



# Web Services

---

Jeffrey Miller  
Under Supervision of  
Dr. Ellis Horowitz  
Spring 2002



# Web Services Definition

---

- A web service is a piece of functionality exposed through a web interface and made accessible through standard internet protocols such as HTTP
- Web service messages are transmitted in XML format, defined by the Simple Object Access Protocol (SOAP)



# Web Services Features

---

- Web services allow a server to expose a set of functions that other computers can then call
- Platform and language independent (though not necessarily protocol-implementation independent)
- Location of client and server is irrelevant
- Firewall-friendly, since transmission occurs over the HTTP protocol, which is usually open for web servers anyway



# WSDL Document

---

- The Web Service Description Language document tells a server which functions are exposed to clients in a web service
- The WSDL document is an XML document that provides a client with all of the necessary information to invoke the web service offered by the server
- WSDL documents are very long and verbose. An example WSDL document can be found at <http://www.imaginarytechnology.com/sort/sort.asmx?WSDL>
- WSDL documents can be written by a user, though Microsoft's .NET framework provides a tool to automatically generate a WSDL document from a given web service (wsdl.exe)



# UDDI

---

- Universal Description, Discovery, and Integration is a registry that is intended to contain a repository of web services where users can search for a web service
- The search would return a DISCO document, which is an XML document that provides the location of the WSDL document for the given web service
- An example UDDI registry can be found at <http://www-3.ibm.com/services/uddi/find>



# Platform Independence

---

- The platforms of the client and server are transparent (similar to browsing the web, where the client browser could be running Internet Explorer in Windows and the web server could be running Apache on Solaris)



# Language Independence

---

- The client and server applications can both be written in any language that is capable of packaging and parsing an XML document, and transmitting and receiving messages over a network
- Two languages widely used for this are C# and Java



# Protocol

---

- SOAP, as defined by W3C, is an XML-based lightweight protocol used for exchange of information in a decentralized, distributed environment
- SOAP messages are transmitted over the HTTP or HTTPS protocol (port 80 or port 443, respectively)
- SOAP is a special type of XML
- This implies that the messages generated and the protocol to transmit these messages are platform independent



# Protocol Implementation

---

- The format of the SOAP message that the web service will receive has not yet been standardized across implementations of the protocol
- W3C has developed a standard for SOAP messages, but the SOAP vendors (i.e. Microsoft, Apache, IBM) do not all implement the protocol exactly in compliance with the W3C specification
- This means that, at least for now, the web service protocol implementation must be known by a client calling the web service so that the SOAP message can be properly created



# Differences

---

- Microsoft and Apache have implemented SOAP slightly different, so a client that has created a message for a web service using Apache SOAP will have to be modified to call the same web service using Microsoft SOAP



# Microsoft SOAP Request

---

POST [/sort/sort.asmx](#) HTTP/1.0  
Content-Type: text/xml  
SOAPAction: <http://www.imaginarytechnology.com/sort/sort>  
User-Agent: Java1.2.2  
Host: nunki.usc.edu:11069  
Accept: text/html, image/gif, image/jpeg, \*; q=.2, \*/\*; q=.2  
Content-length: 552

```
<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>
    <m:sort xmlns:m="http://www.imaginarytechnology.com/sort/">
      <m:numbers>
        <m:int>21</m:int> <m:int>65</m:int>
        <m:int>87</m:int> <m:int>78</m:int>
        <m:int>45</m:int>
      </m:numbers>
    </m:sort></soap:Body></soap:Envelope>
```



# Apache SOAP Request

---

POST /soap/servlet/rpcrouter/ HTTP/1.0  
Content-Type: text/xml  
SOAPAction: urn:sort  
User-Agent: Java1.2.2  
Host: nunki.usc.edu:11069  
Accept: text/html, image/gif, image/jpeg, \*, q=.2, \*/\*; q=.2  
Content-length: 798

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"  
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"  
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">  
  <SOAP-ENV:Body>  
    <m:sort xmlns:m="urn:sort"  
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">  
      <numbers xmlns:ns="http://schemas.xmlsoap.org/soap/encoding/"  
        xsi:type="ns:Array" ns:arrayType="xsd:int[5]">  
        <item xsi:type="xsd:int">21</item><item xsi:type="xsd:int">65</item>  
        <item xsi:type="xsd:int">87</item><item xsi:type="xsd:int">78</item>  
        <item xsi:type="xsd:int">45</item>  
      </numbers></m:sort></SOAP-ENV:Body></SOAP-ENV:Envelope>
```



# Microsoft SOAP Response

---

HTTP/1.1 200 OK  
Server: Microsoft-IIS/5.0  
Date: Sun, 28 Apr 2002 06:58:37 GMT  
Cache-Control: private, max-age=0  
Content-Type: text/xml; charset=utf-8  
Content-Length: 414

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <sortResponse
      xmlns="http://www.imaginarytechnology.com/sort/">
      <sortResult>
        <int>21</int><int>45</int><int>65</int>
        <int>78</int><int>87</int>
      </sortResult></sortResponse></soap:Body></soap:Envelope>
```



# Apache SOAP Response

---

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: 690

Date: Sun, 28 Apr 2002 06:31:55 GMT

Server: Apache Tomcat/4.0.3 (HTTP/1.1 Connector)

Set-Cookie: JSESSIONID=4E68BC1513006B5BFFD44421832255EF;Path=/soap

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:sortResponse xmlns:ns1="urn:sort"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <return xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
        xsi:type="ns2:Array" ns2:arrayType="xsd:int[5]">
        <item xsi:type="xsd:int">21</item> <item xsi:type="xsd:int">45</item>
        <item xsi:type="xsd:int">65</item> <item xsi:type="xsd:int">78</item>
        <item xsi:type="xsd:int">87</item>
      </return></ns1:sortResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>
```



# Microsoft vs. Apache

	<b>Microsoft</b>	<b>Apache</b>
Web Service Product	.NET Framework	Tomcat 4.0.3
Web Service Authoring Language	Many, including C, C++, C#, Cobol, Visual Basic	Java
Creation/Deployment Tool	GUI or Command Line	Command Line
Platforms Supported	Windows NT4.0, 2000 or higher only, though you may port to other platforms on your own	Most Windows and Unix-based platforms



# Sort Example

---

- This example is focused more on the details of writing a web service and a client application, and less on the algorithm of the web service
- This web service, which has been written on a Windows 2000 platform using .NET and a Sun OS platform using Java, takes an array of integers and returns a sorted array



# Windows 2000 Web Service

---

- First, we need to write the code for the web service. Since we are writing this web service in Windows 2000, we will use C#, which is part of the .NET framework
- .NET allows a web service to be written in-line within an ASP.NET file (similar to an ASP file), which then allows for easy viewing of the WSDL document and an automatically generated test client accessible through a browser



# ASP.NET

---

- An ASP.NET file has a .asmx extension
- If you are writing a web service within an ASP.NET file, there is only one line of ASP code, and the rest is the web service, written in whatever .NET language specified in the ASP WebService tag
- To allow other people to use your web service, make this .asmx file available through your web server
- There is also no need to compile your web service - .NET will compile the service when necessary
- Note: To run .NET web services, you must have the .NET Framework SDK installed on the server
- You can download the .NET SDK from <http://msdn.microsoft.com/netframework/>



# ASP.NET/C# Web Service

---

```
<%@ WebService Language="C#" class="SortNumbers" %>
using System;
using System.Web;
using System.Web.Services;
[WebService(Namespace="http://www.imaginarytechnology.com/sort/")]
public class SortNumbers : System.Web.Services.WebService {

    [WebMethod(Description="Sorts a sequence of numbers in ascending orders")]
    public int[] sort(int [] numbers) {
        return sortNumbers(numbers);
    }

    private int[] sortNumbers(int [] numbers) {
        int temp;
        for (int i=0; i < numbers.Length; i++) {
            for (int j=0; j < i; j++) {
                if (numbers[i] < numbers[j]) {
                    temp = numbers[i];
                    numbers[i] = numbers[j];
                    numbers[j] = temp;
                }
            }
        }
        return numbers;
    }
}
```



# ASP.NET WebService Tag

---

- The ASP.NET WebService tag is used to specify the class name of the web service (required), the language in which the web service is written (optional, which defaults to C# if no language is given), and the CodeBehind attribute (optional, which specifies where web service code is located if it is in an external file)



# C# Web Service Code

---

- The C# code is similar to Java, but to make the class a web service, a few steps are required:
  1. Any class that is to be a web service can be immediately preceded by a `WebService` attribute, which is used in WSDL generation to provide more information to the client application
  2. Any method that is to be exposed in the web service must be public and be immediately preceded by a `WebMethod` attribute
  3. No other methods in the class are exposed to the web service, but may be used in the accessible methods



# WebService Attribute Properties

---

- Name – This property is not restricted by the Common Language Runtime (CLR) identifier naming rules. The name will be seen in the WSDL document
- Namespace – Change the default XML namespace
- Description – Describe the web service



# WebMethod Attribute Properties

---

- `BufferResponse` – if true, buffer the entire response; else, buffer response in 16Kb chunks
- `CacheDuration` – cache results for a certain number of seconds, or 0 for no cache
- `Description` – use in web service's help page
- `EnableSession` – if true, enable session state information through `WebService.Session` or `HttpContext.Current.Session` property
- `MessageName` – allows web service to uniquely identify overloaded methods using an alias
- `TransactionOption` – allows web service to participate as the root object of a transaction by being set to `TransactionOption=TransactionOption.RequiresNew`



# Client Proxy Class

---

- Instead of having the client code call the web service directly, .NET allows for an automatically generated proxy class which serves as the link between the client code and the web service
- This proxy class packages and parses the messages so that the client does not have to worry about the messages that are traveling across the network
- To generate the proxy class, you either need the WSDL document or the location of the web service
- The following command will generate the proxy class for you: `wsdl <webservice or WSDL location>`
- Note: The client machine needs to have the .NET Framework SDK installed to run this utility



# Web-Based Client

---

- If your client is web-based and you are not using a proxy class, the SOAP message returned will be displayed in the browser as a parsed XML document (for Internet Explorer version 5.5 and greater)
- Other browsers, such as Netscape, do not have built-in XML parsers, and therefore do not display the XML document parsed (or sometimes they throw error messages)



# Windows Client Application

---

- For the sort client, we wrote the client code to be accessed through a browser as a Java Server Page (JSP). This code could be running on any platform that supports JSPs, though it will connect to the Windows web service.
- Sun has a JSP IO tag library, which is used in this example to package the SOAP message that is traveling to the web service



# Windows Client Application Code

```
<%@ taglib uri="http://jakarta.apache.org/taglibs/io" prefix="io" %>
<%@ page contentType="text/xml" %>
<% int [] array1 = {21, 65, 87, 78, 45}; %>
<io:soap url="http://www.imaginarytechnology.com/sort/sort.asmx"
  SOAPAction="sort">
  <io:header name="SOAPAction"
    value="http://www.imaginarytechnology.com/sort/sort"/>
  <io:pipe>
    <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>
        <m:sort xmlns:m="http://www.imaginarytechnology.com/sort/">
          <m:numbers>
            <% for (int i=0; i < array1.length; i++) { %> <m:int><%=array1 [i]%></m:int>
            <% } %>
          </m:numbers>
        </m:sort>
      </soap:Body>
    </soap:Envelope>
  </io:pipe>
</io:soap>
```



# Windows SOAP Request

---

POST /sort/sort.asmx HTTP/1.0  
Content-Type: text/xml  
SOAPAction: http://www.imaginarytechnology.com/sort/sort  
User-Agent: Java1.2.2  
Host: nunki.usc.edu:11069  
Accepts: text/html, image/gif, image/jpeg, \*; q=.2, \*/\*; q=.2  
Content-length: 552

```
<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>
    <m:sort xmlns:m="http://www.imaginarytechnology.com/sort/">
      <m:numbers>
        <m:int>21</m:int> <m:int>65</m:int>
        <m:int>87</m:int> <m:int>78</m:int>
        <m:int>45</m:int>
      </m:numbers>
    </m:sort></soap:Body></soap:Envelope>
```



# Windows SOAP Response

---

HTTP/1.1 200 OK  
Server: Microsoft-IIS/5.0  
Date: Sun, 28 Apr 2002 06:58:37 GMT  
Cache-Control: private, max-age=0  
Content-Type: text/xml; charset=utf-8  
Content-Length: 414

```
<?xml version="1.0" encoding="utf-8" ?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body>
      <sortResponse
        xmlns="http://www.imaginarytechnology.com/sort/">
        <sortResult>
          <int>21</int><int>45</int><int>65</int>
          <int>78</int><int>87</int>
        </sortResult></sortResponse></soap:Body></soap:Envelope>
```



# Sun OS Web Service

---

- The first step is to write the web service, which we will write in Java since we are running Tomcat4.0.3 on a Unix-based machine
- The code for the web service is not any different than any other class written in Java
- Compile the code after you have written the web service class



# Java Web Service

---

```
public class sort {  
    public int[] sort (int [] numbers) {  
        int temp;  
        for (int i=0; i < numbers.length; i++) {  
            for (int j=0; j < i; j++) {  
                if (numbers[i] < numbers[j]) {  
                    temp = numbers[i];  
                    numbers[i] = numbers[j];  
                    numbers[j] = temp;  
                } } }  
        return numbers;  
    }  
}
```



# Document Descriptor

---

- To deploy a web service using Tomcat, a document descriptor must be created, which is merely an XML document with a few required tags
- The id attribute of the service tag gives the name of the web service that is exposed to the client
- The methods attribute of the provider tag, which is a child of the service tag, gives the name of the methods that are exposed to the client
- The class attribute of the java tag, which is a child of the provider tag, gives the name of the class that is exposed to the client
- The mappings tag, which is a child of the service tag, provides mappings from Java objects to the equivalent SOAP objects if SOAP does not already know how to convert the data type
- All primitive Java data types, primitive arrays, and Strings are known by the Apache SOAP implementation



# Sort Document Descriptor

---

```
<dd:service xmlns:dd="http://xml.apache.org/xml-  
soap/deployment" id="urn:sort">  
  <dd:provider type="java" scope="Application"  
    methods="sort">  
    <dd:java class="sort" static="false" />  
  </dd:provider>  
  <dd:faultListener>  
    org.apache.soap.server.DOMFaultListener  
  </dd:faultListener>  
  <dd:mappings />  
</dd:service>
```



# Deploying the Web Service

---

- The ServiceManagerClient class, which is packaged with Tomcat 4.0.3, is used to deploy web services based on the document descriptor
- Assuming sort.xml is the name of the document descriptor for our web service, Tomcat is installed on port 8080 of nunki.usc.edu, and all of the jar files that came with Tomcat are in the system classpath, type the following command:
- ```
java org.apache.soap.server.ServiceManagerClient  
http://nunki.usc.edu:8080/soap/servlet/rpcrouter  
deploy sort.xml
```
- It is then necessary to restart Tomcat before the new web service will be recognized



# Apache Client Application

---

- The client application for our sort web service is a Java Server Page (JSP) with a hard-coded array of integers to sort. A variable number of integers to sort using different client and web service implementations can be seen at <http://nunki.usc.edu:11069/webservices.html>



# Apache Client Proxy

---

- Apache has a form of a client proxy, although it does not work exactly the same as the Windows proxy
- The `org.apache.soap` and `org.apache.soap.rpc` packages provide classes for packaging the SOAP message with Java variable data
- The classes in these packages then parse the SOAP response and return only the data back to the client application in the form of a `Response` class
- This eliminates all packaging and parsing of SOAP messages communicating with the web service



# Apache Client Application Code

---

```
<%@ page language="java" import="java.io.*, java.net.*, java.util.*, org.apache.soap.*,
    org.apache.soap.rpc.*" %>
<%
    URL url = new URL("http://nunki.usc.edu:11069/soap/servlet/rpcrouter");
    Call call = new Call();
    call.setTargetObjectURI("urn:sort");
    call.setMethodName("sort1");
    call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
    Vector params = new Vector();
    int [] array1 = {21, 65, 87, 78, 45};
    params.addElement(new Parameter("numbers", int[].class, array1,
        Constants.NS_URI_SOAP_ENC));
    call.setParams(params);
    Response resp = call.invoke(url, "");
    if (resp.generatedFault()) {
        Fault fault = resp.getFault(); // generate fault information here
    }
    else {
        Parameter result = resp.getReturnValue();
        result.setEncodingStyleURI(Constants.ATTR_ARRAY_TYPE);
        int [] sortedarray = (int[])result.getValue(); // contains sorted array
    }
%>
```



# Apache SOAP Request

---

POST /soap/servlet/rpcrouter/ HTTP/1.0  
Content-Type: text/xml  
SOAPAction: urn:sort  
User-Agent: Java1.2.2  
Host: nunki.usc.edu:11069  
Accept: text/html, image/gif, image/jpeg, \*; q=.2, \*/\*; q=.2  
Content-length: 798

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"  
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"  
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">  
  <SOAP-ENV:Body>  
    <m:sort xmlns:m="urn:sort"  
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">  
      <numbers xmlns:ns="http://schemas.xmlsoap.org/soap/encoding/"  
        xsi:type="ns:Array" ns:arrayType="xsd:int[5]">  
        <item xsi:type="xsd:int">21</item><item xsi:type="xsd:int">65</item>  
        <item xsi:type="xsd:int">87</item><item xsi:type="xsd:int">78</item>  
        <item xsi:type="xsd:int">45</item>  
      </numbers></m:sort></SOAP-ENV:Body></SOAP-ENV:Envelope>
```



# Apache SOAP Response

---

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: 690

Date: Sun, 28 Apr 2002 06:31:55 GMT

Server: Apache Tomcat/4.0.3 (HTTP/1.1 Connector)

Set-Cookie: JSESSIONID=4E68BC1513006B5BFFD44421832255EF;Path=/soap

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:sortResponse xmlns:ns1="urn:sort"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <return xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
        xsi:type="ns2:Array" ns2:arrayType="xsd:int[5]">
        <item xsi:type="xsd:int">21</item> <item xsi:type="xsd:int">45</item>
        <item xsi:type="xsd:int">65</item> <item xsi:type="xsd:int">78</item>
        <item xsi:type="xsd:int">87</item>
      </return></ns1:sortResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>
```



# Conclusion

---

- Web Services provide a new method of distributed computing
- Using standard protocols, the theory behind web services makes it appear as if a client does not need to concern itself with the platform running the web service
- Although this is somewhat true, the implementation of the SOAP protocol does need to be known, as can be seen by the differences between the SOAP messages that need to be received by the web services, and the messages returned from the web services, on the different SOAP implementations



# Conclusion

---

- When objects start being used, the protocol implementation becomes even more critical because those object types must be defined on the server running the web service so that it knows how to decode the object into the types SOAP is capable of understanding
- Even in our small example using arrays, we saw that the Microsoft and Apache implementations of SOAP drastically differed for sending and receiving the data



# Conclusion

---

- Web services will most likely become the next paradigm for distributed computing, overcoming the platform dependency issues with RPC and COM
- However, if the different companies implementing the SOAP protocol do not conform to a standard, web services will lose one of the key features underlying the technology: the ability for different platforms and different languages to seamlessly integrate over a network.