



C# Programming in Depth

Prof. Dr. Bertrand Meyer

March 2007 - May 2007

Lecture 12: Web Service

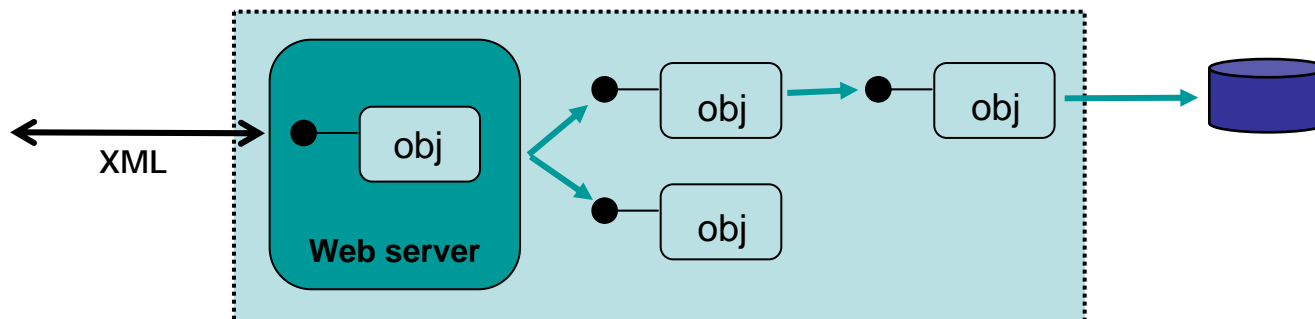
Lisa (Ling) Liu

Web service?

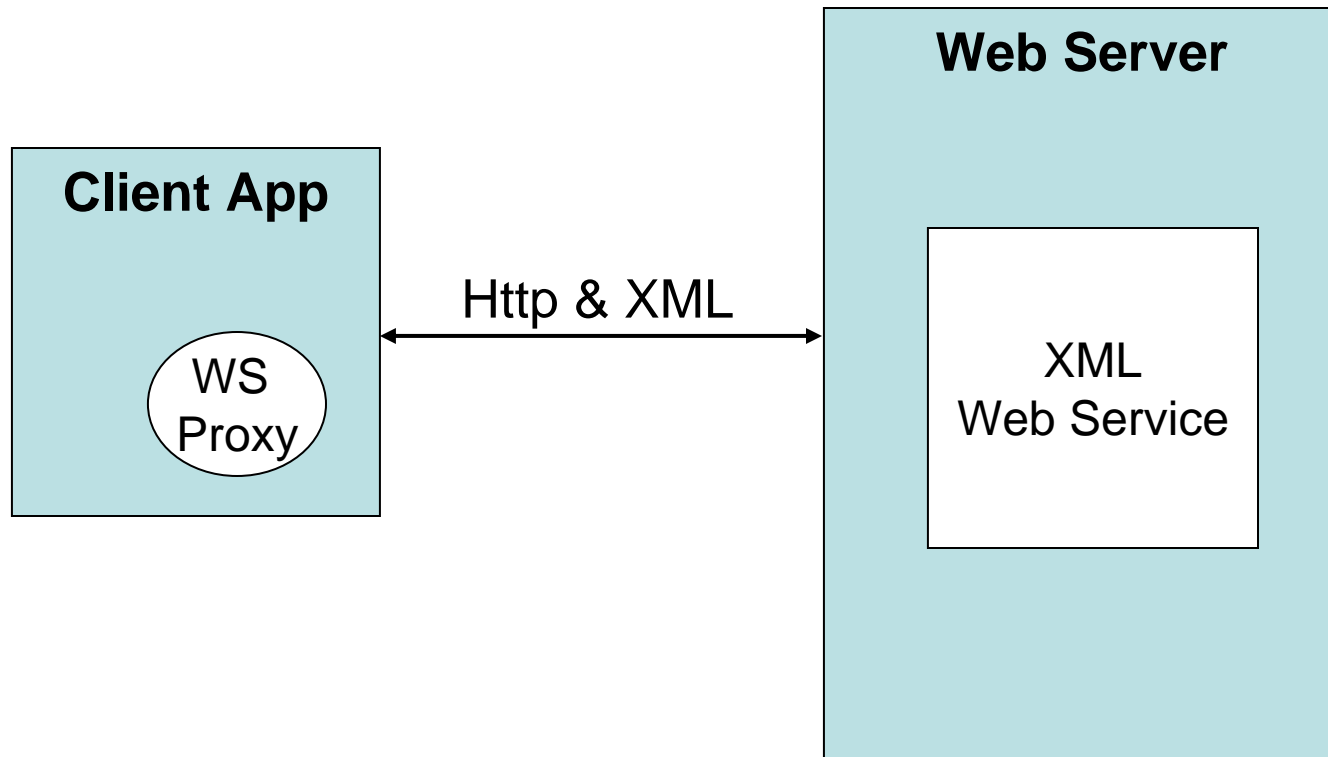


An XML web service is a unit of code hosted by a web server that can be accessed using industry standards such as HTTP and XML.

- Why?
 - cross-platform application development
 - legacy system integration



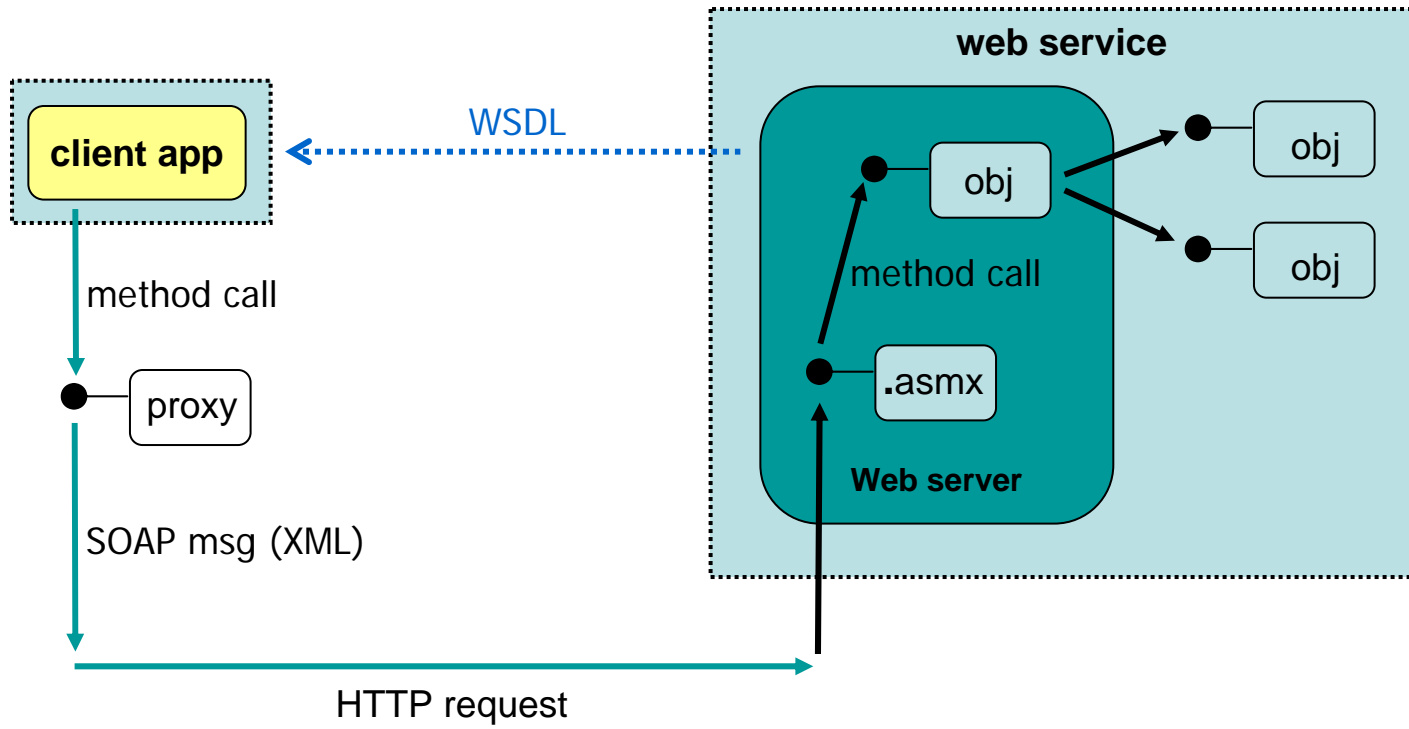
Overview



Technologies



- A discovery service - UDDI
(so clients can resolve the location of the XML web service)
- A description service - WSDL
(so clients know what the XML web service can do)
- A transport protocol - HTTP GET, HTTP Post, SOAP
(to pass the information between the client and XML web service)



Working with web service



- Two steps:
 - build a web service
 - build clients to use it

Building a XML web service by hand



Create a *.asmx file using any text editor

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

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

Test the XML web service

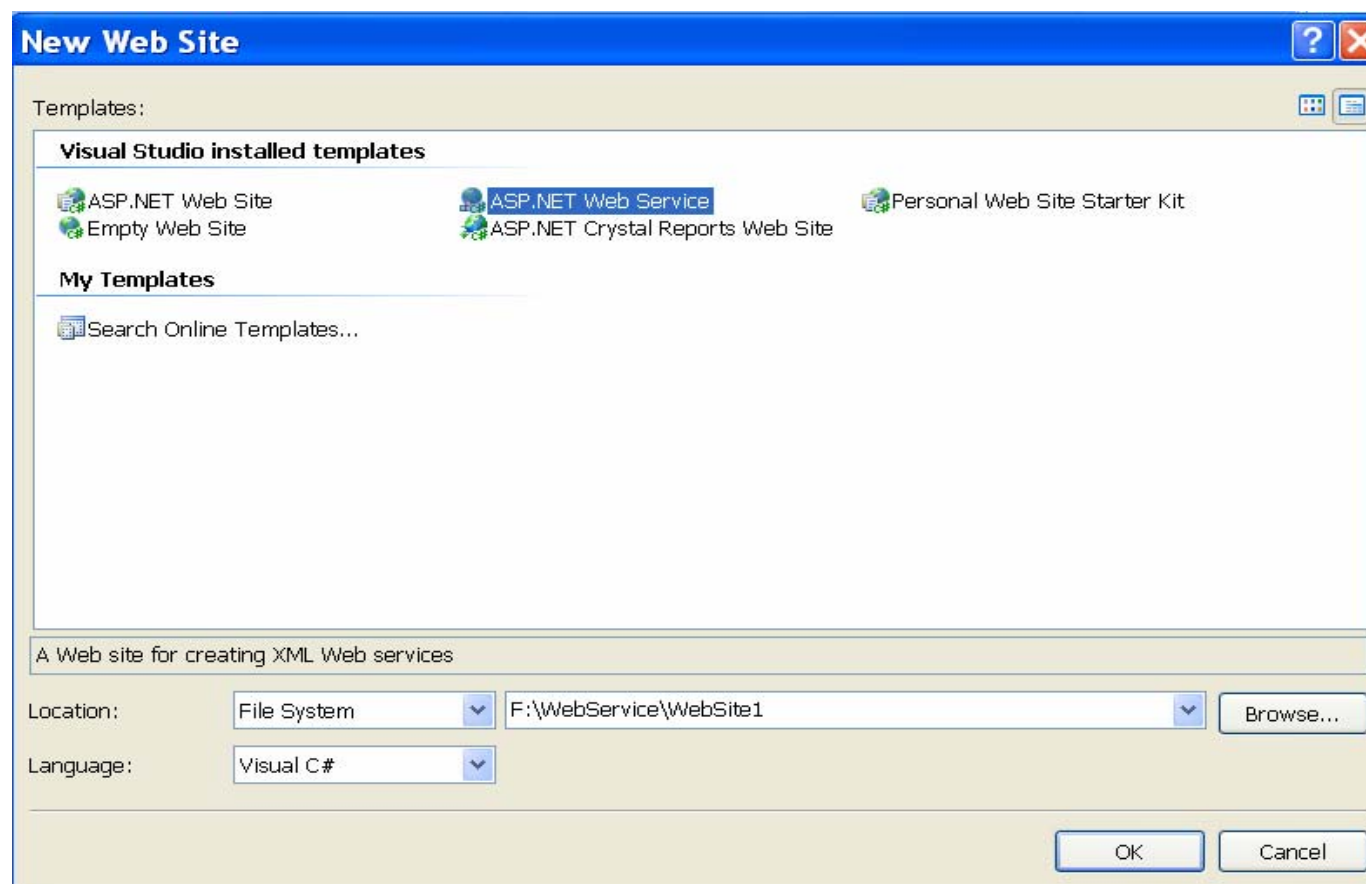


- Using WebDev.WebService.ext
`WebDev.WebServer /port:4000 /Path:"F:\WebService"`
- Using IIS
 - <http://localhost/HelloWS/HelloWorldWebService.aspx>
- The autogenerated test page
 - `DefaultWsdHelpGenerator.aspx`
(C:\Windows\Microsoft.NET\Framework\<version>\CONFIG)

Building an XML web service using visual studio 2005



- Start by creating a project of type "ASP.NET Web Service"

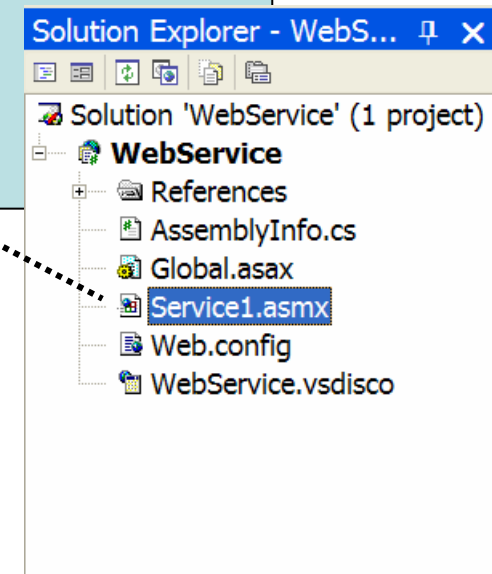


A web service is ...



- One or more objects that respond to web-based method calls
 - there is no GUI design to a web service
 - only raw classes with methods...

```
public class Service1 : System.Web.Services.WebService
{
    .
    .
    .
}
```



Example



- Looks like C#, but keep in mind these are web-based methods
 - client could be calling from any platform
 - parameters passed using XML

```
public class Service1 : System.Web.Services.WebService
{
    [WebMethod]
    public int Add(int x, int y)
    {
        return x + y;
    }

    [WebMethod]
    public string[] Attendees()
    {
        <<open DB, read attendees into array, return it>>
    }
}
```

attribute → [WebMethod]

inherit → System.Web.Services.WebService

```
using System;
using System.Text;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services; // contains Web service related
```

Set **Namespace** property of **WebService** attribute to specify the namespace that the Web service belongs to

Set **Description** property of **WebService** attribute to describe the function of the Web Service

```
namespace HugeIntegerWebService
{
    /// <summary>
    /// performs operations on large integers
    /// </summary>
    [ WebService(Namespace = "http://www.tempuri.org/",
Description = "A Web service which provides methods that" +
" can manipulate large integer values." ) ]
    public class HugeInteger : System.Web.Services.WebService
    {
        // default constructor
        public HugeInteger()
        {
            // CODEGEN: This call is required by the ASP .NET Web
            // Services Designer
            InitializeComponent();

            number = new int[ MAXIMUM ];
        }
    }
}
```

...

[WebMethod] attribute



- The [WebMethod] attribute must be applied to each method you wish to expose from an XML web service
- Avoiding WSDL name clashes via the MessageName property

Web service description language



- WSDL is an XML-based grammar that describes how external clients can interact with the web methods at a given URL, using each of the supported wire protocols.

- WSDL is used to describe the following characteristics for each exposed web method:
 - The name of the XML web method
 - The number of, type of, and ordering of parameters
 - The type of return value
 - The HTTP GET, HTTP POST, and SOAP calling conventions

Defining a WSDL document



- A valid WSDL document is opened and closed using the root `<definition>` element

`<wsdl:definition>`

`</wsdl:definition>`

The <types> element



- The <types> element contains descriptions of any and all data types exposed from the web service.

```
<s:element name="AddInt">
  <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 <message> Element



- The <message> element is used to define the format of the request and response exchange for a given web method.

```
<wsdl:message name="AddIntSoapIn">
    <wsdl:part name="parameters" element="tns:AddInt"/>
</wsdl:message>

<wsdl:message name="AddIntSoapOut">
    <wsdl:part name="parameters" element="tns:AddIntResponse"/>
</wsdl:message>
```

The <portType> element



- 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

```
wsdl:operation name="Add">  
  <wsdl:input name="AddInt" message="tns:AddIntSoapIn"/>  
  <wsdl:output name="AddInt" message="tns:AddIntSoapOut"/>  
</wsdl:operation>
```

The <binding> element



- This element specifies the exact format of the HTTP GET, HTTP POST, and SOAP exchanges.

```
<wsdl:binding name="Fortune_x0020_Predictor_x0020_Web_x0020_ServiceSoap" type="tns:Fortune_x0020_Predictor_x0020_Web_x0020_ServiceSoap">
```

```
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
```

```
<wsdl:operation name="TellFortune">
```

```
<soap:operation soapAction="http://tempuri.org/TellFortune" style="document"/>
```

```
<wsdl:input>
```

```
    <soap:body use="literal"/>
```

```
</wsdl:input>
```

```
<wsdl:output>
```

```
    <soap:body use="literal"/>
```

```
</wsdl:output>
```

```
</wsdl:operation>
```

The <service> element



- The <service> element specifies the characteristics of the web service itself. The chief duty of this element is to describe the set of ports exposed from a given web server



-
- SOAP is a wire protocol that specifies how to submit data and invoke methods across the wire using XML
 - SOAP itself does not define a specific protocol and can be used with any number of existing Internet protocols (HTTP, SMTP, and others)
 - SOAP encodes each complex method with a SOAP message.

SOAP message



- SOAP envelope
- SOAP body

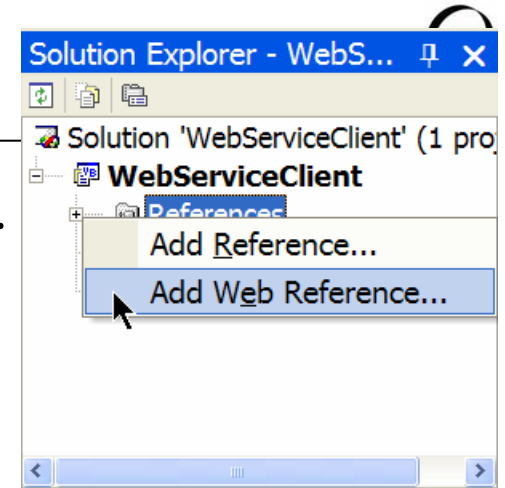
Building a client



- Start by creating a client...
 - WinForm, WebForm, console-based, anything you want!

Reference the component

- As usual, we need to reference component
 - this will activate IntelliSense
 - this will make sure we call it correctly
 - this will enable underlying XML + SOAP communication



- How?
 - project references, right-click, Add web reference...
 - type URL for web service, e.g.
 - `http://localhost/WebService/Service1.asmx`

↑
*web
server*

↑
*service
name*

↑
*class
name*

Program against component

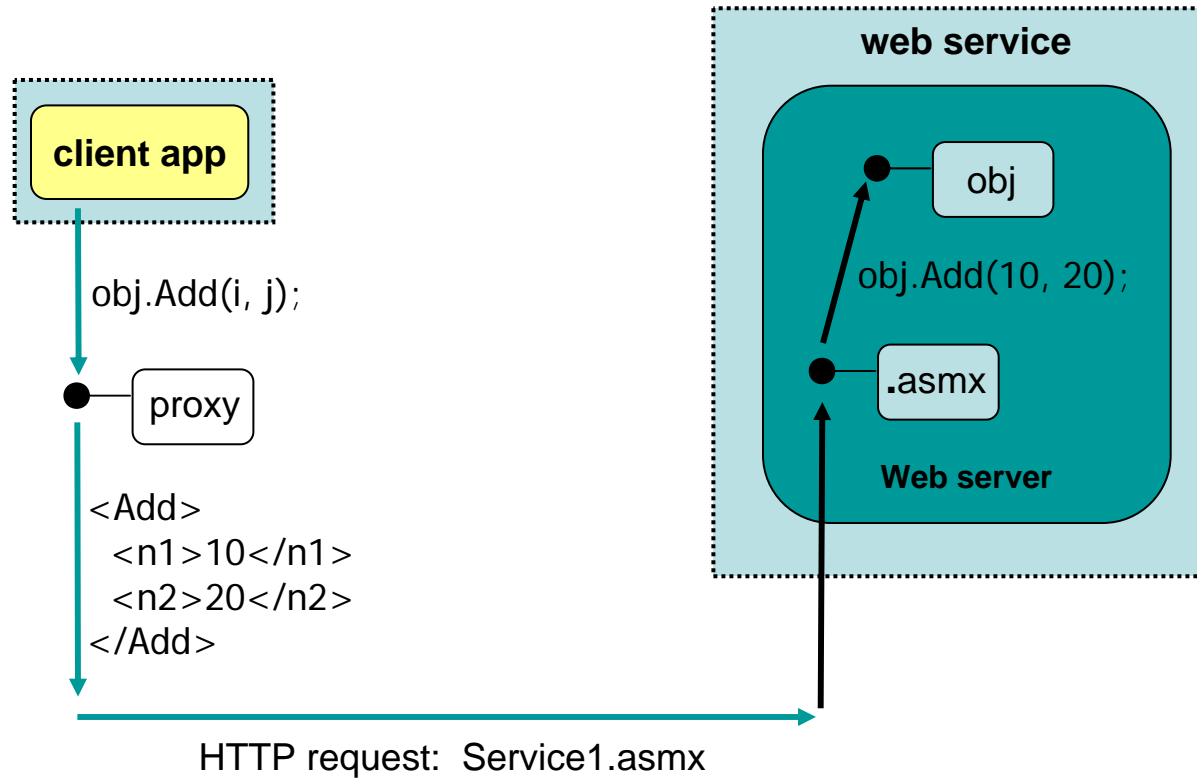


- Treat web service like any other class!
 - use new to create instances
 - make method calls
 - pass parameters

Underlying execution...



Here's what the call to `Add()` actually looks like:



Summary



- Pretty powerful stuff!

- Lots of technology be used underneath:
 - XML for parameter-passing
 - SOAP as protocol
 - HTTP
 - ASP.NET
 - IIS