INTERNET & WEB APPLICATION DEVELOPMENT SWE 444

Fall Semester 2008-2009 (081)

Module 8: Web Services

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives/Outline

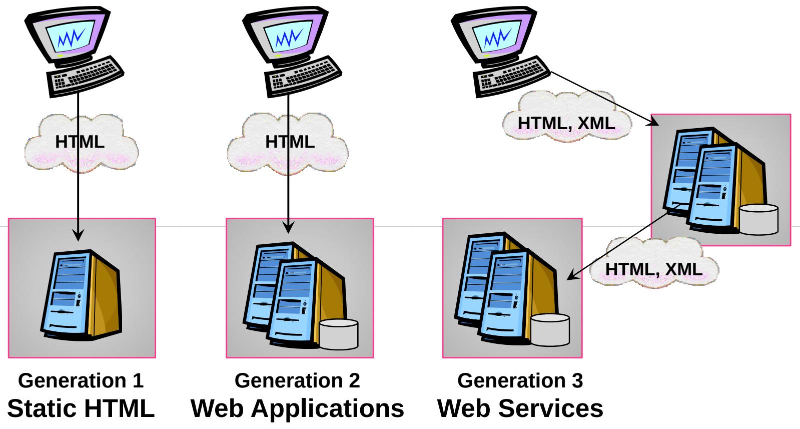
- Objectives
  - Learn about Web Services: What, Why and How

- Outline
  - Introduction
  - Example Web Services
  - Web Services vs Web Applications
  - Benefits of Web Services
  - Underlying Technologies
  - Developing a Web Service
  - Consuming Web Services

Evolution of the Web

- First generation of Web applications (called brochure web)
  - Largely about delivering non-interactive content, i.e. publishing non-interactive HTML pages.
  - For example, many applications simply operated in client/server mode and rendered HTML pages to send across the internet to browsers.
- Second generation of Web applications
  - About creating applications usable over the Web.
    - E-commerce is an example; you can go to http://www.barnesandnoble.com/ select books, order them and pay for them.
  - Second generation also includes a more scalable back-end and a richer UI (e.g. DHTML).
  - However, the second generation largely resulted in application islands on the Web.
    - Yes, there are hyperlinks between sites, but for the most part, the actual applications at different sites do not interact.
- Third generation of Web applications
  - About using Web protocols and XML throughout to allow better integration between services on the Web.
  - Protocols such as XML and SOAP allow you to create Web Services, enabling people and companies to easily create integrated applications.

Evolution of the Web (cont.)



What is a Web Service?

- A Web Service (sometimes called an XML Web Service) is an application that enables distributed computing by allowing one machine to call methods on other machines
  - Via common data formats and protocols such as XML and HTTP
  - Web Services are accessed, typically, without human intervention
- In .NET and other frameworks these method calls are implemented using the Simple Object Access Protocol (SOAP)
  - An XML-based protocol describing how to mark up requests and responses so that they can be transferred via protocols such as HTTP
  - Using SOAP, applications represent and transmit data in a standardized format—XML
- In its simplest form, a Web Service is a class whose methods are callable remotely
- Methods in a Web Service are executed through a Remote Procedure Call (RPC)
- Method calls to and responses from Web Services are transmitted via SOAP
  - Thus any client capable of generating and processing SOAP messages can use a Web Service regardless of the language in which the Web Service is written

What is a Web Service?

- Web services are application components
- Web services communicate using open protocols
- Web services are self-contained and self-describing
- Web services can be discovered using UDDI
- Web services can be used by other applications
- XML is the basis for Web services

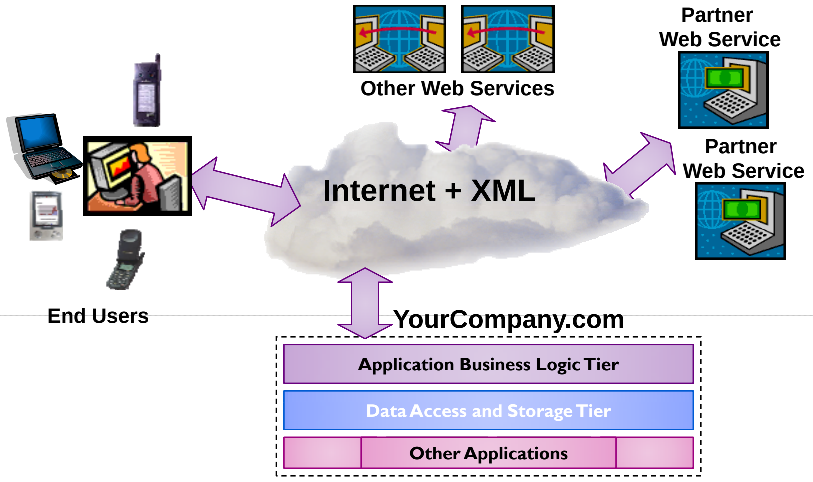
Why Web Services?

- Web services are used to address the fundamental challenge of distributed computing:
  - locating and accessing remote systems
- There are many scenarios where computers need to access and share data with others remotely
  - Businesses need to share information with partners
  - Divisions need to send data to other divisions
  - Consumer applications need to work with other consumer applications
- This challenge has been around for decades
  - Solutions provided using CORBA, COM/DCOM, Java RMI, etc
- The big difference is that Web Services address the problem using open technologies
  - Web Services use XML to code and decode data and use SOAP to transport it using open protocols
  - Platform independence opens up the opportunity for heterogeneous systems to access Web services

Problems with Existing Technologies

- With distributed object models (CORBA, DCOM etc), applications can be built that can send data back and forth
  - CORBA: Common Object Request Broker Architecture
  - DCOM: Distributed Component Object Model
- However, many of these technologies suffer from the following limitations:
  - Not Internet-friendly
  - Interoperability issues: poor/non-existent standards
  - Tightly coupled: cannot integrate with other applications
- DCOM, based on COM, is a binary standard that has virtually no deployment outside of MS Windows platform
- Although CORBA can (and does) work in heterogeneous environments, many developers find it difficult to implement CORBA-based applications and services

Application Model



Sample Web Services

- Google search
  - Incorporating a search engine seamlessly into one's site is a feature very commonly needed.
- Package tracking:
  - A Web service provided by a delivery company like UPS, FedEx
  - A book seller like Amazon.com can incorporate package-tracking capability
- News (RSS) feed
  - An online news provider could provide a Web service to all the Web sites publishing the news
- Currency conversion
- Weather forecast

Web Applications versus Web Services

Web Applications	Web Services
Human clients	Clients are other applications
Need to have some user interface	User interface not needed
Usually homogenous	Can connect heterogeneous systems
Tightly coupled	Loosely coupled

Benefits of Web Services

- Facilitates B2B transactions
  - By purchasing Web Services relevant to their business, companies can spend less time coding and more time developing new products
  - Thus Web Services make it easier to businesses to collaborate and grow together
- Extend the benefits of object-oriented programming
  - Instead of developing software from a small number of class libraries provided at one location, programmers can access countless libraries in multiple locations
- Benefits various systems (not just Web browsers)
  - Including slow systems, those with limited amounts of memory or resources, those without access to certain data, and those lacking the code necessary to perform specific computations
- Code and data can be stored on another computer
  - The client does not need to access or store the database on its machine

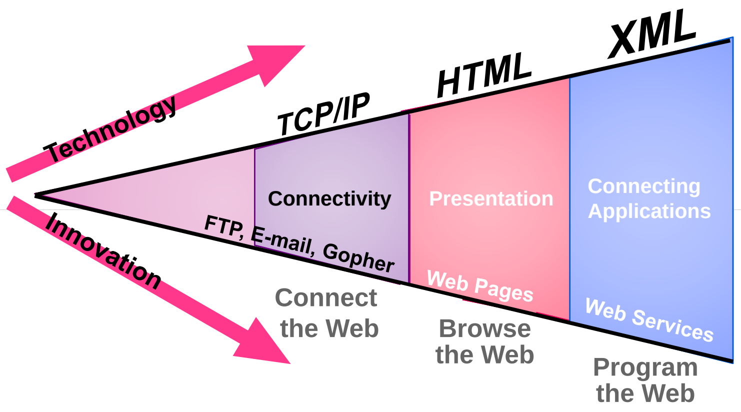
Criticisms of Web Services

- They are (can be) too complex
  - http://www.tbray.org/ongoing/When/200x/2004/09/21/WS-Research
- Biased towards large software vendors or integrators, rather than open source implementations.
- There are also concerns about performance, because of Web services' use of XML as a message format.
  - Web services are inherently slower than other standards because of the need to convert an XML blob of text into an object.

Underlying Technologies for Web Services

- A Web service *consumer* should be aware of
  - the existence of the Web Service,
  - the functionality provided by the Web Service,
  - the arguments to be provided by the Web Service to utilize the functionality, and
  - a standard way of communicating with the Web Service
- A Web service *provider* must provide all necessary information required to access and use the Web service including:
  - the methods exposed by a Web Service,
  - the parameters expected, and
  - the values returned by the methods
- The following technologies are used by Web service providers and consumers
  - XML: eXtensible Markup Language
  - SOAP: Simple Object Access Protocol
  - WSDL: Web Services Description Language
  - UDDI: Universal Description, Discovery, and Integration

XML is the Glue

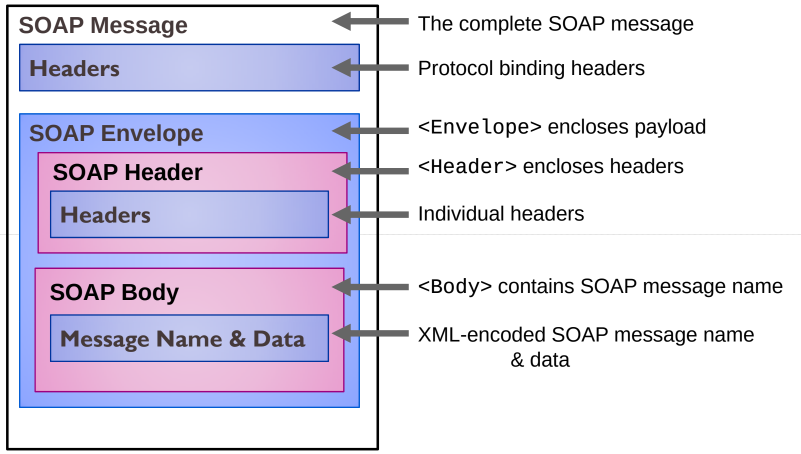


SOAP

- A new lightweight protocol based on XML and HTTP used to exchange data between a Web Service provider and consumer
- It enables cross-platform interoperability
  - Allows exchanging information in a decentralized, distributed, heterogeneous environment (W3C)
  - OS, object model, programming language neutral
  - Hardware independent
  - Protocol independent
- Works over existing Internet infrastructure
  - Guiding principle: "invent no new technology"
  - Builds on key Internet standards: SOAP ~ HTTP + XML
- The SOAP specification defines a number of things; the most important are:
  - the format of a SOAP message
  - how data should be encoded
  - how to send messages (method calls)
  - how to receive responses
- http://www.w3schools.com/soap/default.asp

SOAP Message Structure

A SOAP message is an XML document that consists of a mandatory SOAP envelope, an optional SOAP header, and a mandatory SOAP body.



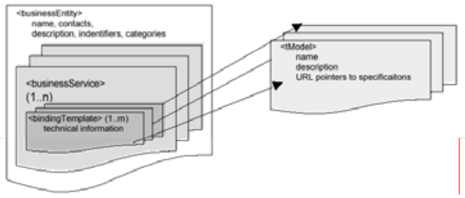
SOAP Message
Headers
SOAP Envelope
SOAP Header
Headers
SOAP Body
Message Name & Data
The complete SOAP message
Protocol binding headers
<Envelope> encloses payload
<Header> encloses headers
Individual headers
<Body> contains SOAP message name
XML-encoded SOAP message name & data

WSDL

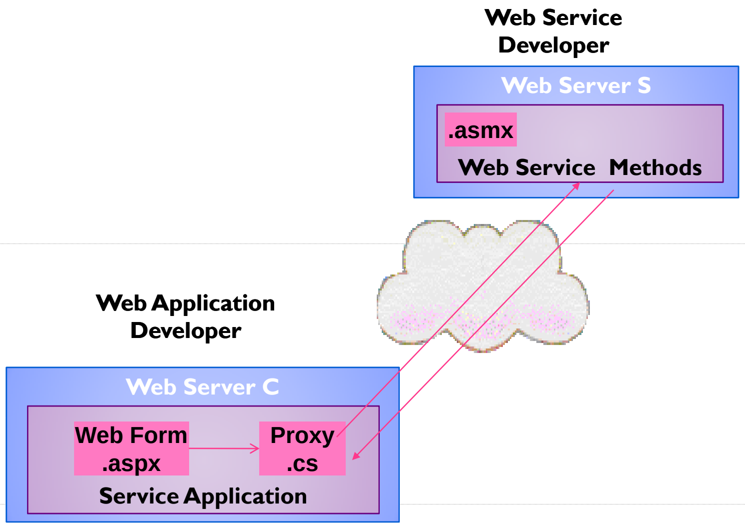
- What Web Services are there and where they are
- XML schema for describing Web Services
  - Service interface definition
    - Abstract semantics for Web Service
  - Service implementation definition
    - Concrete end points and network addresses where Web Service can be invoked
- Provide information like:
  - the methods exposed by a Web Service,
  - the parameters expected, and
  - the values returned by the methods
- Clear description between abstract and concrete messages

UDDI

- An open Industry Initiative to address discovery
  - A platform-independent, XML-based registration database for Web Services
  - A project started by Microsoft, IBM, and Ariba
- Web services can be published through the industry-wide UDDI initiative or Microsoft's DISCO
  - DISCO is a Microsoft publishing/discovery technology built into .NET
- UDDI registry consists of four main types of information
  - Business information consists of name, contact info, services, etc.
  - Service information provides business/technical descriptions of service
  - `bindingTemplate` defines how to connect to and communicate with service
  - `tModel` provides a technical fingerprint for interfaces/behavior (or anything else).
- More info:
  - Java Web Services http://oreilly.com/catalog/javawebserv/chapter/ch06.html
  - http://www.jisc.ac.uk/whatwedo/services/techwatch/reports/horizonscanning/hs0101.aspx



Developing a Web Service



Developing a Web Service

- Web Service
  - Implemented in ASP.NET
  - Similar to Web Forms, but
    - have a .asmx file extension
    - contains code, w/o UI
  - Can have a code-behind file
  - Store in an IIS-controlled directory
  - ASP.NET provides simple test harness
  - ASP.NET automatically generates WSDL
  - Can use .NET Framework classes and custom assemblies and classes

Tools

- Visual Studio.NET
  - Web Services can be written in any of the languages supported by the Visual Studio.NET
  - Create ASP.NET Web Service project
    - Start Visual Studio .NET >> Select New Project >> Select Visual C# Project as Project Type and Asp.Net Web Service as Template
      - A .NET Web service class that inherits from System.Web.Services.WebService
    - Add methods and properties to the .asmx file

Example 1

saved as an .asmx file

```
<%@ WebService Language="C#" Class="MathService" %>


using System;
using System.Web.Services;


public class MathService: WebServices {
   [WebMethod]
   public int Add(int num1, int num2) {
      return num1 + num2;
   }
}
```

main class must match the name declared in the Class attribute and must be public

Defining Web Methods

Example 2

saved as an .asmx file

```
<%@ WebService Language="VBScript" Class="TempConvert" %>
Imports System
Imports System.Web.Services

Public Class TempConvert :Inherits WebService

<WebMethod()> Public Function FahrenheitToCelsius (ByVal Fahrenheit
As String) As String
  dim fahr
  fahr=trim(replace(Fahrenheit,",","."))
  if fahr="" or IsNumeric(fahr)=false then return "Error"
  return ((((fahr) - 32) / 9) * 5)
end function

<WebMethod()>
Public Function CelsiusToFahrenheit (ByVal Celsius As String) As
String
   dim cel
   cel=trim(replace(Celsius,",","."))
   if cel="" or IsNumeric(cel)=false then return "Error"
   return ((((cel) * 9) / 5) + 32)
end function
end class
```

WebService Attribute

- The WebService attribute is
  - optional and does not affect the activity of the Web service class as to what is published and executed.
  - represented by an instance of the WebServiceAttribute class.
  - allows you to change three default settings for the Web service: *namespace*, *name*, and *description*.

```
[WebService(Namespace=“mathlib”,
    Name=“MathSev",
    Description=“Math Web Service")]
public class  MathService : WebService {
.
.
.

}
```

Defining Web Methods

- A Web service class is a regular .NET class, so it can have public as well as protected or private members
- Public methods marked with the WebMethod attribute
  - exposed over the Web
- The WebMethod attribute has several properties that you can use to tailor the behavior of a Web method

Property	Description
BufferResponse	Set to true (default) → indicates that ASP.NET should buffer the entire method's response before sending it to the client. Set to false only if you know that the method returns large amounts of data; the response is buffered but only in chunks of 16 KB.
CacheDuration	Specifies the length of time ASP.NET should cache the response of the method. This information is useful when you can foresee that the method will handle several calls in a short period of time. The duration is expressed in seconds and disabled by default. (The default is 0.) Caching recognizes distinct parameter values.
Description	Provides a description for the method which is then written to the WSDL for the service.
EnableSession	Set to false (default) → makes available to the method the Session object of the ASP.NET environment. Depending on how Session is configured, using this property might require cookie support on the client.
MessageName	Allows you to provide a publicly callable name for the method. Set the name of this property, and the resulting SOAP messages for the method target the name you set instead of the actual name. Use this property to give distinct names to overloaded methods in the event you use the same class as part of the middle tier and a Web service.

Using Codebehind File

```
<%@ WebService Language="c#"
Codebehind="MyWebService.cs"
Class="FirstWebService.MathService" %>
```

Consuming Web Services

- Web Services are <u>not</u> designed to be viewed in a browser
  - Instead, Web Services are consumed by a client application using protocols
- SOAP
  - SOAP is a protocol for accessing a web service
  - A simple XML-based protocol that allows applications to exchange information over HTTP
  - http://www.w3schools.com/soap/default.asp
- Locate the desired Web Service
  - UDDI
  - DISCO
- Get detailed description of Web Service
  - WSDL
    - XML-based language for describing Web services and how to access them
    - WSDL describes a web service, along with the message format and protocol details for the web service.
    - http://www.w3schools.com/wsdl/default.asp
- Create a proxy that represents the Web Service
  - Proxy has the same methods/arguments/return values as the Web Service
  - Application instantiates and uses the proxy as if it were a local object

Testing the Web Service

- Publish the .asmx file on a server with .NET support, and you will have your first working Web Service.
  - Web Services are URL addressable
    - HTTP request/response
- With ASP.NET, you do not have to write your own WSDL and SOAP documents.
- Can request WSDL via URL
- Can invoke via:
  - HTTP-GET
  - HTTP-POST

Trying It Out

- Request without method name or parameters
  - ASP.NET returns a page listing all methods

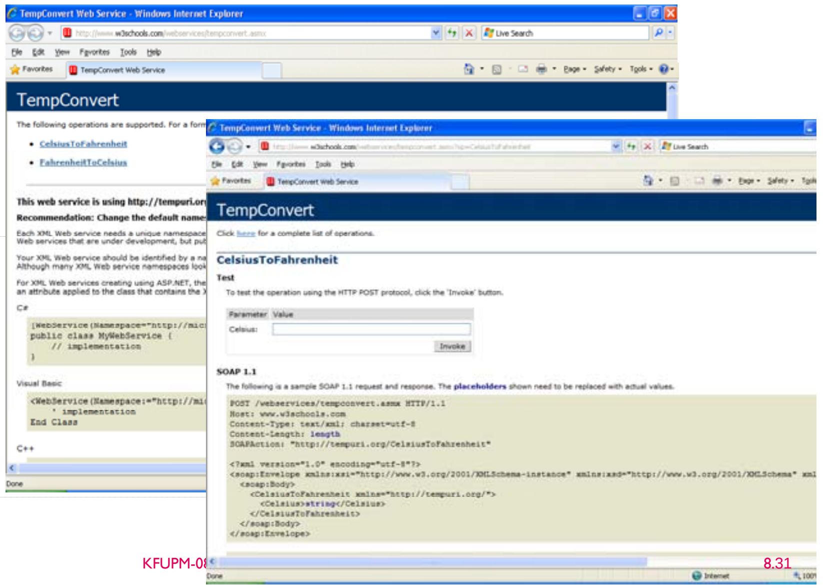
```
http://localhost/MathService.asmx
```

- Click one of the methods and you can test it out

```
http://localhost/MathService.asmx?op=Multiply
```

  - Specify parameters and Invoke
    - Only for primitive data types
  - Sample requests/responses

Testing the Web Service: Example


TempConvert Web Service - Windows Internet Explorer
http://www.w3schools.com/webservices/tempconvert.asmx
Live Search
File Edit View Favorites Tools Help
Favorites
TempConvert Web Service
Page Safety Tools
Done

TempConvert

The following operations are supported. For a form

- CelsiusToFahrenheit
- FahrenheitToCelsius

This web service is using http://tempuri.or

Recommendation: Change the default name

Each XML Web service needs a unique namespace
Web services that are under development, but pub

Your XML Web service should be identified by a na
Although many XML Web service namespaces look

For XML Web services creating using ASP.NET, the
an attribute applied to the class that contains the X

C#

```
[WebService(Namespace="http://mic
public class MyWebService {
    // implementation
}
```

Visual Basic

```
<WebService(Namespace:="http://mi
    ' implementation
End Class
```

C++

TempConvert Web Service - Windows Internet Explorer
http://www.w3schools.com/webservices/tempconvert.asmx?op=CelsiusToFahrenheit
Live Search
File Edit View Favorites Tools Help
Favorites
TempConvert Web Service
Page Safety Tools
Invoke
Done
Internet
100%

TempConvert

Click here for a complete list of operations.

CelsiusToFahrenheit

Test

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

Parameter	Value
Celsius:	

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 /webservices/tempconvert.asmx HTTP/1.1
Host: www.w3schools.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/CelsiusToFahrenheit"

<?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" xml
  <soap:Body>
    <CelsiusToFahrenheit xmlns="http://tempuri.org/">
      <Celsius>string</Celsius>
    </CelsiusToFahrenheit>
  </soap:Body>
</soap:Envelope>
```

Invoking via HTTP-GET

- HTTP-GET

```
http://localhost//MathService.asmx/Multiply?a=11&b=11
```

  - Result is an XML document

```
<?xml version="1.0" encoding="utf-8" ?>
<int xmlns="http://www.microsoft.com/MathService/">121</int>
```

Invoking via HTTP-POST

- HTTP-POST

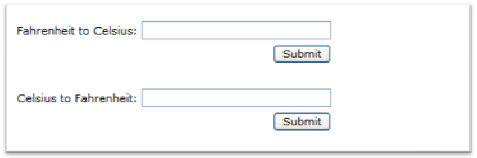
```
POST /MathService.asmx/Multiply HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

a=11&b=11
```

  - Result is an XML document

```
<?xml version="1.0" encoding="utf-8" ?>
<int xmlns="http://www.microsoft.com/MathService/">121</int>
```

Put the Web Service on Your Web Site

- Using a form and HTTP POST, you can put the web service on your site, like this:



Put the Web Service on Your Web Site (cont.)

- How to do it:

```
<form target="_blank" action='http://www.example.com
/webservices/tempconvert.asmx/FahrenheitToCelsius'
method="POST">
<table>
  <tr>
    <td>Fahrenheit to Celsius:</td>
    <td><input class="frmInput" type="text"
    size="30" name="Fahrenheit"></td>
  </tr>
  <tr>
    <td></td>
    <td align="right"> <input type="submit"
    value="Submit" class="button"></td>
  </tr>
</table>
</form>

<form target="_blank" action='http://www.example.com
/webservices/tempconvert.asmx/CelsiusToFahrenheit'
method="POST">
<table>
  <tr>
    <td>Celsius to Fahrenheit:</td>
    <td><input class="frmInput" type="text"
    size="30" name="Celsius"></td>
  </tr>
  <tr>
    <td></td>
    <td align="right"> <input type="submit"
    value="Submit" class="button"></td>
  </tr>
</table>
</form>
```

Creating a Proxy

- Use `wsdl.exe` to generate a proxy

```
>>wsdl http://localhost/MathService.asmx
```

  - Creates MathService.cs
  - Contains MathService class, derived from `SoapHttpClientProtocol` in the `System.Web.Services.Protocols` namespace
    - Or `HttpGetClientProtocol` or `HttpPostClientProtocol`
    - You can instantiate these classes dynamically
  - Proxy embeds URL to the Web Service in the constructor
  - After the .cs file is generated, you add it to compile line
    - Or you can create DLL library for the file

Using Visual Studio.NET

- Use Add Web Reference to search UDDI or to discover Web Services given a URL
- This builds a proxy, and you can start using the Web Service immediately
  - Visual Studio.NET essentially calls `disco.exe` and `wsdl.exe` for you

Q & A

?

Resources & Examples

- Building Web Solutions with ASP.NET and ADO.NET, by D. Esposito, 2002.
- Web Services with ASP.NET
  - http://msdn.microsoft.com/en-us/library/ms972326.aspx
- ASP.NET Web Services QuickStart Tutorial
  - http://quickstarts.asp.net/QuickStartv20/webservices/
- W3Schools Web Services Tutorial
  - http://www.w3schools.com/webservices/default.asp
- W3Schools SOAP Tutorial
  - http://www.w3schools.com/soap/default.asp
- Web Services Essentials
  - http://msdn.microsoft.com/library/default.asp?URL=/library/techart/webservicesessentials.htm
- SOAP
  - http://msdn.microsoft.com/soap
- SOAP Specification
  - http://www.w3.org/TR/SOAP/
- Don Box on SOAP
  - http://msdn.microsoft.com/msdnmag/issues/0300/soap/soap.asp
- Introduction to SOAP
  - http://www.w3.org/2000/xp/Group/Admin/minutes-oct1100/soap-xp-wg_files/frame.htm
- ASP.NET Official Website
  - http://www.asp.net/