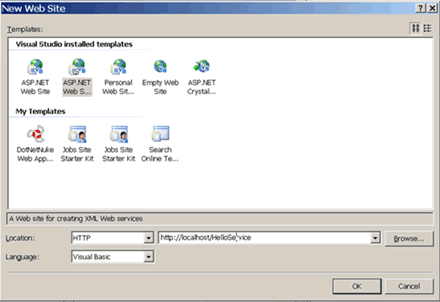
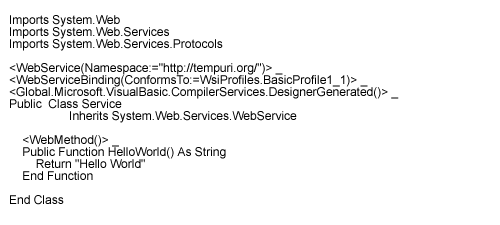
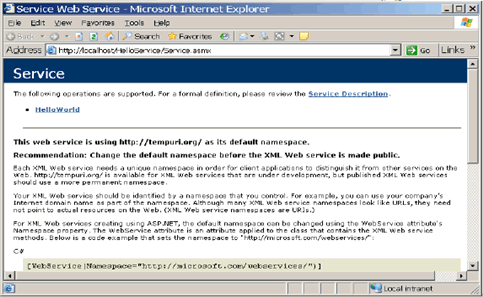
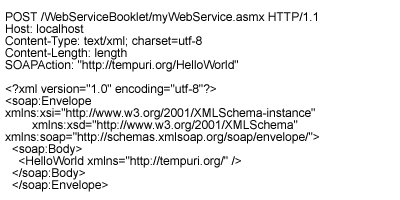
How to create a Web service using Visual Studio.net

Creating a Web service in Visual Studio .NET is very simple. ASP.Net server side technology allows developers to create web services and the .NET framework provides a sophisticated set of tools and code to consume web services.   
  
In this exercise, we will build a simple web service ?Hello World'. Most articles teach about web services using the Hello World example. But few explain the technology behind this web services example in detail. In this article we will learn both.   
  
Let us start by creating the Hello world web service. We will create the web service using asp.net and visual basic language. The IDE we will use is Visual Studio. Net. We can also create a ASP.NET Web Service in C# programming language using visual studio. Net IDE.   
  
  
  
To create the web service, open Visual Studio and select File-> New Web site from the menu. Make sure Visual Basic is selected in the language box and HTTP in the location box. Select ASP.NET Web service from the template list. Enter the name as http://localhost/HelloService as shown in above figure and click OK.   
  
Visual Studio 2005 will create a new web service project. A new virtual directory - HelloService - will be created on your computer's Web server. This action also creates two files called Service.asmx and service.vb. Please note that Visual Studio creates the entry point file with .asmx extension and the other code behind file with .vb extension. These files are automatically added to the project, which we can see in a solution explorer.   
  
The service.asmx amd service.vb files represent a single web service. WebServices do not have a visible interface and so, the design interface is empty. When we open the Service.asmx file, only the following code is seen.   
  
http://www.vkinfotek.com/webservice/createwebservice1.gif  
  
The service.asmx file serves as the entry point into the web service.   
  
The other file, service.vb, is the code-behind file for the web service. We can write the functionality of the web service in this file. The code-behind file imports the System.Web.Services namespace. This namespace contains the classes that are required to build and use Web services. We can view this file by switching to the code view for the service.asmx file.   
  
Note that a single project can contain multiple WebServices, each web service implemented by a different class.   
  
If we open the service.vb code-behind page, we can see the default code which is created by vs.NET automatically, as listed below.   
  
  
  
From the above code, we see that service is the class that contains all the code that implements the web service methods. This is actually the name of the WebService; it's named after the class and not after the project. The .vb and .asmx files together form the Code model for the web services.   
  
As we have already seen a web service is basically a class that sits on the server. Building a Web Service with Visual Studio is as simple as writing a class. The class inherits from System. Web. Services, and this base class includes all that is necessary to make the functions (methods) of the class available on the Web. The Web Service class belongs to the system.Web.Services namespace. This class provides direct access to built-in ASP.NET objects, such as Application and Session. If an application does not require these objects, we can create a Web service without deriving it from the Web Service class. So, Note that deriving from a web service base class is optional.   
  
Examining the Web Service Files   
  
Service.asmx file   
  
http://www.vkinfotek.com/webservice/createwebservice1.gif  
  
In the above line of code, @ WebService is the web processing directive that specifies a class name, here service for a web service and the language used to create the web service is visual basic.   
  
Service.vb file   
  
Explanation of the WebService attribute   
  
The WebService attribute is an optional attribute that can be used to provide additional information about the Web service. We can also use the WebService attribute to provide a unique namespace for the web service. This namespace can be used by client applications to resolve conflicts that may arise due to different Web services having methods with the same name. The Webservice attribute can be provided before the Web service class declarations as shown below.   
  
  
  
The description of the Web service appears in the browser window when we try to access the web service directly by providing the path.   
  
The WebMethod Attribute   
  
The methods that we want to expose to web service are marked with Webmethod attribute.   
  
The Webmethod attribute enables us to call remotely, the web service methods over the internet. The syntax for applying the Webmethod attribute to a method is given below.   
  
syntax for applying the Webmethod attribute to a method is given below.   
  
http://www.vkinfotek.com/webservice/createwebservice5.gif  
  
The methods of this class don't return an HTML page, instead, they return one or more values, packaged as XML documents.   
  
Basically, if you can write a function, you can write a web method; it's that simple. The difference between a class and a WebService class, is that the members of the WebService can be called remotely.   
  
VB.NET uses less than and greater than symbols to the attribute to appear on the line before the function definition.   
  
From the above code, we note that class name is Service and method is HelloWorld.   
  
If we want to change its name, we must rename the Service item in the Solution Explorer's window, as well as the name of the class in the code.   
  
Note that we can also create methods without preceding them with the WebMethod attribute. However, these methods cannot be used to expose some functionalities to applications. They can only be used by the other methods within the Web service class. 

Testing the web services .Net

Testing a web service is checking whether the .asmx file we create using visual studio.net is bug free and without any errors. We test the .asmx file and then we upload it on to the web server.   
  
Before we start with test with web service, we should note that a web service does not have a interface. To over come this, .Net CLR creates a small site that enables us to test each method of a web service.   
  
After we create the web service, we press F5 to build and run the web service. If successful, internet explorer will come up displaying the page containing list of the methods as well as a link to the Service Description as shown in below figure.   
  
  
  
The Service Description link (shown in the top part of the picture) provides a description of all the methods, such as the number of parameters and their data types, the sequence of the parameters, and the return value of the methods exposed by the Web service.   
  
In this case, this page displays the methods of the WebService, HelloWorld and warning to the effect that it uses the default namespace. After you decide to make the WebService available to others on the Internet, you should change the namespace to something like http://yourserver.com/ (where your Server is the name of the Web Server where the code resides).   
  
By clicking on a method name, HelloWorld, we can see information about this particular method. This information includes the format of the SOAP request message it expects as well as the format of the SOAP response message it will send, and also provides a Web Form to test the Web service. A SOAP request message is expected by the Web service.   
  
Recall that this is the message the client will pass to the Web service to indicate that it wants to invoke the HelloWorld() method.   
  
The SOAP messages which travel from the client to the web service and back can be categorized as   
  
1.SOAP request message 2.SOAP response message   
  
Demystifying the SOAP message   
  
SOAP Request message   
  
SOAP documents are constructed with XML. Now we will go little deep into the SOAP document. All SOAP messages must adhere to a basic structure, which identifies the messages with XML namespaces. An XML namespace provides a mechanism to identify XML data types, as well as qualify both your element and attribute names. The XML namespace declarations are specified with the xml ns prefix.   
  
XML namespaces enable us to distinguish between element names to avoid name collisions of elements that use the same name. At the start of a SOAP message, we should include a URL that references an XML schema that will define the structure of your message.   
  
Although not required, we should point to a valid XML schema within a SOAP envelope. You can think of the Envelope tag as the root element of a SOAP message, specified with the XML start and end tags.   
  
Change the fig and show with SOAP request/response documents with helloworld service.   
  
Note that the SOAP Body will need to contain first an element named .   
  
  
  
In the above code, in the statement , the http://tempuri.org/ is the Web Service Namespace.   
  
From the above code we can make out that the child element of the is the element. This specifies what Web service method to call.