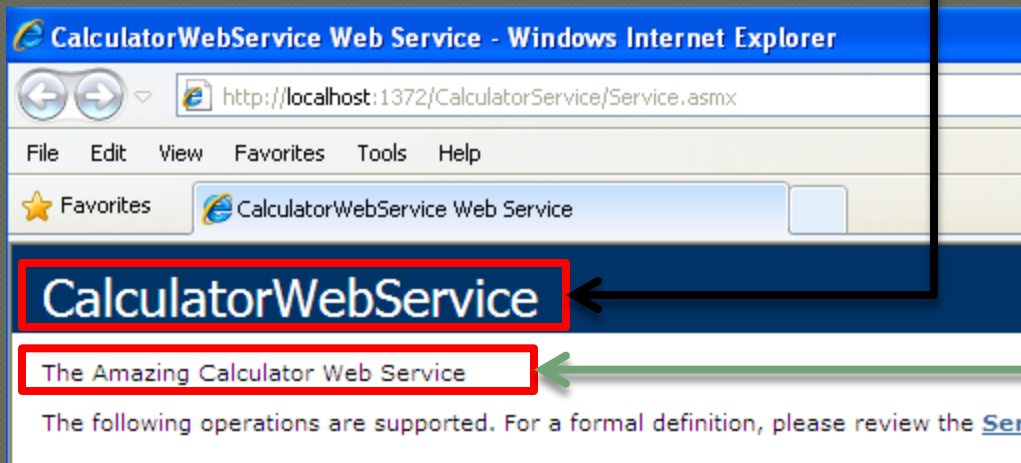
```
App_Code/Service.cs
Service
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

[WebService Description = "The Amazing Calculator Web Service"
    Namespace = "http://www.IntertechTraining.com/",
    Name = "CalculatorWebService"]
[WebServiceBinding(ConformsTo = WsiProfiles.None,
    EmitConformanceClaims = false)]
public class Service : System.Web.Services.WebService
{

```

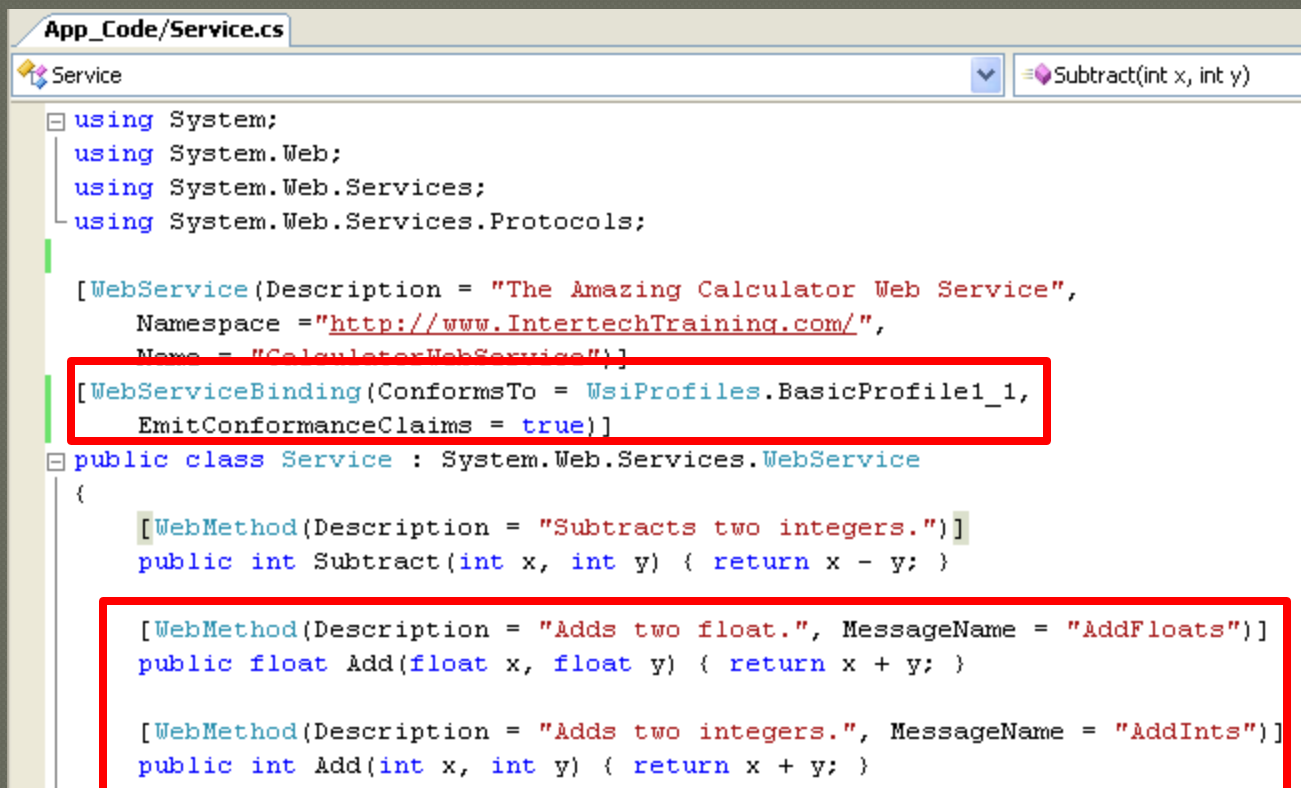
❑ The **Name** Property

- Used to establish the name of the XML web service exposed to the outside world.
- By default, the external name of a web service is identical to the name of the class type itself (Service by default).



[WebServiceBinding] Attribute

- Used to specify if the XML web service conforms to “Web services interoperability (WSI) basic profile 1.1”.
- By default, XML web services generated using Visual Studio 2005 are assumed to conform to the WSI basic profile 1.1.
- *For example: overloading methods are not allowed in WSI basic profile 1.1*
(<http://www.ws-i.org/Profiles/BasicProfile-1.1.html>)



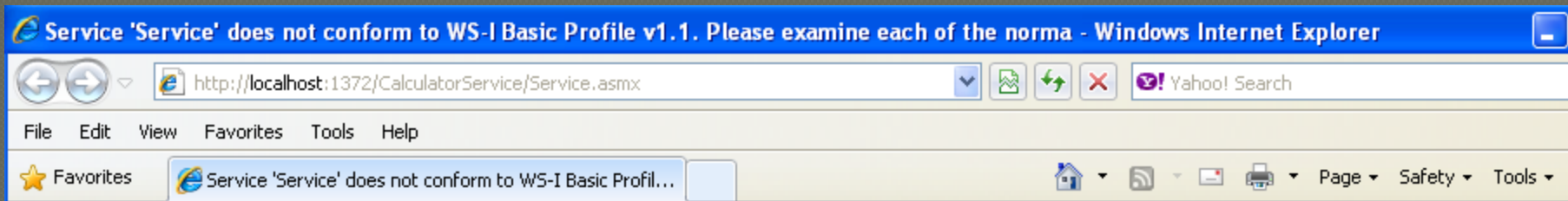
```
App_Code/Service.cs
Service
Subtract(int x, int y)

using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

[WebService(Description = "The Amazing Calculator Web Service",
    Namespace = "http://www.IntertechTraining.com/",
    Name = "CalculatorWebService")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1,
    EmitConformanceClaims = true)]
public class Service : System.Web.Services.WebService
{
    [WebMethod(Description = "Subtracts two integers.")]
    public int Subtract(int x, int y) { return x - y; }

    [WebMethod(Description = "Adds two float.", MessageName = "AddFloats")]
    public float Add(float x, float y) { return x + y; }

    [WebMethod(Description = "Adds two integers.", MessageName = "AddInts")]
    public int Add(int x, int y) { return x + y; }
```



Server Error in '/CalculatorService' Application.

Service 'Service' does not conform to WS-I Basic Profile v1.1. Please examine each of the normative statement violations below. To turn off conformance check set the ConformanceClaims property on corresponding WebServiceBinding attribute to WsiClaims.None.

R2304: Operation name overloading in a wsdl:portType is disallowed by the Profile. A wsdl:portType in a DESCRIPTION MUST have operations with distinct values for their name attributes. Note that this requirement applies only to the wsdl:operations within a given wsdl:portType. A wsdl:portType may have wsdl:operations with names that are the same as those found in other wsdl:portTypes.

- Operation 'Add' on portType 'CalculatorWebServiceSoap' from namespace 'http://www.IntertechTraining.com/'.

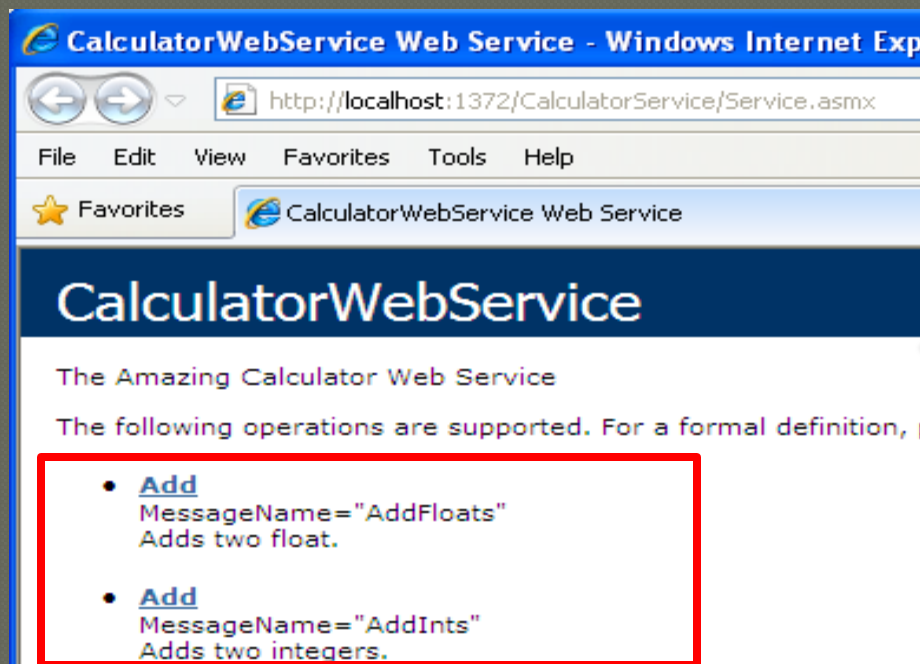
To make service conformant please make sure that all web methods belonging to the same binding have unique names.

Ignoring BP 1.1 Conformance Verification

```
App_Code/Service.cs
Service
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

[WebService(Description = "The Amazing Calculator Web Service",
    Namespace = "http://www.IntertechTraining.com/",
    Name = "CalculatorWebService")]
[WebServiceBinding(ConformsTo = WsiProfiles.None,
    EmitConformanceClaims = false)]
public class Service : System.Web.Services.WebService
{

```

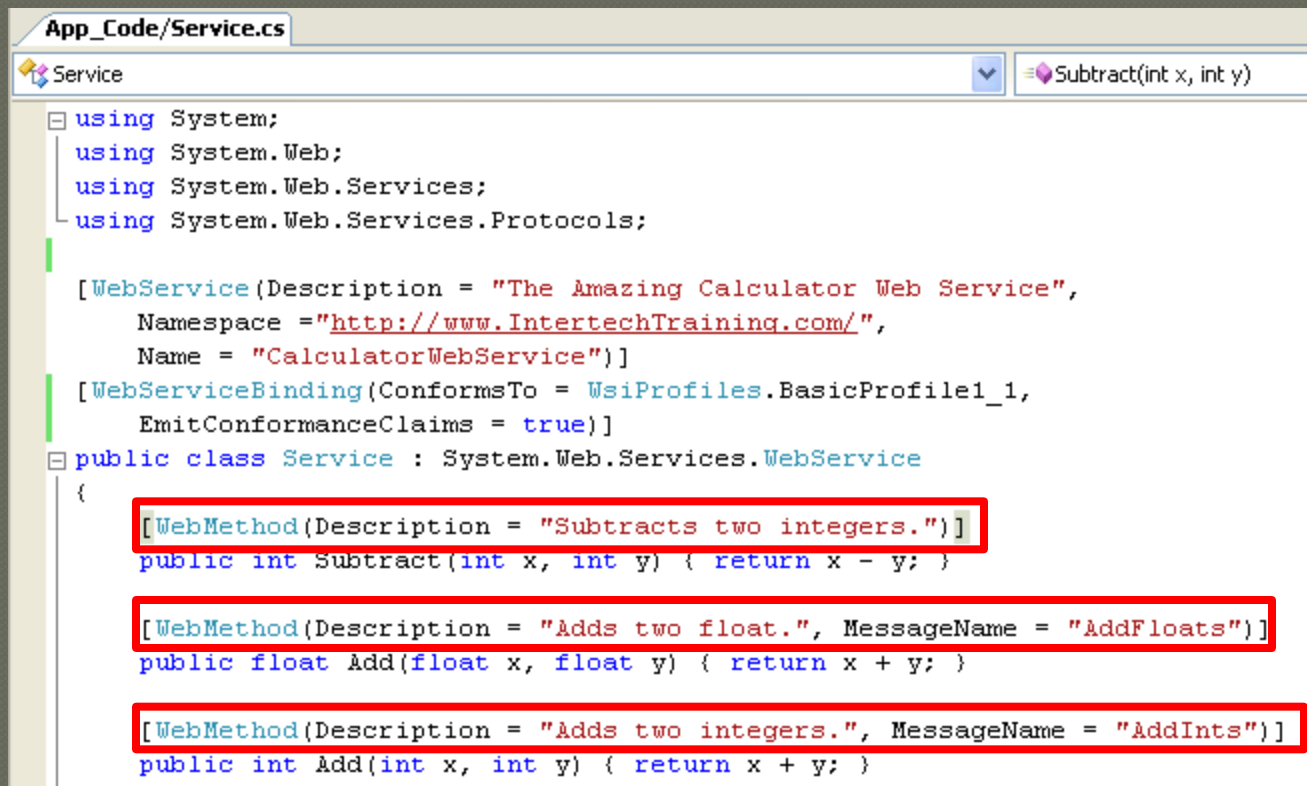


Disabling BP 1.1 Conformance Verification in web.config

```
<configuration>  
  <system.web>  
    <webServices>  
      <conformanceWarnings>  
        <remove name='BasicProfile1_1' />  
      </conformanceWarnings>  
    </webServices>  
  </system.web>  
</configuration>
```


The [WebMethod] Attribute

- The [WebMethod] attribute must be applied to each method you wish to expose from an XML web service



```
App_Code/Service.cs
Service
Subtract(int x, int y)

using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

[WebService(Description = "The Amazing Calculator Web Service",
    Namespace = "http://www.IntertechTraining.com/",
    Name = "CalculatorWebService")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1,
    EmitConformanceClaims = true)]
public class Service : System.Web.Services.WebService
{
    [WebMethod(Description = "Subtracts two integers.")]
    public int Subtract(int x, int y) { return x - y; }

    [WebMethod(Description = "Adds two float.", MessageName = "AddFloats")]
    public float Add(float x, float y) { return x + y; }

    [WebMethod(Description = "Adds two integers.", MessageName = "AddInts")]
    public int Add(int x, int y) { return x + y; }
```

The [WebMethod] Attribute

● Avoiding WSDL Name Clashes via the MessageName Property

```
App_Code/Service.cs
Service
Subtract(int x, int y)

using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

[WebService(Description = "The Amazing Calculator Web Service",
    Namespace = "http://www.IntertechTraining.com/",
    Name = "CalculatorWebService")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1,
    EmitConformanceClaims = true)]
public class Service : System.Web.Services.WebService
{
    [WebMethod(Description = "Subtracts two integers.")]
    public int Subtract(int x, int y) { return x - y; }

    [WebMethod(Description = "Adds two float.", MessageName = "AddFloats")]
    public float Add(float x, float y) { return x + y; }

    [WebMethod(Description = "Adds two integers.", MessageName = "AddInts")]
    public int Add(int x, int y) { return x + y; }
```

Stateful Web Services via the EnableSession Property

```
Global.asax  App_Code/Service.cs
Server Objects & Events (No Events)

<%@ Application Language="C#" %>
<script runat="server">

    void Application_Start(Object sender, EventArgs e)
    {
        Application["SimplePI"] = 3.14F;
    }

    void Session_Start(Object sender, EventArgs e)
    {
        // To prove session state data is available from a Web service,
        // simply assign a random number to each user.
        Random r = new Random();
        Session["SessionRandomNumber"] = r.Next(1000);
    }
</script>
```

Global.asax

This must be set as true if we want to access session variable

```
http://localhost:1372/CalculatorService/Service.asmx
http://localhost:1372/CalculatorService/Service.asmx
File Edit View Favorites Tools Help
http://localhost:1372/CalculatorService/Service.asmx
System.NullReferenceException: Object ref
at Service.GetMyRandomNumber() in g:\F
```

```
Global.asax  App_Code/Service.cs
Service

public int Add(int x, int y) { return x + y; }

[WebMethod(Description = "Get the simple value of PI.")]
public float GetSimplePI()
{
    return (float)Application["SimplePI"];
}

[WebMethod(EnableSession = false,
Description = "Get your random number!")]
public int GetMyRandomNumber()
{
    return (int)Session["SessionRandomNumber"];
}
```

By default each web method has session state disabled

Full example: Ch_25 Code\CalculatorService

Web Service Description Language(WSDL)

- WSDL is used to describe the following characteristics for each exposed web method:
 - The name of the XML web methods
 - The number of, type of, and ordering of parameters (if any)
 - The type of return value (if any)
 - The HTTP GET, HTTP POST, and SOAP calling conventions
- In most cases, WSDL documents are generated automatically by web server.
 - <http://localhost/SomeWS/theWS.asmx?wsdl>
- For more information: <http://www.w3.org/tr/wsdl> .

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions ...>
  <wsdl:types>
    <!-- List of types exposed from WS ->
  </wsdl:/types>

  <wsdl:message>
    <!-- Format of the messages ->
  </wsdl:/message>

  <wsdl:portType>
    <!-- Port information ->
  </wsdl:/portType>

  <wsdl:binding>
    <!-- Binding information ->
  </wsdl:/binding>

  <wsdl:service>
    <!-- Information about the XML web service itself ->
  </wsdl:/service>
</wsdl:/definitions>
```

Web Service Description Language(WSDL)

● **The <types> Element**

- Contains descriptions of any and all data types exposed from the web service.
- XML itself defines a number of “core” data types (<http://www.w3.org/2001/XMLSchema>).

Web Service Description Language(WSDL)

- **Example:** The Subtract() method of CalculatorService. the CLR System.Int32 is described within a <complexType> element:

```
<s:element name="Subtract">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="x" type="s:int" />
      <s:element minOccurs="1" maxOccurs="1" name="y" type="s:int" />
    </s:sequence>
  </s:complexType>
</s:element>
```

- The integer that is returned from the Subtract() method:

```
<s:element name="SubtractResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="SubtractResult" type="s:int" />
    </s:sequence>
  </s:complexType>
</s:element>
```

Web Service Description Language(WSDL)

● Custom data types:

```
public struct Point
{
    public int x;
    public int y;
    public string pointName;
}
```



```
<s:complexType name="Point">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="x" type="s:int" />
    <s:element minOccurs="1" maxOccurs="1" name="y" type="s:int" />
    <s:element minOccurs="0" maxOccurs="1" name="pointName" type="s:string" />
  </s:sequence>
</s:complexType>
```

Web Service Description Language(WSDL)

- The <message> Element

- Used to define the format of the request and response exchange for a given web method.
- It is permissible for a single WSDL document to define multiple <message> elements which are transmitted between the sender and receiver,
- <message> tend to occur in pairs.

```
<wsdl:message name="SubtractSoapIn">
    <wsdl:part name="parameters" element="tns:Subtract" />
</wsdl:message>
<wsdl:message name="SubtractSoapOut">
    <wsdl:part name="parameters" element="tns:SubtractResponse" />
</wsdl:message>
```

Web Service Description Language(WSDL)

- To HTTP POST bindings, the generated WSDL show the following <message> data:

```
<wsdl:message name="SubtractHttpPostIn">
    <part name="n1" type="s:string" />
    <part name="n2" type="s:string" />
</wsdl:message>
<wsdl:message name="SubtractHttpPostOut">
    <part name="Body" element="s0:int" />
</wsdl:message>
```

- If a web method is a one-way method, then it only is necessary a request <message> element of which method can be marked by applying the [SoapDocumentMethod] attribute.

Web Service Description Language(WSDL)

● The <portType> Element

- Defines the characteristics of the various correspondences that can occur between the client and server, each of which is represented by an <operation> subelement.
- EX: The one-way operation, the solicit/response operation.

```
<wsdl:portType name="CalculatorWebServiceSoap">
  <wsdl:operation name="Subtract">
    <wsdl:input message="tns:SubtractSoapIn" />
    <wsdl:output message="tns:SubtractSoapOut" />
  </wsdl:operation>
</wsdl:portType>
```

Web Service Description Language(WSDL)

- The <binding> Element

- Specifies the exact format of the HTTP GET, HTTP POST, and SOAP exchanges.

```
<wsdl:binding name="CalculatorWebServiceSoap12"
              type="tns:CalculatorWebServiceSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="Subtract">
    <soap12:operation soapAction="http://www.IntertechTraining.com/Subtract"
                      style="document" />
    <wsdl:input>
      <soap12:body use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

Web Service Description Language(WSDL)

● The <service> Element

- Specifies the characteristics of the web service itself (such as its URL).
- The chief duty of this element is to describe the set of ports exposed from a given web server.
- The <services> element makes use of any number of <port> subelements.

Web Service Description Language(WSDL)

•The <service> Element

```
<wsdl:service name="CalculatorWebService">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    The Amazing Calculator Web Service
  </wsdl:documentation>

  <wsdl:port name="CalculatorWebServiceSoap"
    binding="tns:CalculatorWebServiceSoap">
    <soap:address location= "http://localhost:1109/CalculatorService/Service.asmx" />
  </wsdl:port>

  <wsdl:port name="CalculatorWebServiceSoap12"
    binding="tns:CalculatorWebServiceSoap12">
    <soap12:address location= "http://localhost:1109/CalculatorService/Service.asmx" />
  </wsdl:port>
</wsdl:service>
```

XML Web Service Wire Protocol

- XML web services can use any RPC protocol to facilitate communication (such as DCOM or CORBA).
- However, most web servers bundle this data into the body of an HTTP request and transmits it to the consumer using one of three core bindings:

Transmission Binding	Description
HTTP GET	GET submissions append parameters to the query string of the URL.
HTTP POST	POST transmissions embed the data points into the header of the HTTP message rather than appending them to the query string.
SOAP	SOAP is a wire protocol that specifies how to submit data and invoke methods across the wire using XML.

XML Web Service Wire Protocol

- HTTP GET and HTTP POST Bindings
 - Not rich enough to represent such complex items as structures or classes.
 - Supported POST and GET Data Types

Data Types	Description
Enumerations	GET and POST verbs support the transmission of .NET System.Enum types
Simple arrays	You can construct arrays of any primitive type.
Strings	GET and POST transmit all numerical data as a string token.

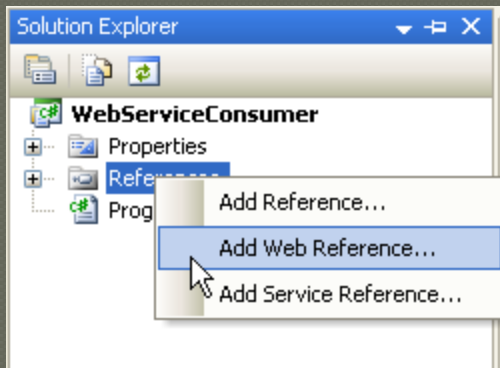
XML Web Service Wire Protocol

● SOAP Bindings

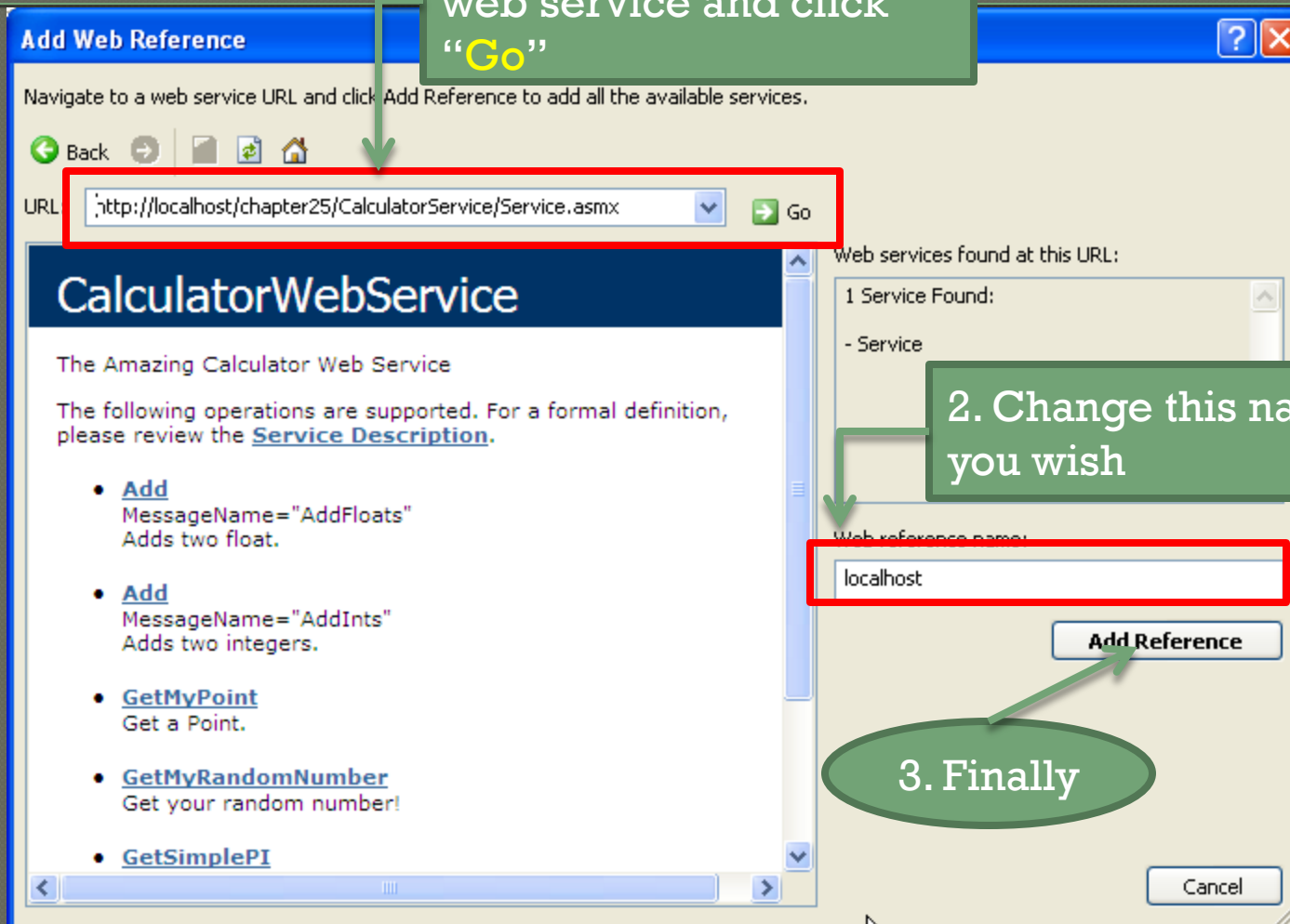
- SOAP itself does not define a specific protocol and can thus be used with any number of existing Internet protocols (HTTP, SMTP, and others).
- Provide a mechanism to invoke methods using complex types in a language- and platform-neutral manner.
- A SOAP message defines two core sections:
 - SOAP envelope: understood as the conceptual container for the relevant information.
 - SOAP body: contains rules that are used to describe the information in said message.
 - An optional third section (the SOAP header) may be used to specify general information regarding the message itself, such as security or transactional information.

Building Client application

Create a console application consuming web service
(Ch_25 Code\CalculatorService)

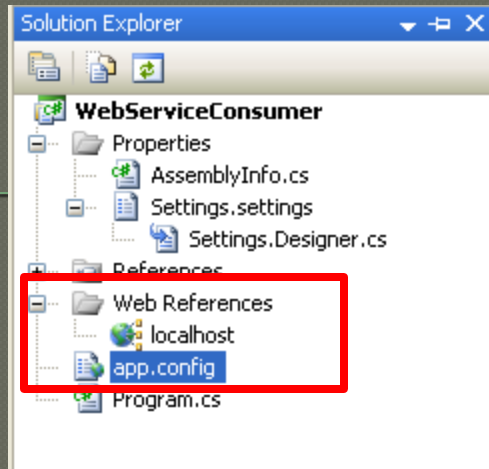


1. Input the url of your web service and click "Go"



2. Change this name if you wish

3. Finally



If the url location of the web service changed, we just need to modify this value

```
app.config Program.cs
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <sectionGroup name="applicationSettings" type="System.Configuration.ApplicationSettingGroup" base="System.Configuration.ApplicationSettingsGroup">
      <section name="WebServiceConsumer.Properties.Settings" type="System.Configuration.ApplicationSettingsGroup" base="System.Configuration.ApplicationSettingsGroup">
        </section>
      </sectionGroup>
    </configSections>
    <applicationSettings>
      <WebServiceConsumer.Properties.Settings>
        <setting name="WebServiceConsumer_localhost_CalculatorWebService"
          serializeAs="String">
            <value>http://localhost/chapter25/CalculatorService/Service.asmx</value>
          </setting>
      </WebServiceConsumer.Properties.Settings>
    </applicationSettings>
  </configuration>
```

```
using System;
using System.Collections.Generic;
using System.Text;
using WebServiceConsumer.localhost;
```

```
namespace WebServiceConsumer
```

```
{
    class Program
    {
        static void Main(string[] args)
```

```
        {
            Console.WriteLine("***** Fun with WS Proxies *****\n");
```

```
            // Make the proxy.
```

```
            CalculatorWebService ws = new CalculatorWebService();
```

```
            // Call the Add() method synchronously.
```

```
            Console.WriteLine("10 + 10 = {0}", ws.Add(10, 10));
```

```
            // Call the Subtract method asynchronously
```

```
            // using the new .NET 2.0 event approach.
```

```
            ws.SubtractCompleted += new SubtractCompletedEventHandler(ws_SubtractCompleted);
```

```
            ws.SubtractAsync(50, 45);
```

```
            Console.ReadLine();
        }
    }
}
```

```
static void ws_SubtractCompleted(object sender, SubtractCompletedEventArgs e)
```

```
{
    Console.WriteLine("Your answer is: {0}", e.Result);
}
```

Synchronous call

Asynchronous call

C:\ file:///G:/FU/Demo/CSharpNET/WebServiceC

***** Fun with WS Proxies *****

10 + 10 = 20

Your answer is: 5

More example:

Ch_25 Code\CarSalesInfoWS,

Ch_25 Code\CarsSalesInfoClient

Summary

- XML Web Services introduction
- Web Service components: Discovery, Description, Transport protocol
- Understanding the [WebService] Attribute
- Understanding the [WebServiceBinding] Attribute
- Understanding the [WebMethod] Attribute
- XML Web Service Wire Protocols: HTTP GET, HTTP POST, SOAP

Chapter 25: Q & A