ASP.NET Web Services and Web Clients

Web Services Overview

- The World Wide Web has opened up the possibility of large-scale **distributed computing**
- Web Applications only allow interaction between a client browser and web server hosting a web page
- Web Service: A web-based program that exposes member functions that other programs running on other computers can call.
  - AWeb Service has no user interface
- Web Client: A program that consumes a web service (calls the functions of a web service)
  - Could be a Web Form, a Windows Form, or even a command line application

Some Examples of Web Services

- There are lots of them out there
- http://seekda.com has a great Web Services search engine

ASP.NET Web Services

- Before ASP.NET, distributed computing was highly dependent on OS and language
- ASP.NET web services and clients are entirely independent of either
  - Could have a web service written in VB running on Windows 2000 consumed by a web client written in C++ running on a UNIX box
- What is needed?
  - Both client and server must use industry standard protocols
    - SOAP – Simple Object Access Protocol: a lightweight object-oriented communication protocol based on XML

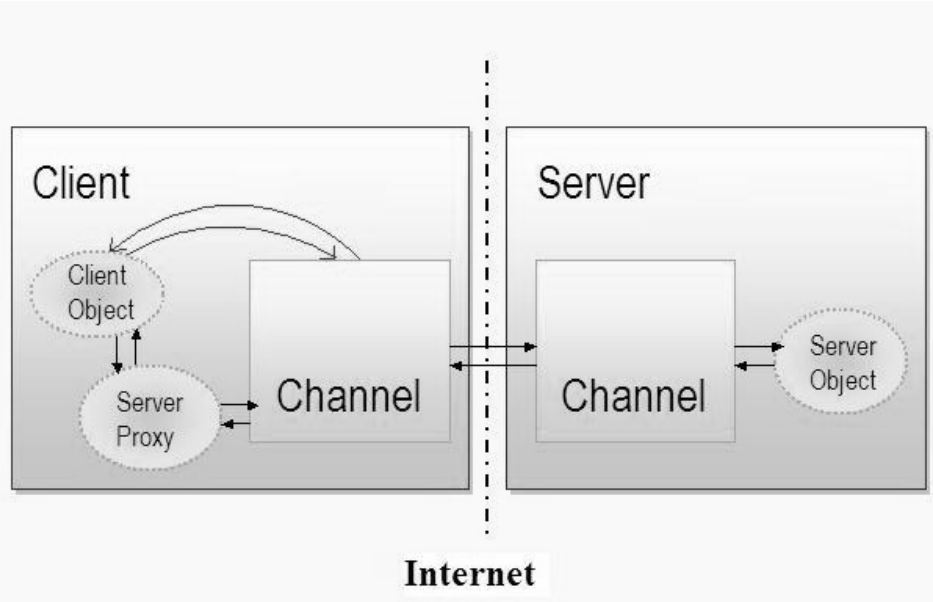
How Web Services Work

- A web service contains one or more functions or methods called over the internet
  - Clients call exposed methods of the web service using standard internet protocols
  - Both client and server must be connected to the internet
  - Data format used for requests is usually SOAP
    - Self-describing text-based XML documents
    - Only requirement is that both server & client be able to send & receive messages that conform to the proper protocol standard

Sequence of Events

- Client makes a call to the web service method
  - It appears as though it's talking directly to the web service over the internet
  - But the actual call is being made to a "proxy class" local to the client
    - Proxy is a substitute or stand-in for the actual code to be called
    - An object that provides a local representation of a remote service
    - It's really a DLL that handles all the complexities of encoding & sending requests over the internet and getting responses back
    - It "marshalls" the call to exposed methods across the internet
    - Proxy class object must be created by the client app
      - Done by Visual Studio when you create a "web reference"
      - Actually it's done by the Wsdl.exe (Web Services Description Language) utility program

Web Service/Client Interaction



Writing Web Services

- Use the .NET Framework
  - Easy: ASP.NET does most of the work for you
  - Store code in a .asmx file
  - .asmx file begins with <@ WebService…> directive
    - Must identify a Class encapsulating the web service
    - Class definition has a [WebService…] attribute to assign a name and description of the service
    - Each class method has a [WebMethod…] attribute that describes the functionality of the method
  - Can be done manually or with VS Designer

Manual coding of a Web Service

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

[WebService (Name="Add Service", Description="Adds two integers over Web")]
class AddService
{
    [WebMethod (Description="Computes sum of two integers")]
    public int Add(int a, int b)
    {
        return a+b;
    }
}
```

- Store this with a .asmx extension in the default IIS directory (c:\inetpub\wwwroot)
  - e.g., AddService.asmx

Testing the Web Service

- 1. Just call it up in a browser
  - http://localhost/AddService.asmx
  - ASP.NET responds to the HTTP request by generating an HTML page for the browser
    - Name and description of the service appear
    - Also the names of methods provided by the server that, when clic ked, allow the user to test them
    - Also a link to a WSDL (Web Services Description Language) XML document describing in detail the "service contract"
      - This is an HTML document with ?wsdl at the end of its URL
- 2. Or write a .NET client program to use the service
  - e.g., AddClient – a Windows Form application
  - Must add a Web Reference to the AddService.asmx web service
    - Proxy class is generated ASP.NET
  - And invoke its Add(…) method after instantiating the proxy class object

AddClient Code

```
localhost.AddService myaddservice = new localhost.AddService();
        //This is the proxy class object
int z = myaddservice.Add(x, y);
```

Creating a Web Service w/ Visual Studio

- Using IIS (if not use the Visual Web Developer)
  - "File" | "New" | "Web Site" | "ASP.NET Web Service"
    - "Project Type": C#
    - "Location": HTTP, http://localhost/WebserviceName
      - Project directory will be put in the home (Inetpub\wwwroot) directory of your IIS server
  - Creates Service.asmx file
    - Executed by IIS
      - Gives access to the web service
      - Specifies the implementation class of the web service
  - And Service.cs file
    - contains skeleton C# code for the web service
    - Note the "WEB SERVICE EXAMPLE HelloWorld()"
      - Comment it out or remove it
    - Just add the methods you want the service to expose at that place in the Service.cs file
  - Change its name (Service) everywhere it appears:
    - class name, constructor, also twice in .asmx file
    - Also rename the two files

Example Web Service: ConvertTemperature

- Has temperature conversion methods ctf() and ftc():

```
[WebMethod (Description="Converts a Centigrade temperature to Fahrenheit")]
public float ctf(float ctemp)
{   return (1.8F * ctemp + 32.0F); }
[WebMethod (Description="Converts a Fahrenheit temperature to Centigrade")]
public float ftc(float ftemp)
{   return ((5F/9F)*(ftemp - 32.0F));  }
```

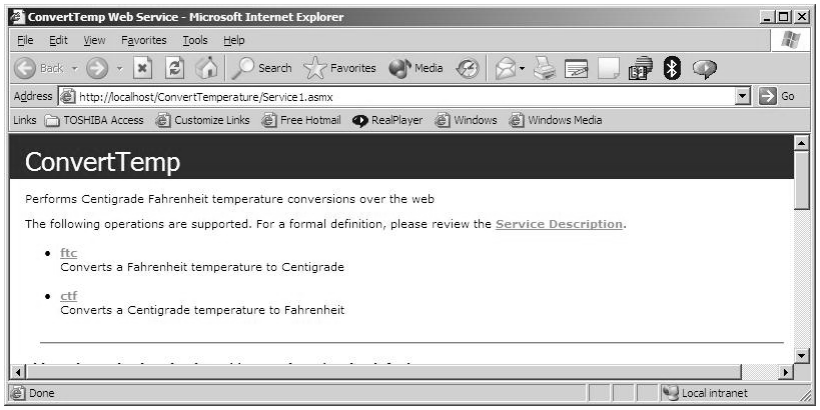
  - Note use of [WebMethod] attribute
    - Specifies that these methods are available to be used by web clients
    - Description will appear if service is tested in a browser
- Modify top line of file: the [WebService] attribute

```
[WebService (Namespace = "http://tempuri.org/", Name="ConvTemp2008", Description =
    "Performs Centigrade Fahrenheit temperature conversions over the web")]
```

    - tempuri: Temporary Uniform Resource Identifier (name)
    - Default namespace used by VS to distinguish this service from others on web
  - "Name" and "Description" will appear in the HTML page generated when user calls up the service in a browser
    - "Name" determines name of Proxy class created by client

Running and Testing the Web Service

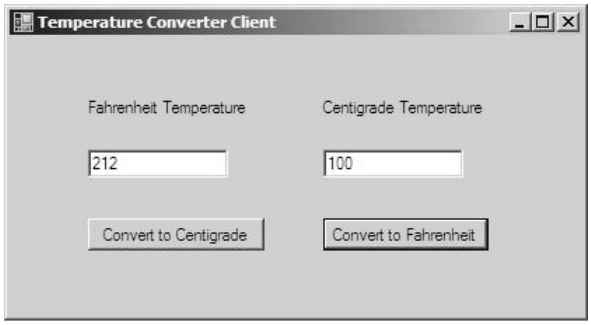
- Run the Web Service from Visual Studio just as for any other application
  - "Debug" | "Start without debugging"
  - Brings up the following web page in your browser:



  - Clicking on ftc or ctf allows you to test the service's methods

Creating a Web Client for the Service

- Can use Visual Studio to build a Windows Form or Web Form application to use the Web Service
- Example "ConvertTempClient"
  - A Windows Form app



  - User enters Fahrenheit or Centigrade temperature in a textbox
  - Presses appropriate button
  - Other textbox will contain the converted temperature

Using Visual Studio to Create a Web Client that consumes a Web Service on the local computer

- Start a Windows Application project as usual
- Drag the controls over to the form and rename them as usual
- Add a Web Reference:
  - In Solution Explorer, right click on References
  - Click on "Add Web Reference", or "Project | Add Web Reference"
    - "Add Reference Browser" page comes up
    - Select "Web Services on the Local Machine" and choose the ConvertTemp service
    - Click "Add Reference" button
  - A new "Web References" folder also was created
    - Contains a node name after the domain name where the Web service is
  - Also notice in Class View that under {} localhost, a ConvertTemp class has been added
    - This is the proxy class and contains the local representations of the ftc and ctf methods

Web Client Creation: Coding

- Double click the Convert Fahrenheit to Centigrade button and add the following button click event handler code

```
localhost.ConvTemp2008 obj = new localhost.ConvTemp2008();
string fstr = textBoxFahr.Text;
float ftemp = float.Parse(fstr);
float ctemp = obj.ftc(ftemp);
textBoxCent.Text = ctemp.ToString();
```

- Double click the Convert Centigrade to Fahrenheit button and add the following button click event handler code

```
localhost.ConvTemp2008 obj = new localhost.ConvTemp2008();
string cstr = textBoxCent.Text;
float ctemp = float.Parse(cstr);
float ftemp = obj.ctf(ctemp);
textBoxFahr.Text = ftemp.ToString();
```

- When you run the program, it will use the web service to perform the temperature conversions