

# Web Services

Jim Fawcett

CSE686 – Internet Programming

Summer 2004

# References

---

- ◆ Programming Microsoft .Net, Jeff Prosise, Microsoft Press, 2002
- ◆ Web Services, Mark Saposnek, Powerpoint presentation available from [www.gotdotnet.com](http://www.gotdotnet.com)

# Web Service Definition

- ◆ A web service is a set of methods exposed through a web interface.
  - Accessible through HTTP
  - Provides internet access to RPC-like calls that define the service
  - Web service messages are encoded in an XML dialect called Simple Object Access Protocol (SOAP)
- ◆ Service model assumes services are always available

# Benefits of the Web Service Model

- ◆ Web services use this special architecture because it:
  - Can be used from any platform.
  - Uses a standard, well-know channel.
  - Is routable and will pass through most firewalls.
  - Uses the same security mechanisms as any web site.

# Service Oriented Architecture

- ◆ Framework provides a set of fundamental operations via web services
  - May also provide local services using Windows services
- ◆ All applications based on that framework share the common services
  - Don't have to recreate the same functionality for each new application
- ◆ Can provide those same services to Partner businesses, suppliers, and customers
- ◆ Cloud-based services use web services for accessing platform facilities on cloud servers

# Comparing MicroSoft Web Service with ASP.Net

## ◆ ASP.NET

- Uses ASP pipeline
- `Applic.aspx`
- `Applic.aspx.cs`
- Uses Session, ...
- Visual Interface  
invoked from  
browser

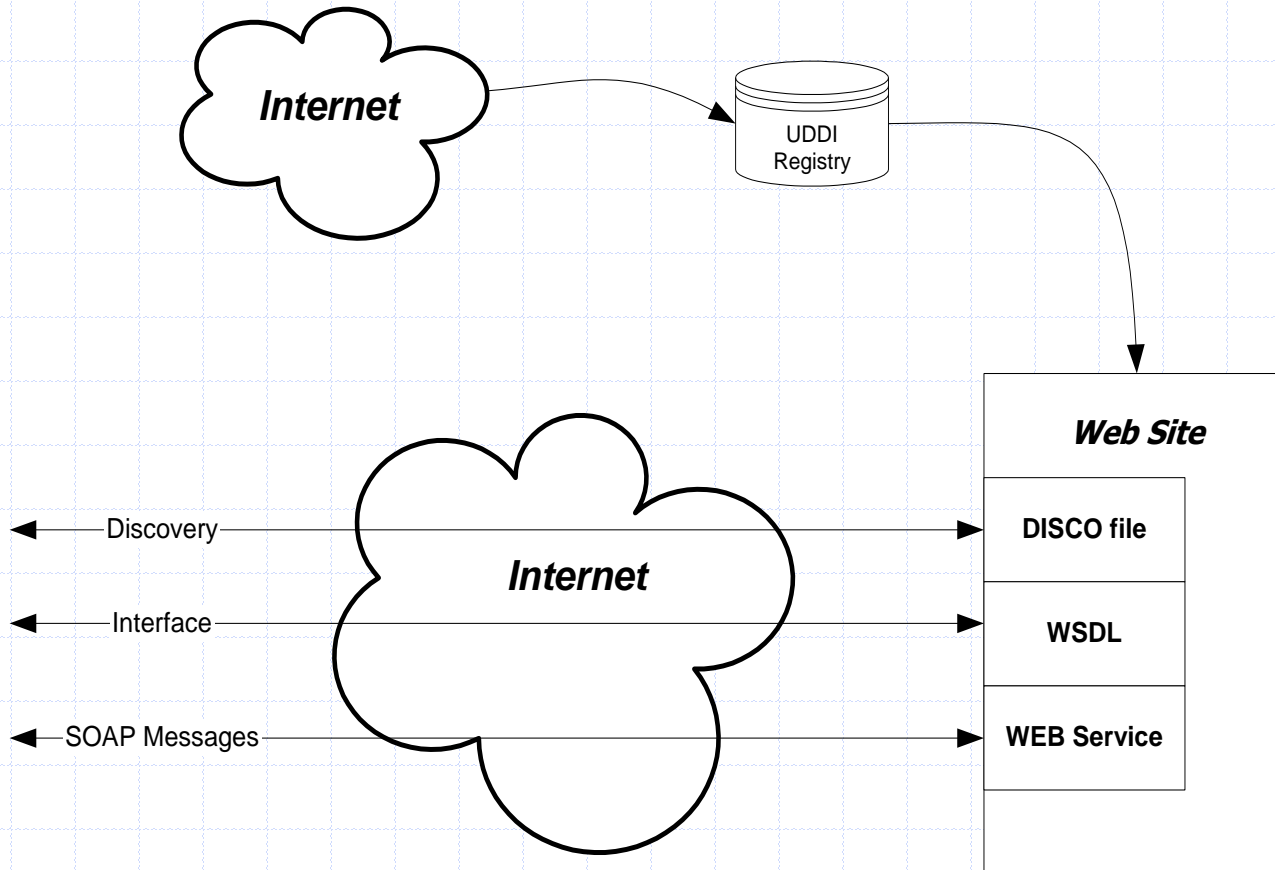
## ◆ Web Service

- Uses ASP pipeline
- `Applic.asmx`
- `Applic.asmx.cs`
- Uses Session, ...
- RPC Interface  
invoked by ASP or  
Winform app through  
proxy

# Web Service Protocols

- ◆ Web services are based on four protocols:
  - Extensible Markup Language (XML)
    - ◆ defines complex data structures
  - Web Service Description Language (WSDL)
    - ◆ Specifies the interface of the web service
  - Discovery Protocol (DISCO)
    - ◆ Pointer to all web services on a particular web site
  - Universal Description, Discovery, and Integration (UDDI)
    - ◆ Central repository of web service descriptions

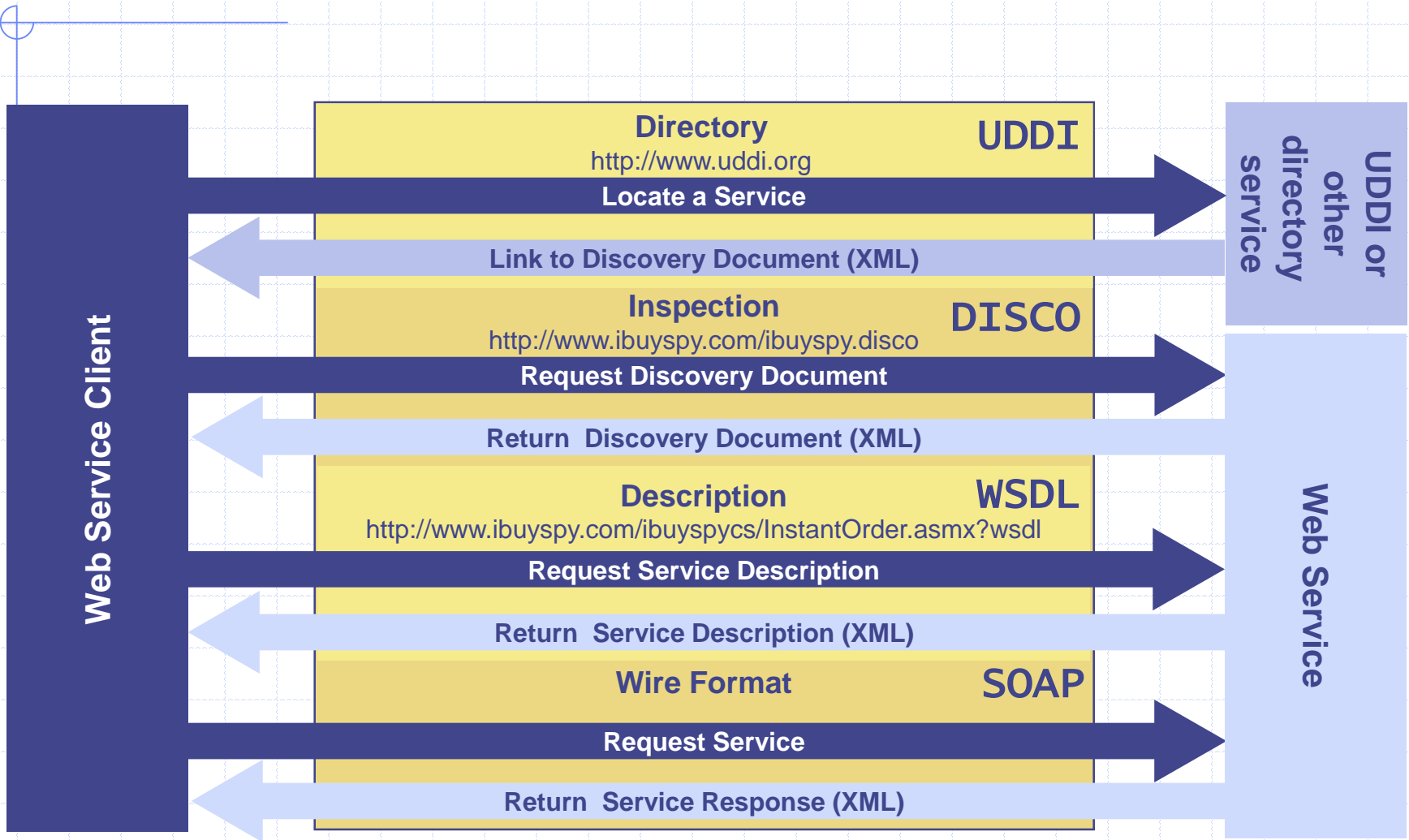
# Web Service Structure





# Underlying Technologies

## Web Services Stack

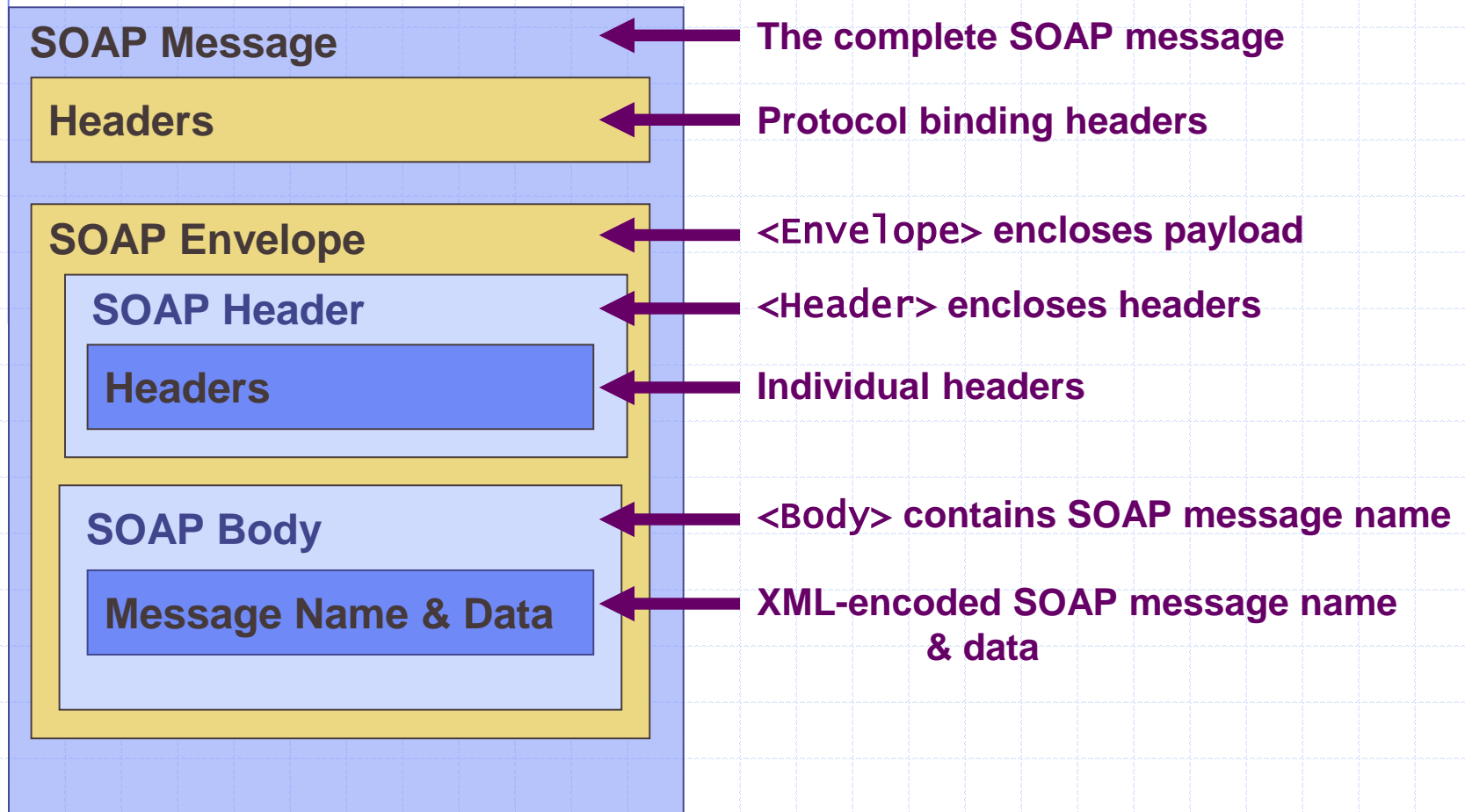


# SOAP Messages

- ◆ A SOAP Message can be one of three types:
  - Method call
    - ◆ Contains name of method and parameters
  - Method Response
    - ◆ Return values
  - Fault Message
    - ◆ SOAP fault message if service throws an exception
    - ◆ Will get standard HTTP message if transport fails.

# SOAP

## Message Structure



# WSDL

## WSDL Schema

### Interface

**<definitions>**

**<import>**

**<types>**

**<message>**

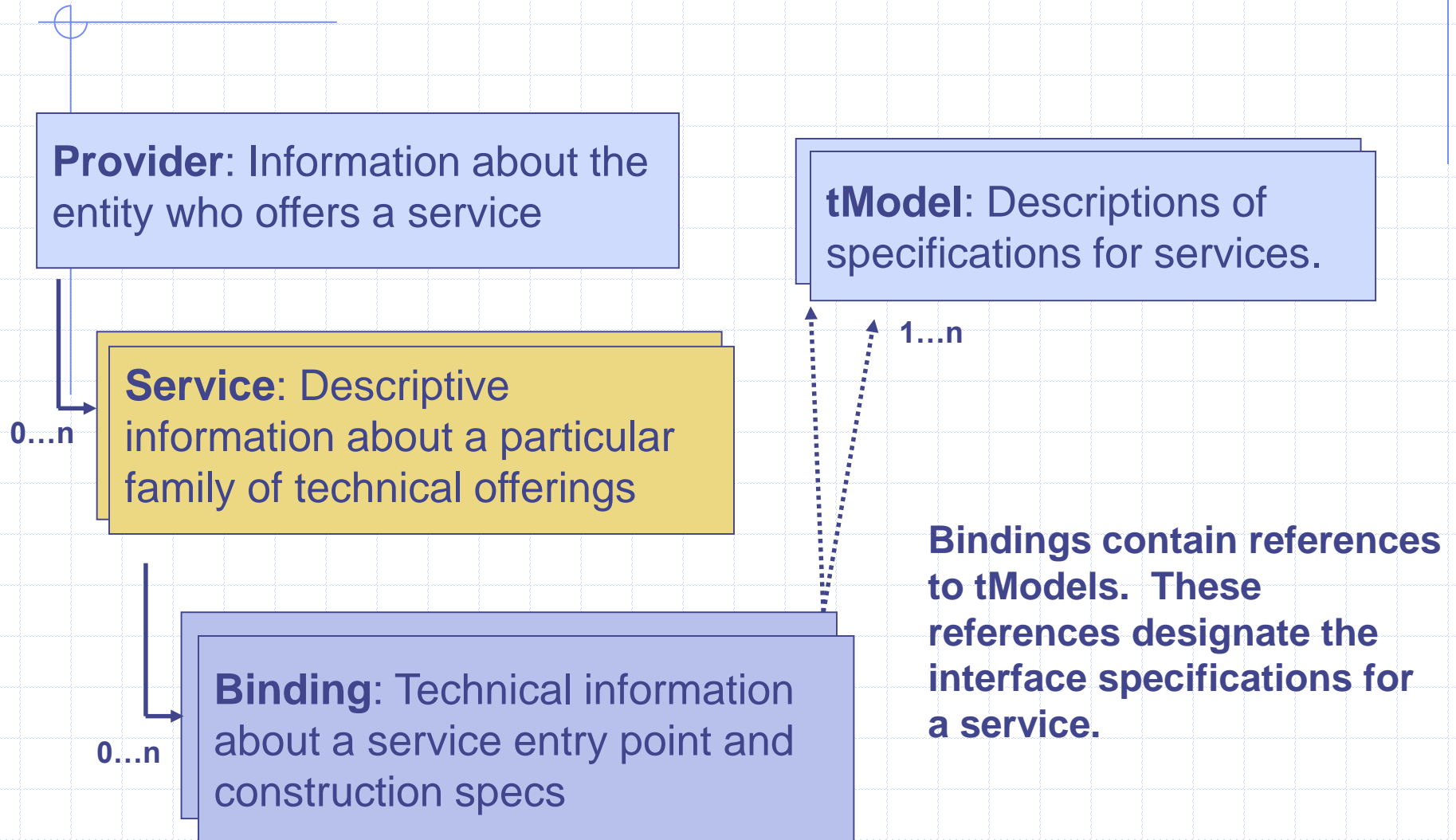
**<portType>**

**<binding>**

- **<definitions>** are root node of WSDL
- **<import>** allows other entities for inclusion
- **<types>** are data definitions - xsd
- **<message>** defines parameters of a Web Service function
- **<portType>** defines input and output operations
- **<binding>** specifies how each message is sent over the wire

# UDDI

## UDDI Information Model



# Structure of a Microsoft WebService

## ◆ MyService.asmx, MyService.asmx.cs

### ■ Page Directive:

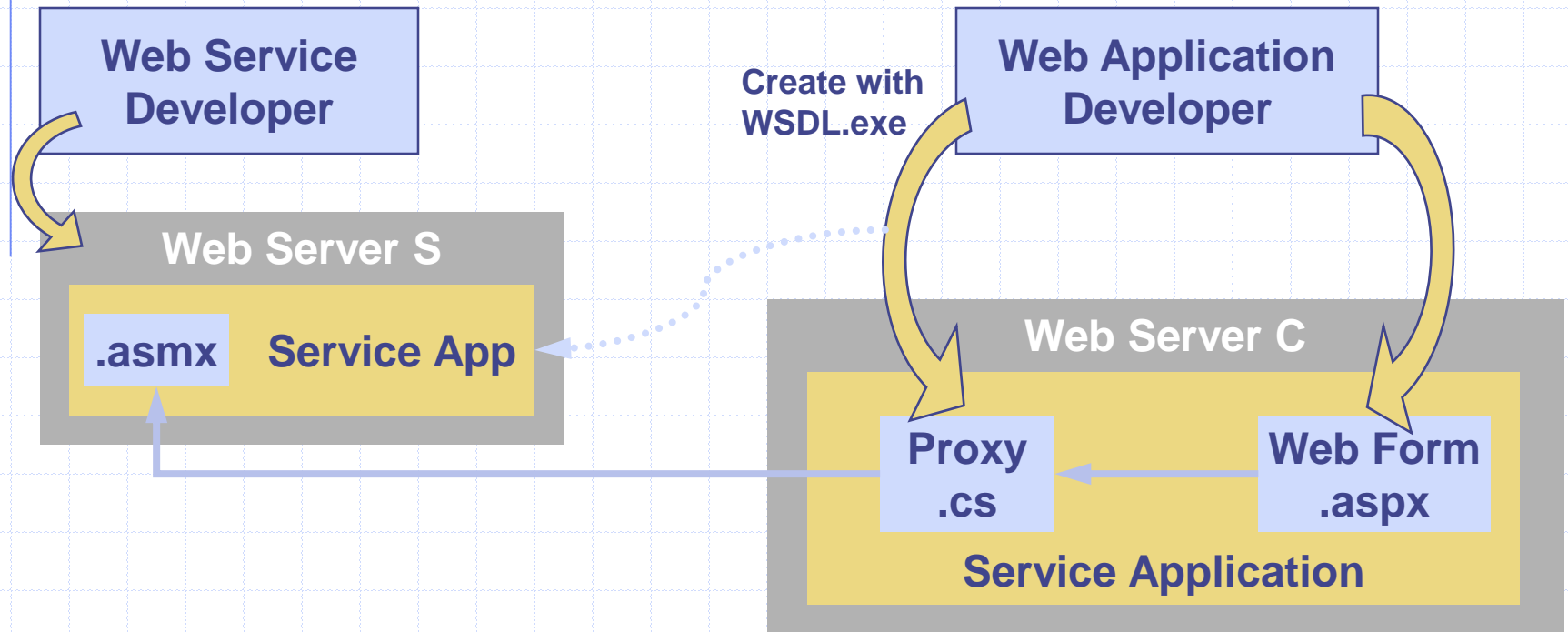
```
<%@ Webservice Language="C#"
Class="myService" %>
```

### ■ Class [derived from System.Web.Services.WebService]

### ■ Methods decorated with [WebMethod]

## ◆ Virtual Directory hosting this Application

# Consuming Web Services



# Structure of WebService Client

## ◆ myService Proxy code

- Generated using disco.exe and wsdl.exe  
(see CalcClient.cs code comments)

## ◆ myServiceClient code

- Ordinary ASP or Winform application
- `myService Proxy = new myService();`
- `Result = Proxy.myMethod(args);`



# DemoWebService Running

The screenshot shows the Microsoft Visual Studio IDE with the following components:

- Code Editor:** Displays the source code for `demo1.asmx.cs`. The code defines a web service namespace and a public method `demoMethod`.

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;

namespace demo1WebService
{
    [WebService(Namespace="http://Syracuse_Software_Technology_Company/")]
    public class demo1 : System.Web.Services.WebService
    {
        public demo1()
        {
            //CODEGEN: This call is required by the ASP.NET Web Services Designer
            InitializeComponent();
        }

        [WebMethod]
        public string demoMethod(string fromClient)
        {
            return ("Hello Internet Programming. You sent \"\" + fromClient + "\"");
        }
    }
}
```
- Web service description:** A yellow callout bubble points to the `[WebService(Namespace="http://Syracuse_Software_Technology_Company/")]` attribute in the code.
- Public web method:** A yellow callout bubble points to the `demoMethod` method in the code.
- Console Window:** Shows the output of the web service method.

```
Testing demo1WebService
=====
Hello Internet Programming. You sent ""
Press any key to continue
```
- Output Window:** Shows the build status.

```
Build
----- Done -----
Build: 2 succeeded, 0 failed, 0 skipped
```
- Solution Explorer:** Shows the project structure for `demo1WebService`, including `demo1.asmx`, `demo1.cs`, and `demo1.wsdl`.

# Client of DemoWebService

```
using System;

namespace testDemo1
{
    class demo1
    {
        [STAThread]
        static void Main(string[] args)
        {
            Console.WriteLine("\n Testing demo1WebService ");
            Console.WriteLine("\n =====\n");

            demo1WebService.demo1 proxy = new demo1WebService.demo1();
            string result = proxy.demoMethod("string from client");
            Console.WriteLine("\n " + result);
            Console.WriteLine("\n\n");
        }
    }
}
```

Instantiating web service proxy

Adding web reference creates proxy

Output

Debug

'DefaultDomain': Loaded 'c:\winnt\microsoft.net\framework\v1.0.3705\mscorlib.dll', No symbols loaded.  
'testDemo1': Loaded 'C:\inetpub\wwwroot\Lecture8\testDemo1\bin\Debug\testDemo1.exe', Symbols loaded.  
'testDemo1.exe': Loaded 'c:\winnt\assembly\gac\system.web.services\1.0.3300.0\_b03f5f7f11d50a3a\system.web.services.dll', No symbols loaded.  
'testDemo1.exe': Loaded 'c:\winnt\assembly\gac\system\1.0.3300.0\_b77a5c561934e089\system.dll', No symbols loaded.  
'testDemo1.exe': Loaded 'c:\winnt\assembly\gac\system.xml\1.0.3300.0\_b77a5c561934e089\system.xml.dll', No symbols loaded.  
'testDemo1.exe': Loaded 'c:\winnt\assembly\gac\system.web\1.0.3300.0\_b03f5f7f11d50a3a\system.web.dll', No symbols loaded.  
'testDemo1.exe': Loaded 'ljzr-mm', No symbols loaded.

# SOAP Request and Response

demo1 Web Service - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address <http://localhost/Lecture8/demo1WebService/demo1.asmx?op=demoMethod>

## demoMethod

### Test

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

Parameter	Value
fromClient:	<input type="text" value="a message from client"/>

Invoke

### SOAP

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

```
POST /Lecture8/demo1WebService/demo1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://Syracuse_Software_Technology_Company/demoMethod"

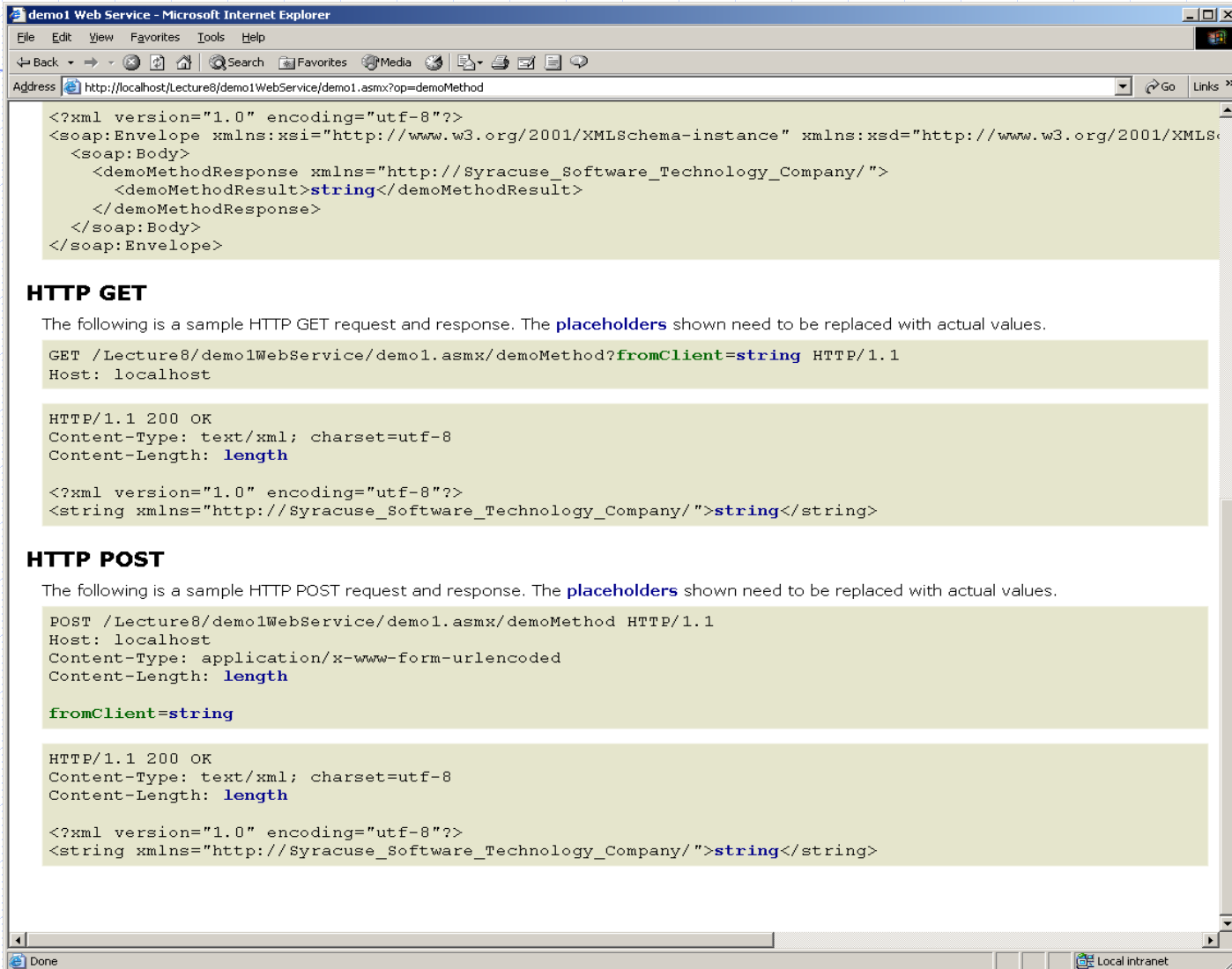
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <demoMethod xmlns="http://Syracuse_Software_Technology_Company/">
      <FromClient>string</fromClient>
    </demoMethod>
  </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-instance">
  <soap:Body>
    <demoMethodResponse xmlns="http://Syracuse_Software_Technology_Company/">
      <demoMethodResult>string</demoMethodResult>
    </demoMethodResponse>
  </soap:Body>
</soap:Envelope>
```

Done Local intranet

# HTTP GET and POST exchanges



The screenshot shows a Microsoft Internet Explorer window titled "demo1 Web Service - Microsoft Internet Explorer". The address bar contains the URL "http://localhost/Lecture8/demo1WebService/demo1.asmx?op=demoMethod". The main content area displays a SOAP XML response:

```
<?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"
  <soap:Body>
    <demoMethodResponse xmlns="http://Syracuse_Software_Technology_Company/">
      <demoMethodResult>string</demoMethodResult>
    </demoMethodResponse>
  </soap:Body>
</soap:Envelope>
```

## HTTP GET

The following is a sample HTTP GET request and response. The **placeholders** shown need to be replaced with actual values.

```
GET /Lecture8/demo1WebService/demo1.asmx/demoMethod?fromClient=string HTTP/1.1
Host: localhost
```

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

```
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://Syracuse_Software_Technology_Company/">string</string>
```

## HTTP POST

The following is a sample HTTP POST request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /Lecture8/demo1WebService/demo1.asmx/demoMethod HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

fromClient=string
```

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

```
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://Syracuse_Software_Technology_Company/">string</string>
```

# demo1.wSDL

The screenshot displays the Microsoft Visual Studio .NET interface. The main window shows the WSDL file 'demo1.wsd' in design view. The WSDL content is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:s="http://schemas.xmlsoap.org/wsdl/XMLSchema-instance/" targetNamespace="http://schemas.xmlsoap.org/wsdl/http/">
  <types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://Syracuse_Software_Technology_Company/">
      <s:element name="demoMethod">
        <s:complexType base="string">
          <s:sequence base="string">
            <s:element minOccurs="0" maxOccurs="1" name="fromClient" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="demoMethodResponse">
        <s:complexType base="string">
          <s:sequence base="string">
            <s:element minOccurs="0" maxOccurs="1" name="demoMethodResult" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="string" nillable="true" type="s:string" />
    </s:schema>
  </types>
  <message name="demoMethodSoapIn">
    <part name="parameters" element="s0:demoMethod" />
  </message>
  <message name="demoMethodSoapOut">
    <part name="parameters" element="s0:demoMethodResponse" />
  </message>
  <message name="demoMethodHttpGetIn">
    <part name="fromClient" type="s:string" />
  </message>
  <message name="demoMethodHttpGetOut">
    <part name="Body" element="s0:string" />
  </message>
  <message name="demoMethodHttpPostIn">
    <part name="fromClient" type="s:string" />
  </message>
  <message name="demoMethodHttpPostOut">
  </message>

```

The Solution Explorer on the right shows the project structure for 'testDemo1'. The 'demo1.wsd' file is highlighted under the 'References' folder.

The Output window at the bottom shows the following debug messages:

```
'DefaultDomain': Loaded 'c:\winnt\microsoft.net\framework\v1.0.3705\mscorlib.dll', No symbols loaded.
'testDemo1.exe': Loaded 'C:\inetpub\wwwroot\lecture8\testDemo1\bin\Debug\testDemo1.exe', Symbols loaded.
'testDemo1.exe': Loaded 'c:\winnt\assembly\gac\system.web.services\1.0.3300.0_b03f5f7f11d50a3a\system.web.services.dll', No symbols loaded.
'testDemo1.exe': Loaded 'c:\winnt\assembly\gac\system.xml\1.0.3300.0_b77a5c561934e089\system.xml.dll', No symbols loaded.
'testDemo1.exe': Loaded 'c:\winnt\assembly\gac\system.xml\1.0.3300.0_b77a5c561934e089\system.xml.dll', No symbols loaded.
'testDemo1.exe': Loaded 'c:\winnt\assembly\gac\system.web\1.0.3300.0_b03f5f7f11d50a3a\system.web.dll', No symbols loaded.
'testDemo1.exe': Loaded 'ljzr-mmu', No symbols loaded.
```

# demo1.disco

The screenshot shows the Microsoft Visual Studio IDE with the following content:

**XML Content:**

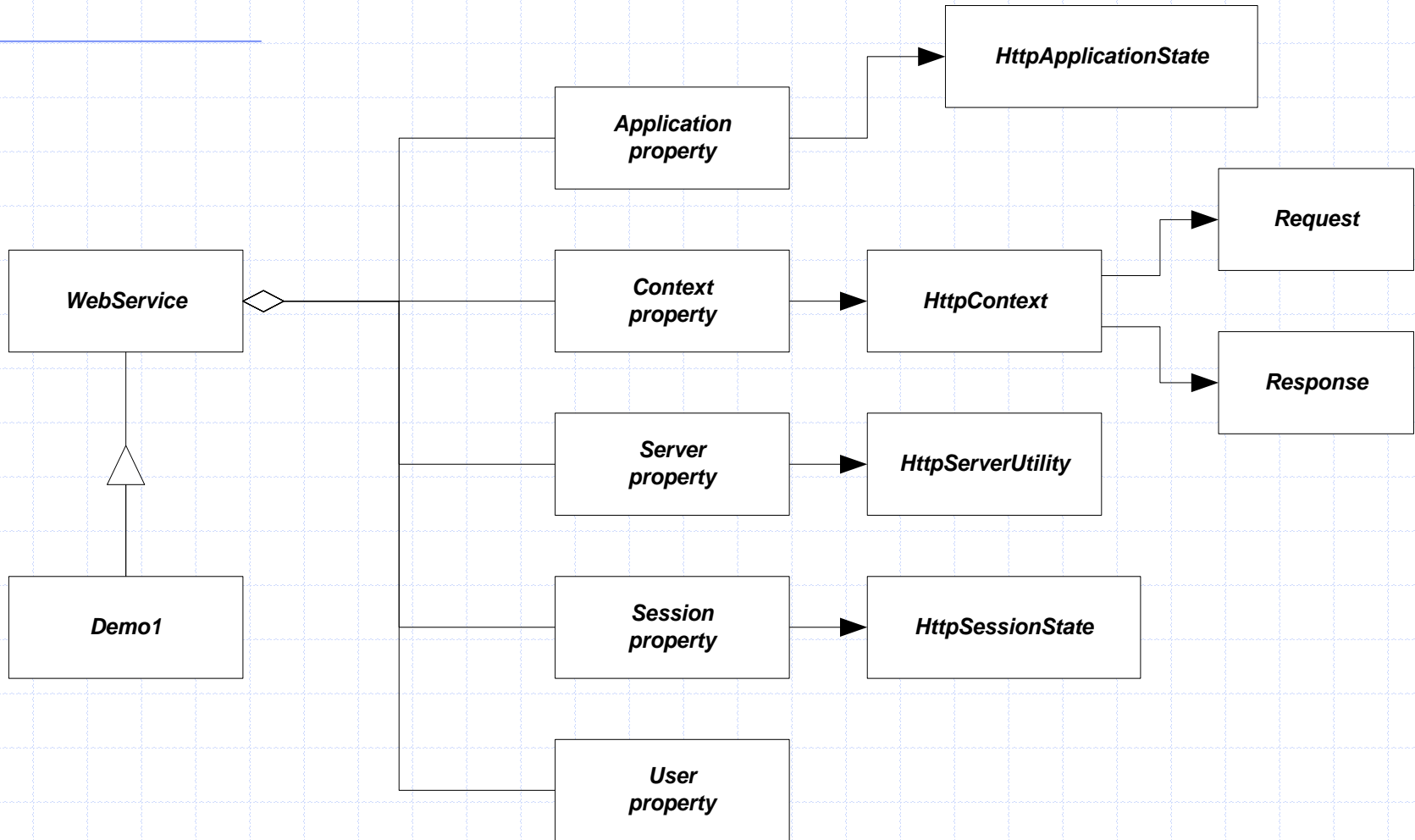
```
<?xml version="1.0" encoding="utf-8"?>
<discovery xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.microsoft.com/2003/10/Serialization/discov"
<contractRef ref="http://localhost/Lecture8/demoWebService/demo1.asmx?wsdl" docRef="http://localhost/Lecture8/demoWebServ
<soap address="http://localhost/Lecture8/demoWebService/demo1.asmx" xmlns:q1="http://Syracuse_Software_Technology_Company/
</discovery>
```

**Index Results for WebService class, all members - 1 topics found**

Title	Location
WebService Members	.NET Framework Class Library

**Solution Explorer:** Shows the project structure for 'testDemo1' and 'demo1WebService'. The 'demo1WebService' folder contains 'References', 'AssemblyInfo.cs', 'demo1.asmx', 'demo1WebService.vsd', 'Global.asax', and 'Web.config'. The 'testDemo1' folder contains 'References', 'Web References', 'demo1WebService', 'demo1.wSDL', and 'Reference.map'. The 'demo1WebService' folder is expanded to show 'demo1.disco' and 'demo1.wSDL'.

# Web Service Application Structure



# WebService Properties

- ◆ **HttpApplicationState**
  - Share state among all users of an application.
- ◆ **HttpSessionState**
  - Share state from page to page for one user.
- ◆ **HttpContext**
  - Provides access to the server Request and Response objects.
- ◆ **HttpServerUtility**
  - Provides CreateObject, Execute, and MapPath methods.
- ◆ **User**
  - Supports authentication of user.



# WebMethods

- ◆ WebMethod methods can pass any of the C# and CLR types
- ◆ User defined objects can also be passed if they are serializable:
  - .Net XML serializer will not serialize non-public members
  - User defined types can only be passed with SOAP. GET and POST won't work.
  - The WSDL contract contains a schema description of any user defined objects passed by a WebMethod

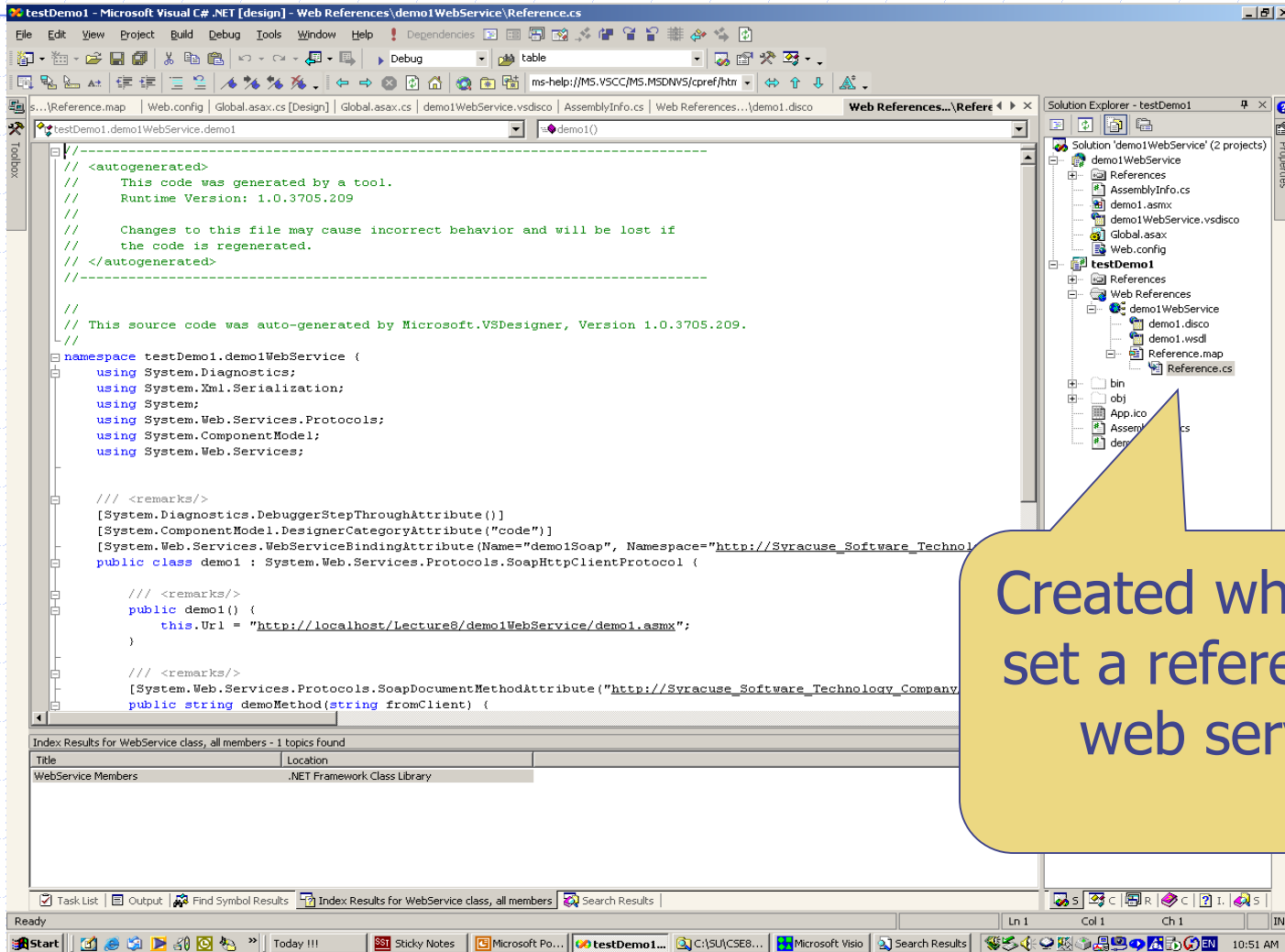
# Web Service Clients

- ◆ Web Service Clients use Web Service proxies to communicate with the remote service:

```
// create proxy instance
demo1WebService.demo1 proxy = new demo1WebService.demo1();

// use proxy
string result = proxy.demoMethod("string from client");
```

# AutoGenerated Proxy



Created when you set a reference to web service

# Web Services versus Remoting

## ◆ Web Services:

- Can be used by any platform that understands XML, SOAP, and WSDL.
  - ◆ Metadata (types) provided by WSDL
- Hosted by IIS and inherits ASP's security model.
- Uses HTTP protocol so accessible by web pages and can pass through most firewalls.
- Can only pass a limited set of user-defined objects:
  - ◆ Can't serialize an object graph or all .Net containers.

# Web Services versus Remoting

## ◆ Remoting:

- Requires .Net platform on client as well as server.
- Requires custom security (notoriously hard to get right).
- Metadata provided by assembly.
  - ◆ Can pass any .Net type, including object graphs and all .Net containers.
  - ◆ Rich, but none portable types.

# An Example

## ◆ FileXferService

### ■ Public Interface:

- ◆ `string[] RequestFileNames();`
- ◆ `Byte[] RequestFile(string FileName);`

# Browser View

WSfileXfer Web Service - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address <http://lusitania/lecture14/fileXfer/fileXferService.aspx> Go Links Norton AntiVirus

## WSfileXfer

The following operations are supported. For a formal definition, please review the [Service Description](#).

- [RequestFileNames](#)
- [RequestFile](#)

---

**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 Web. <http://tempuri.org/> is available for XML Web services that are under development, but 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 Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services created using ASP.NET, the default namespace can be changed using the `WebService` attribute's `Namespace` property. The `WebService` attribute is an attribute applied to the class that contains the 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.NET

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

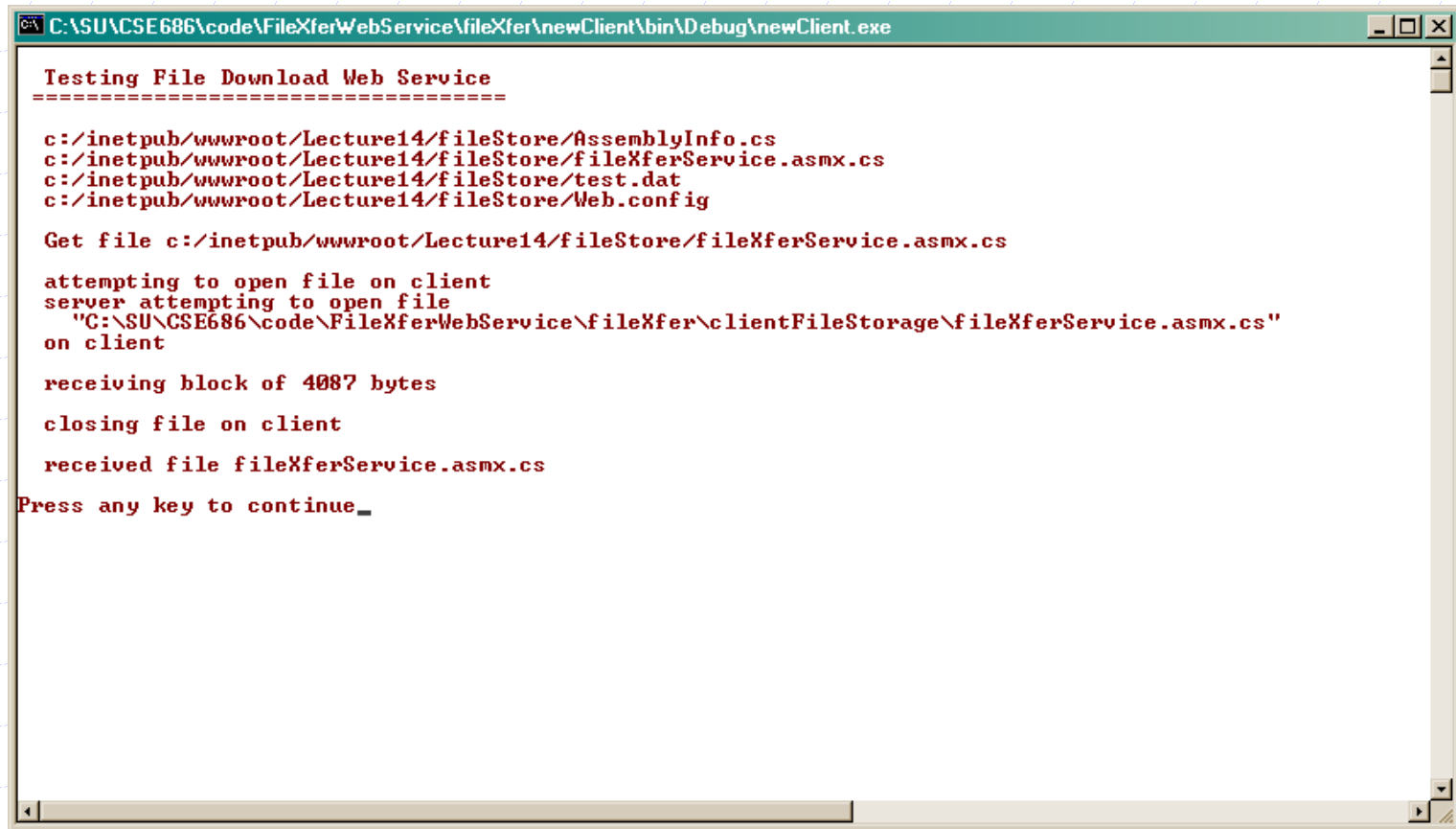
For more details on XML namespaces, see the W3C recommendation on [Namespaces in XML](#).

For more details on WSDL, see the [WSDL Specification](#).

For more details on URIs, see [RFC 2396](#).

Done Local intranet

# Client Application View



```
C:\SU\CSE686\code\FileXferWebService\fileXfer\newClient\bin\Debug\newClient.exe

Testing File Download Web Service
=====

c:/inetpub/wwwroot/Lecture14/fileStore/AssemblyInfo.cs
c:/inetpub/wwwroot/Lecture14/fileStore/fileXferService.asmx.cs
c:/inetpub/wwwroot/Lecture14/fileStore/test.dat
c:/inetpub/wwwroot/Lecture14/fileStore/Web.config

Get file c:/inetpub/wwwroot/Lecture14/fileStore/fileXferService.asmx.cs

attempting to open file on client
server attempting to open file
"C:\SU\CSE686\code\FileXferWebService\fileXfer\clientFileStorage\fileXferService.asmx.cs"
on client

receiving block of 4087 bytes

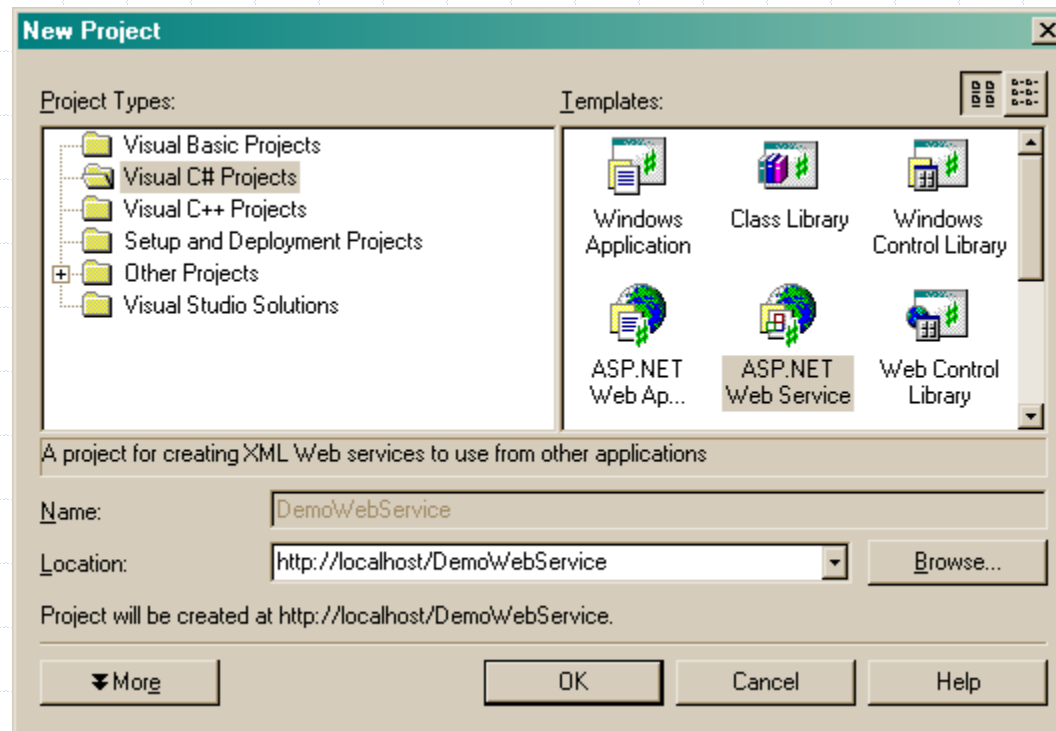
closing file on client

received file fileXferService.asmx.cs

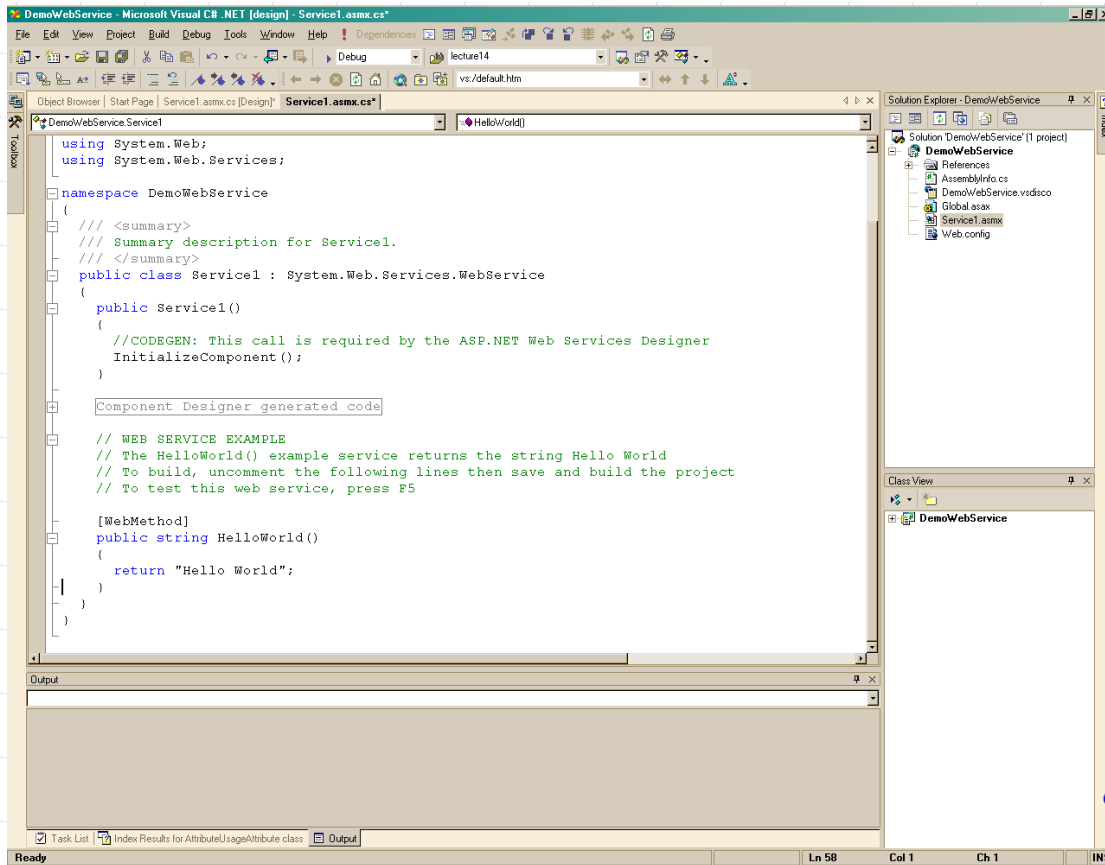
Press any key to continue_
```



# Creating a Web Service Project



# Resulting "Generic" Web Service



```
using System.Web;
using System.Web.Services;

namespace DemoWebService
{
    /// <summary>
    /// Summary description for Service1.
    /// </summary>
    public class Service1 : System.Web.Services.WebService
    {
        public Service1()
        {
            //CODEGEN: This call is required by the ASP.NET Web Services Designer
            InitializeComponent();
        }

        Component Designer generated code

        // WEB SERVICE EXAMPLE
        // The HelloWorld() example service returns the string Hello World
        // To build, uncomment the following lines then save and build the project
        // To test this web service, press F5

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

# Resulting "Generic" Test

Service1 Web Service - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://localhost/DemoWebService/Service1.asmx

## Service1

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 Web. <http://tempuri.org/> is available for XML Web services that are under development, but 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 Internet domain name as part of the namespace. Although many XML Web service namespaces 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 Namespace property. The WebService attribute is an attribute applied to the class that contains the 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.NET

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

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](#).

Done Local intranet

# Sample Soap Request

Service1 Web Service - Microsoft Internet Explorer

Address <http://localhost/DemoWebService/Service1.aspx?top=HelloWorld>

## Service1

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

### HelloWorld

Test

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

SOAP

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

```
POST /DemoWebService/Service1.aspx 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" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <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" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <HelloWorldResponse xmlns="http://tempuri.org/">
      <HelloWorldResult>string</HelloWorldResult>
    </HelloWorldResponse>
  </soap:Body>
</soap:Envelope>
```

HTTP GET

The following is a sample HTTP GET request and response. The **placeholders** shown need to be replaced with actual values.

```
GET /DemoWebService/Service1.aspx/HelloWorld? HTTP/1.1
Host: localhost
```

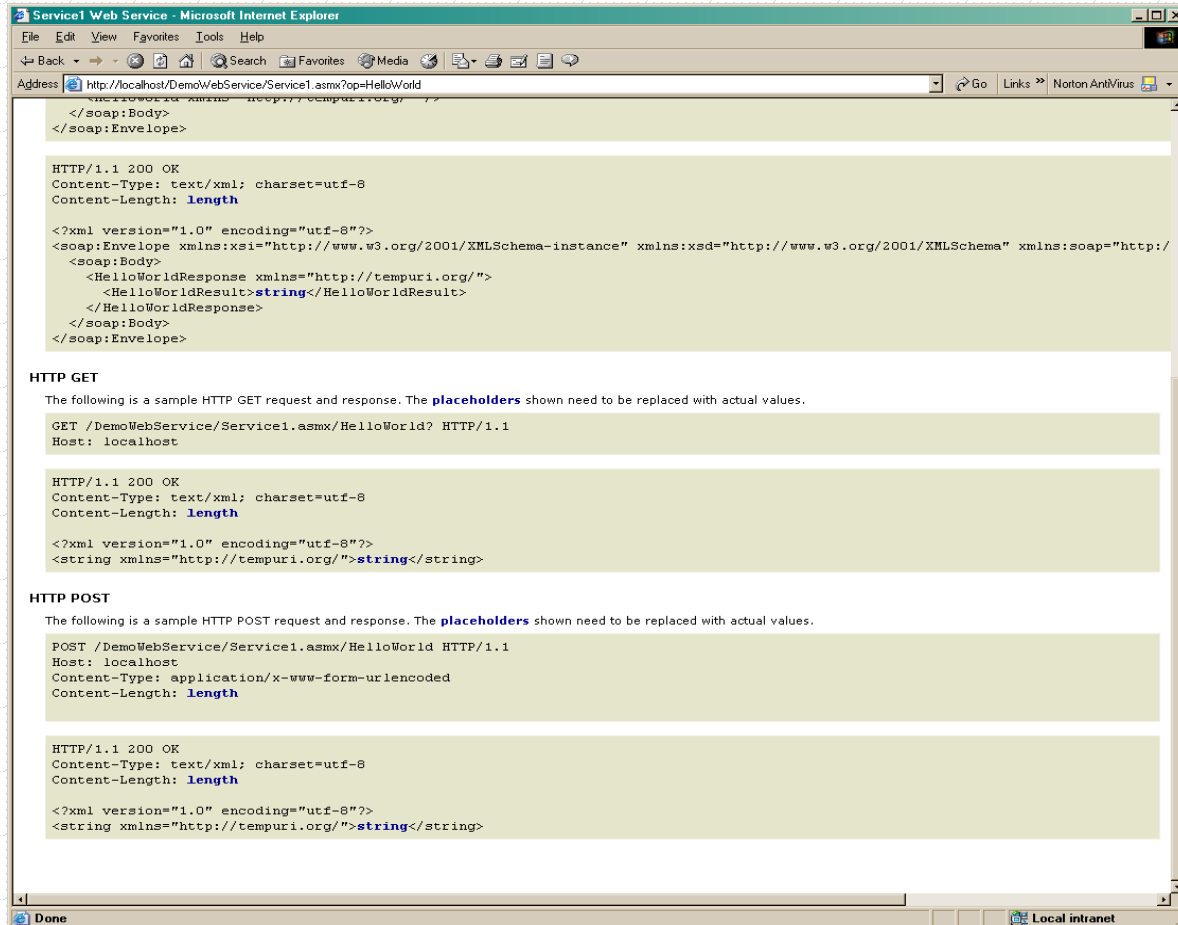
http://localhost/DemoWebService/Service1.aspx/HelloWorld? - Microsoft Inter...

Address <http://localhost/DemoWebService/Service1.aspx/HelloWorld?>

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

Done Local intranet

# Sample "Generic" HTTP GET and POST



The screenshot shows a Microsoft Internet Explorer window titled "Service1 Web Service - Microsoft Internet Explorer". The address bar contains the URL "http://localhost/DemoWebService/Service1.asmx?op=HelloWorld". The main content area displays the raw SOAP message. The response is an HTTP 200 OK with a Content-Type of "text/xml; charset=utf-8" and a Content-Length of "length". The XML body contains a `HelloWorldResponse` element with a `HelloWorldResult` child element containing the string "HelloWorld".

```
</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" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <HelloWorldResponse xmlns="http://tempuri.org/">
      <HelloWorldResult>string</HelloWorldResult>
    </HelloWorldResponse>
  </soap:Body>
</soap:Envelope>
```

**HTTP GET**

The following is a sample HTTP GET request and response. The **placeholders** shown need to be replaced with actual values.

```
GET /DemoWebService/Service1.asmx/HelloWorld? HTTP/1.1
Host: localhost

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

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

**HTTP POST**

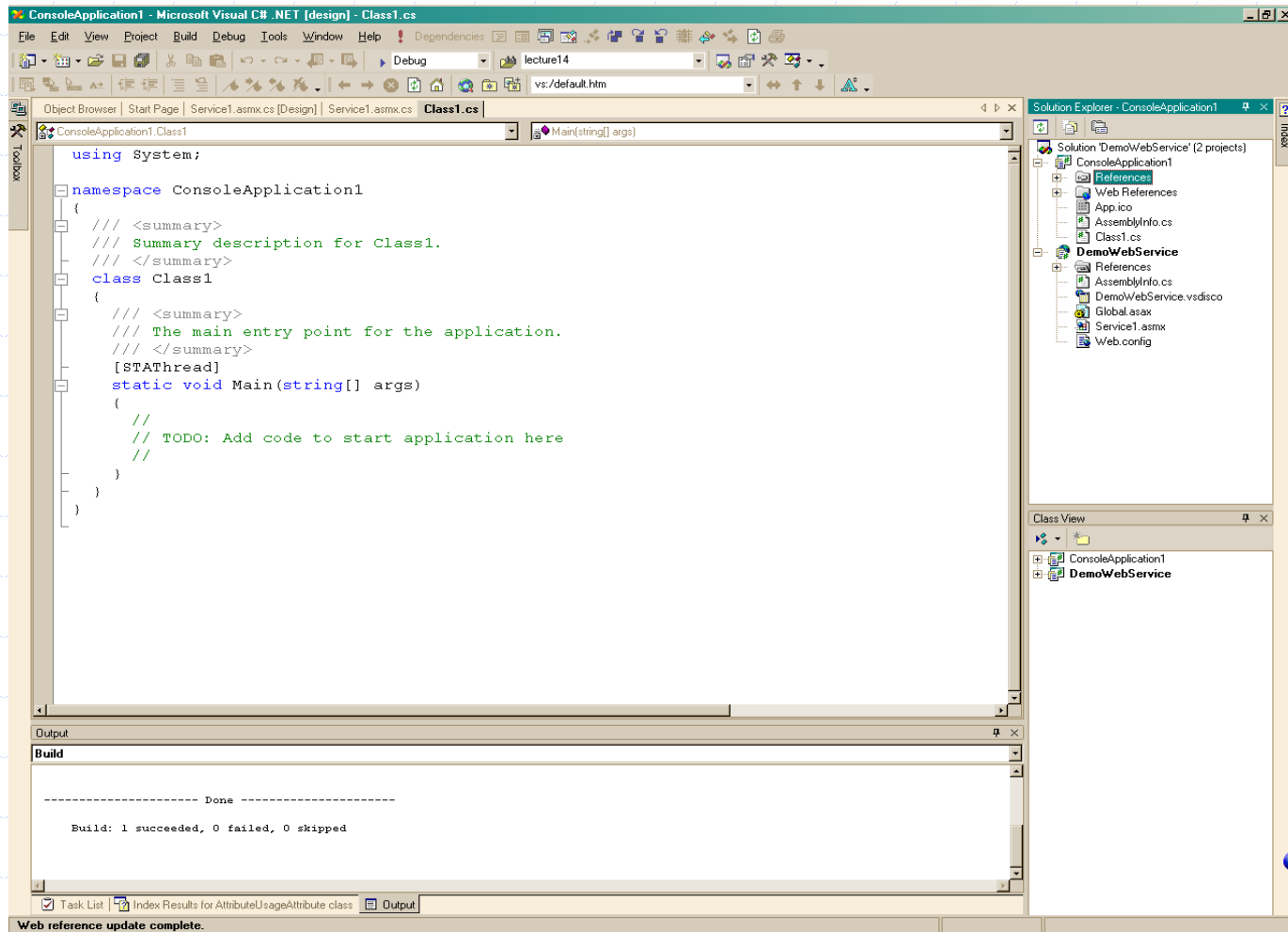
The following is a sample HTTP POST request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /DemoWebService/Service1.asmx/HelloWorld HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

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

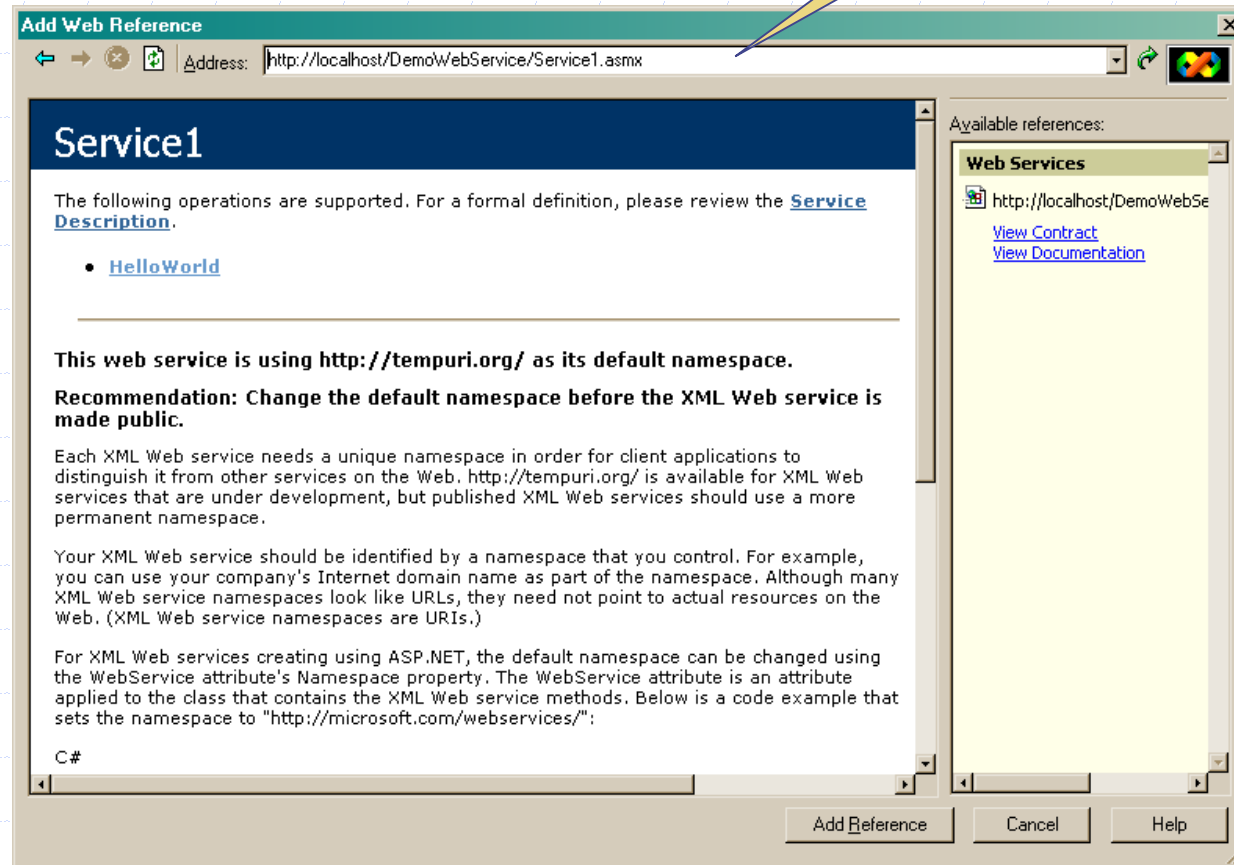
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://tempuri.org/">string</string>
```

# Create Console Client



# Adding a Web Reference

You have to locate the folder and asmx file, using explorer, then type in path here.



# Client Accessing Web Service

Note:  
You also need a  
reference to  
project so client  
knows about web  
service type.

```
using System;

namespace ConsoleApplication1
{
    class Class1
    {
        [STAThread]
        static void Main(string[] args)
        {
            DemoWebService.Service1 proxy = new DemoWebService.Service1();
            Console.WriteLine("\n Result is: {0}\n\n", proxy.HelloWorld());
        }
    }
}
```

Output

Build

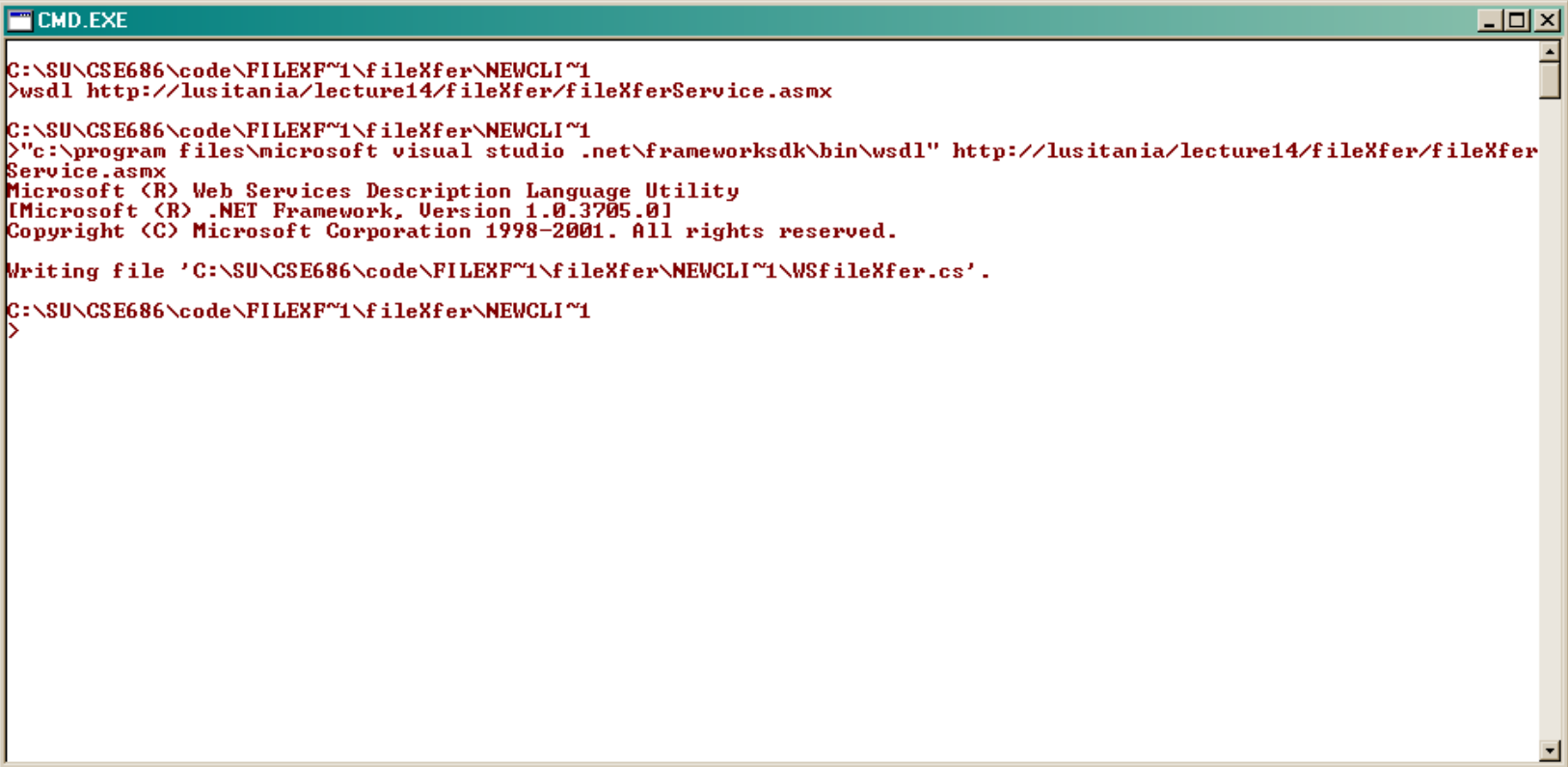
----- Done -----

Build: 2 succeeded, 0 failed, 0 skipped

Build succeeded



# Create Proxy Source Code with WDSL.exe



```
CMD.EXE
C:\SU\CSE686\code\FILEXF~1\fileXfer\NEWCLI~1
>wsdl http://lusitania/lecture14/fileXfer/fileXferService.asmx
C:\SU\CSE686\code\FILEXF~1\fileXfer\NEWCLI~1
>"c:\program files\microsoft visual studio .net\frameworksdk\bin\wsdl" http://lusitania/lecture14/fileXfer/fileXfer
Service.asmx
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 1.0.3705.0]
Copyright (C) Microsoft Corporation 1998-2001. All rights reserved.

Writing file 'C:\SU\CSE686\code\FILEXF~1\fileXfer\NEWCLI~1\WSfileXfer.cs'.

C:\SU\CSE686\code\FILEXF~1\fileXfer\NEWCLI~1
>
```

# Web Services



The End