o

ibm.com/developerWorks/
webservices

Building SOAP applications

Doug Tidwell

IBM developer**Works**

dtidwell@us.ibm.com

o

Agenda

- SOAP overview
- The Axis toolkit
- Handling faults and exceptions
- Advanced SOAP
- Writing clients in other languages
- Resources

o

Key Messages

- **Web services are a new style of computing.**
  - They will change software development as significantly as the Web itself.
  - If you ignore them, your competitors will thank you, possibly over drinks after they acquire what's left of your company.
- **Web services are real…**
  - …but not finished yet. We'll make it clear what's **really** real and what isn't.

o

SOAP overview

o

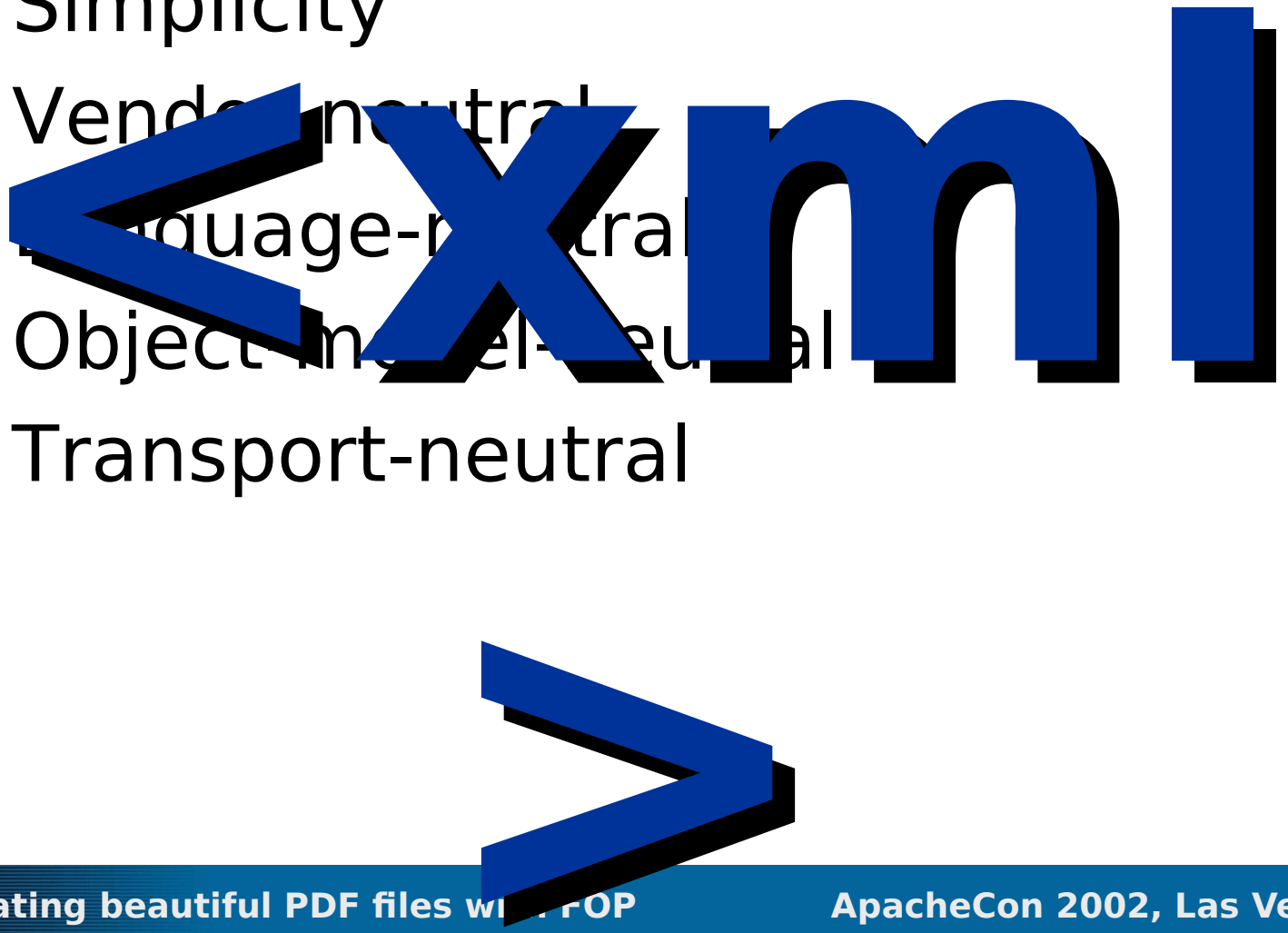
SOAP

- The Simple Object Access Protocol
- Originally developed by Microsoft, UserLand, developMentor, etc.
- 1.0 perceived as too Windows-centric
- IBM joined the fray to help make SOAP vendor- and platform-neutral
- Find IBM's second-generation SOAP implementation at **`xml.apache.org`**.

o

SOAP design

- Simplicity
- Vendor-neutral
- Language-neutral
- Object model-neutral
- Transport-neutral



o

SOAP message structure

- There are request and response messages
  - A request invokes a method on a remote object
  - A response returns the result of running the method

o

Envelopes

- A SOAP envelope contains the message itself.
- The message is in an application-specific vocabulary; namespaces are used to distinguish the parts.

o

A SOAP request

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="…"
   SOAP-ENV:encodingStyle="..">
   <SOAP-ENV:Body>
      <m:GetLastTradePrice xmlns:m="my-ns">
         <symbol>IBM</symbol>
      </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

o

A SOAP response

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="…"
   SOAP-ENV:encodingStyle="..">
   <SOAP-ENV:Body>
      <m:GetLastTradePriceResponse
        xmlns:m="my-ns">
         <price>65.25</price>
      </m:GetLastTradePriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

o

Application view

- If your application is using SOAP to invoke a service, you need to build the request and parse the response.
- You can use XSLT, SAX, DOM, or JDOM to convert the XML data into anything else.
- As tools improve, much of the XML parsing will disappear.

o

SOAP specs

- SOAP V1.2
  - **http://www.w3.org/TR/SOAP/**
- SOAP Messages with Attachments
  - **http://www.w3.org/TR/SOAP-attachments/**
- SOAP Security Extensions: Digital Signature
  - **http://www.w3.org/TR/SOAP-dsig/**

o

The Magic Eight Ball

A small Web service

-or-

American Garbage Culture 101

o

The Magic Eight Ball

- “50,000 years of technological advancement has culminated in a system to bring you mysticism on demand.”
- Feel free to shout your questions...

o

The service code

- **`EightBall.java`** consists of an array of 20 phrases and a random number generator.
- Nothing in the code is XML-aware, SOAP-aware, network-aware, etc.
- We can still take this code and deploy it as a SOAP service.

o

The service code

- The code only uses basic Java classes:

```
import java.util.Random;
import java.lang.Double;
import java.util.Date;
```

o

Deploying the service

- This is done with a **deployment descriptor** file. It contains:
  - An ID for the service
  - The thing (Java class, JavaScript, PerlScript, etc.) providing the service
  - The methods it supports
- **In Axis, you can also do this with a .JWS file.**

o

Simple deployment descriptor

```
<deployment
   xmlns="http://xml.apache.org/axis/wsdd/"
   xmlns:java="http://xml.apache.org/axis/wsdd
    /providers/java">

   <service name="urn:EightBall"
            provider="java:RPC">
     <parameter name="className"
                value="EightBall"/>
     <parameter name="allowedMethods"
                value="getAnswer askQuestion"/>
   </service>

</deployment>
```

ApacheCon U.S. 2002
THE ONLY SURE THING IN LAS VEGAS!
Las Vegas, Nevada November 18-21, 2002

ibm.com/developerWorks/
webservices

o

Deploying the service

- Axis provides a utility class, **`org.apache.axis.client.AdminClient`**, that processes the deployment descriptor.
- You also need to copy your **`.class`** file to the appropriate directory, such as the Tomcat **`classes`** directory.

o

Deploying the service

- **We didn’t change any of our original code** to deplore it as a SOAP service.
  - We also didn’t have to think about SOAP when we wrote or designed the code.
- It’s likely you can take the useful applications you already have and deploy them as SOAP services as well.

o

Accessing the service

- To access the SOAP service, you need five things:
  - The URL of the host machine
  - The ID of the service
  - The name of the method
  - The parameters for the method (if any)
  - The value of the **`SOAPAction`** field
- This is similar to the information we needed to deploy the service.

o

The client code

- First, we create an Axis `Call` object:

```
Call call = (Call)
  service.createCall();
```

o

The client code

- We set the URL of the service (the endpoint, in Axis terms):

```
String
  endpt="http://localhost:8070…";
call.setTargetEndpointAddress
  (new URL(endpt));
```

The client code

- We set the method name (the operation name, in Axis terms):

```
call.setOperationName (new
  QName("urn:EightBall",
  "getAnswer"));
```

- Notice that this call contains both the namespace (service ID) and the method name.

o

The client code

- To simplify things, we'll set the return type of our method on the `Call` object:

```
call.setReturnType(org.apache.
  axis.encoding.XMLType.
  XSD_STRING);
```

o

The client code

- Now we're ready to invoke the service, so we'll use the **`Call.invoke`** method:

```
call.invoke(new Object[]{ });
```

- The Axis toolkit does the tedious work for us:
  - Building the SOAP request
  - Sending the SOAP request to the correct URL
  - Parsing the SOAP response

o

Passing parameters

- You can pass any parameters to the **`Call`** object.
- This code adds a parameter named **`question`** of type **`String`** (from the XML Schema datatypes spec):

```
call.addParameter("question",
  org.apache...XMLType.XSD_STRING,
  ParameterMode.IN);
```

o

Passing parameters

- If you pass parameters on the **`Call`** object, you must use the **`Call.setReturnType`** method to define what kind of data is returned by the service.

Passing parameters

- To pass the parameters, simply put the objects inside the array when you use the **`invoke`** method:

```
call.invoke(new Object[]
  {"Will I win the lotto?"});
```

- We'll talk about passing things other than simple datatypes in a minute...

o

Passing parameters

- If you just pass some parameters on the **`invoke`** method, there must be a method with that signature in the code that provides the service.
- If you don't use **`Call.addParameter`**, the parameter names will be **`<arg0>`**, **`<arg1>`**, etc.

o

Passing parameters

- If the **`invoke`** method doesn't have the same number and type of parameters as the **`addParameter`** calls, Axis throws an exception.

o

Magic Eight Ball resources

- **`http://ofb.net/8ball/`**
  - A developer site, really, full of 8-ball arcana, the official phrases, how to take an 8-ball apart, etc.
- **`http://8ball.federated.com/`**
  - My personal favorite, a Linux box, a Magic 8-ball, a Web cam, and a Lego MindStorms robot.

o

The Axis project

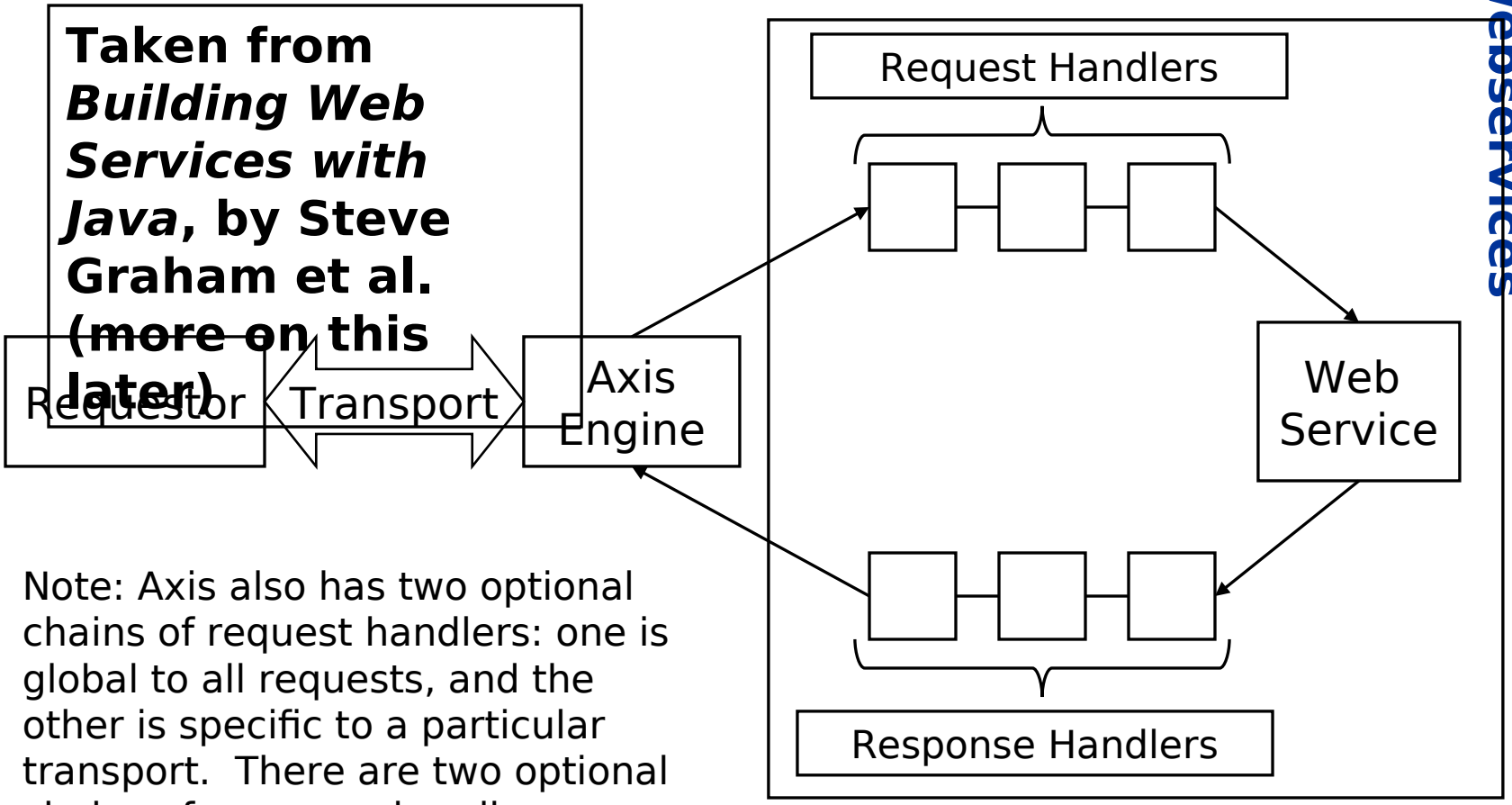
The Axis project

- Based at **`xml.apache.org/axis`**, the Axis toolkit has several important goals:
  - Better performance
  - More flexible architecture
  - Easier to use other transports
  - Make it easy to include other code in message handling (encryption, logging, authentication, etc.)

o

Axis architecture

Taken from *Building Web Services with Java*, by Steve Graham et al. (more on this later)



ibm.com/developerWorks/webservices

Note: Axis also has two optional chains of request handlers: one is global to all requests, and the other is specific to a particular transport. There are two optional chains of response handlers as well.

Web service specific chain

o

Axis components

- Axis is made of several components, all of which are interchangeable:
  - The Axis engine itself
  - Handlers (for logging, authentication, etc.)
  - Chains (a series of handlers)
  - Transports (HTTP, FTP, JMS, etc.)

o

The engine

- The Axis engine is the entry point to the message processing system.
- For flexibility, it can be replaced, although most of your applications will use the **`AxisEngine`** that's built in to the toolkit.

o

Handlers

- A handler's job is to look at a SOAP message and (maybe) modify it.
  - A handler might simply log how many times a given Web service has been invoked
  - A handler might look for encrypted data inside a SOAP message and decrypt it.
- Handlers are the basic building blocks in Axis.

ibm.com/developerWorks/webservices

o

Handlers

- If a handler wants to stop a transaction, it throws an **`AxisFault`**. Here's how you might process a **`userID`** parameter that's either missing or blank:

```
if (userID = null || userID = "")
    throw new AxisFault(…);
```

- Axis sends a SOAP fault to the client.

o

Deploying the service

- You associate handlers, chains, etc. with a particular Web service in the deployment descriptor. Here's an example:

```
<service name="urn:Services:EightBall"
 provider="java:rpc">
 <requestFlow>
  <handler type="java:org.apache.axis.
 handlers.HeaderAuthenticationHandler"/>
 </requestFlow>
```

o

Chains

- A chain is simply a sequence of handlers.
  - A chain might be an authentication handler, followed by a logging handler, followed by an unencryption handler.

o

Transports

- Although the most common transport is HTTP, Axis supports other transports, such as FTP and SMTP.
  - FTP and SMTP only make sense for one-way messaging.
- If you like, you can also create your own transport handler.

o

WSDL

- The Web Services Description Language is a key technology for Web services.
- You can use WSDL to generate most of a client application or most of a service.
- WSDL also plays a key role in Web service discovery...

o

Axis support for WSDL

- Axis supports WSDL in three ways:
  1. You can get the WSDL file associated with a service by opening the url **`http://my-service's-url?WSDL`**. (Axis generates the file dynamically.)
  2. You can use the **`Java2wsdl`** tool to generate a WSDL file for your Web service.
  3. You can use **`Wsdl2java`** to generate Java proxies and skeletons from existing WSDL files.

ibm.com/developerWorks/webservices

o

Handling errors

What to do when something goes wrong

o

Debugging SOAP apps

- The Axis toolkit includes a SOAP sniffer.
  - **`java org.apache.axis.utils.tcpmon 8070 localhost 8080`**
  - This tells the sniffer to monitor port 8070, and send all requests to **`http://localhost:8080`**.
- You can see the SOAP requests and responses as they go.

o

SOAP faults

- The SOAP spec defines a set of fault codes that indicate where something went wrong.
- In addition to faults, SOAP applications written in Java can throw exceptions.

o

SOAP faults

- We'll look at a SOAP application that handles faults.
  - If we're using Axis, a SOAP fault maps to a Java exception, usually **`java.rmi.RemoteException`**.

o

SOAP exceptions

- Your Java client may get a SOAP exception; they’re handled just like any other exception.
- A SOAP exception typically means that something has gone very wrong outside your application:
  - Network troubles
  - SOAP server configuration problems
  - etc.

o

Advanced SOAP

o

Advanced SOAP

- All of the SOAP applications we've written to this point used the basic features.
- We'll take a look at some examples that use more advanced facilities of SOAP.

o

SOAP headers

- A SOAP header is a flexible way of adding features to a SOAP message.
- With Axis, you can set up a handler to process the SOAP header; the handler can take care of authentication, logging, etc., while the SOAP body doesn't change.

o

A SOAP header

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="..."
  SOAP-ENV:encodingStyle="...">
   <SOAP-ENV:Header>
      <m:authentication
        xmlns:m="my-ns">
         <username>dtidwell</username>
         <password>apachecon</password>
      </m:authentication>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>...</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

ibm.com/developerWorks/webservices

ApacheCon U.S. 2002
THE ONLY SURE THING IN LAS VEGAS!
Las Vegas, Nevada November 18-21, 2002

Getting the header

```
SOAPEnvelope env = msgContext
  .getCurrentMessage()
  .getSOAPEnvelope();
SOAPHeaderElement username =
 env.getHeaderByName
 ("http://www.apache.org/",
 "username");
```

- To get the header element, we ask for the element and its namespace.

ibm.com/developerWorks/webservices

o

SOAP headers

- There are two attributes that can appear on the elements inside a SOAP header:
  - **`mustUnderstand`**
  - **`actor`**

o

mustUnderstand

- The **mustUnderstand** attribute can have a value of 0 or 1. If the value is 1, then whatever processes the header must support that element:

```
<authentication
  SOAP-ENV:mustUnderstand="1"> …
```

- The SOAP service must return a **MustUnderstand** fault if it doesn't support **<authentication>**.

o

actor

- The **actor** attribute specifies a SOAP provider that will process a given element.
- A SOAP message may pass through a number of intermediaries as it goes from the client to the service. The SOAP **actor** attribute lets you specify which intermediary should process the header.

o

Marshalling

- To this point, we've passed things like numbers and strings in the SOAP envelopes.
- What if we need to send a data structure or an object? That's what marshalling is for.

o

Serializers

- The Axis toolkit lets you define your own classes for serializing and deserializing objects.
- Take a look at the **`BeanSerializer`** class for an example. This converts any Java bean into an XML document, or vice versa.
- Axis also has serializers for the data types defined in the XML Schema and SOAP specs, including **`base64Binary`** for binary data.

o

Serializers

- We'll look at an example that sends XML elements in the SOAP envelope.
- We'll also look at deploying a SOAP service that defines serializers and deserializers for Java objects.

o

Asynchronous SOAP

- Everything we've done to this point has been a remote procedure call.
- SOAP messages can be passed asynchronously.
- We'll look at an example that uses SOAP over a message queueing system.

o

SOAP over other transports

- Everything we've done to this point was SOAP over HTTP.
- SOAP messages can be passed over any transport. Commonly supported transports are HTTP, FTP, SMTP, Jabber, and IM.

o

SOAP and encryption

- SOAP messages themselves can be encrypted and/or digitally signed.
- If someone else intercepts the message, they can’t make any sense of it.
- If someone else tries to alter the message, the digests won’t match.

o

SOAP and attachments

- Axis supports the proposed SOAP Messages with Attachments spec.
- This allows you to send binary data as an attachment in a MIME multipart message.

o

SOAP in multiple languages

A demonstration of interoperability

o

Multiple languages

- Interoperability is the major value of Web services. To emphasize this, we'll look at SOAP clients and services written in a variety of languages:
  - Java
  - C#
  - Perl
  - Python
  - VisualBasic
  - C++

o

Java

- Here's some of a Java client, written with the Axis APIs:

```
Service service = new Service();
Call call = (Call) service.createCall();
call.setTargetEndpointAddress(new
   java.net.URL(
   "http://localhost:8070/axis/services"));
call.setOperationName(new
   QName("urn:EightBall", "getAnswer"));
call.setReturnType(org.apache...XSD_STRING);
System.out.println(call.invoke(new Object[]{ }));
```

o

C#

- Written for the .NET framework:

```
[System.Web.Services.Protocols.SoapDocumentMethodAt
   tribute("http://www.EightBall.org/services/
   getAnswer",
   RequestNamespace="urn:EightBall",
   ResponseNamespace="urn:EightBall",
   Use=System.Web.Services.Description.
   SoapBindingUse.Encoded, ParameterStyle=
   System.Web.Services.Protocols.
   SoapParameterStyle.Wrapped)]
public System.String getAnswer() {
   object[] args = {"Will I win the lottery?"};
   object[] results = this.Invoke("getAnswer",
   args);
   return ((string)(results[0])); }
```

ibm.com/developerWorks/
webservices

o

Perl

- This uses Paul Kulchenko's SOAP::Lite toolkit (**www.soaplite.com**):

```
use SOAP::Lite;
print ("PERL SOAP client:\n   ");
print SOAP::Lite
  -> proxy('http://localhost…')
  -> uri('urn:EightBall')
  -> getAnswer()
  -> result;
```

Python

- This uses Cayce Ullman and Brian Matthews' SOAP.py toolkit (see **sourceforge.net/projects/pywebsvcs/**):

```
#!/usr/bin/env python
import SOAP
print "Python SOAP client:\n  "
server = SOAP.SOAPProxy(
  "http://localhost:8070…",
  namespace="urn:EightBall")
print server.getAnswer()
```

VisualBasic

- Built with Simon Fell's PocketSOAP (**pocketsoap.com**):

```
dim env
set env = CreateObject
 ("pocketSOAP.envelope.2")
dim http
set http = CreateObject
 ("pocketSOAP.HTTPTransport.2")
http.SOAPAction = ""
http.Send "http://localhost...",
 env.serialize
env.parse http
wscript.echo "Answer from the Great Beyond:"
 & env.Parameters.Item(0).Value
```

C++

- Built with the gSOAP toolkit (**www.cs.fsu.edu/~engelen/soap.html**):

```
#include "soapH.h" // obtain the generated proxy
int main()
{
   struct soap soap; // gSOAP runtime environment
   char* question = "Will I win the lottery?";
   char* answer;
   soap init(&soap); // initialize runtime environment
   if (soap call ns1 getAnswer(&soap,
       "http://localhost:8070/axis/services",
       "urn:EightBall", question, answer) == SOAP OK)
     cout << "The Eight Ball says: " << quote;
   else // an error occurred
     soap print fault(&soap, stderr); // display fault msg
}
```

o

The `SOAPAction` field

- If you're sending SOAP messages across an HTTP connection, you're required to use the **`SOAPAction`** field in the HTTP header.
- The SOAP spec doesn't define how this field should be used.
  - It could be used by a firewall to filter out SOAP requests, for example.

o

The `SOAPAction` field

- Here's how you set **`SOAPAction`** in an Axis client:

```
call.setSOAPActionURI
  ("EightBall#getAnswer");
```

- Other toolkits have similar methods.

o

Visit `ws-i.org`!

- IBM, Microsoft, and several other companies recently announced the Web Services Interoperability organization, **`ws-i.org`**.
- Most of the industry has joined the group...
- This group will provide test suites and tools to ensure interoperability.

o

Interoperability

- We'll look at SOAP services hosted by Axis, SOAP::Lite, and .NET.
- Next, we'll look at Java, Perl, and C# clients written for all three services.
- There are still some gotchas for interoperability...
  - ...but try doing the same thing with CORBA or DCOM.

o

The big picture

- We take a couple of simple Java classes and make them Web services without changing the source code.
- Once we deploy those classes as Web services, we can access those services from any language and any platform we want.
- We also get support for tools that can generate clients and proxies.

ibm.com/developerWorks/
webservices

o

The big picture

- **It’s not often that you get this much power and flexibility without having to write or rewrite a lot of code.**

o

Let’s get started!

The revolution is here...

o

You're all pioneers

- If you've even heard of these technologies, you're ahead of most of your colleagues.
  - Do we know how this will change the Web?
  - Do we have all the business models worked out?
  - Do we know how the software and services industries will change?
- No, no, and no. **The rules have changed; go out there and change the world!**

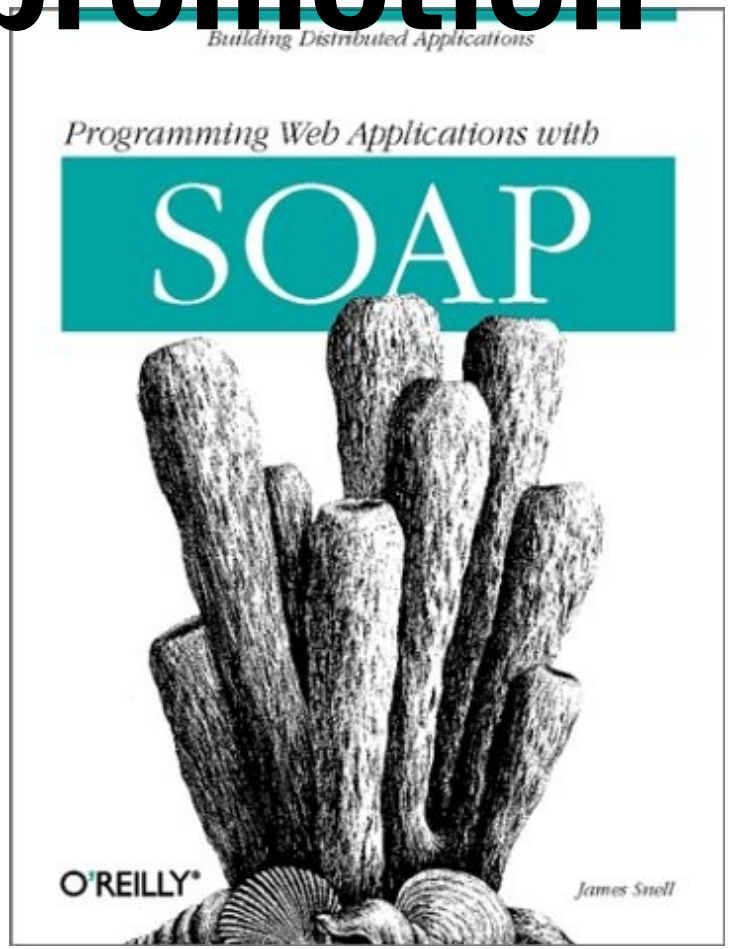
o

Start now!

"Businesses that ignore its potential or decide to sit out its early stages will find themselves outpaced by rivals that take advantage of Web services to improve their agility and even to transform themselves into new kinds of enterprises."

— David Smith, VP and
Research Director for Gartner

Shameless self-promotion



ISBN 0596000952

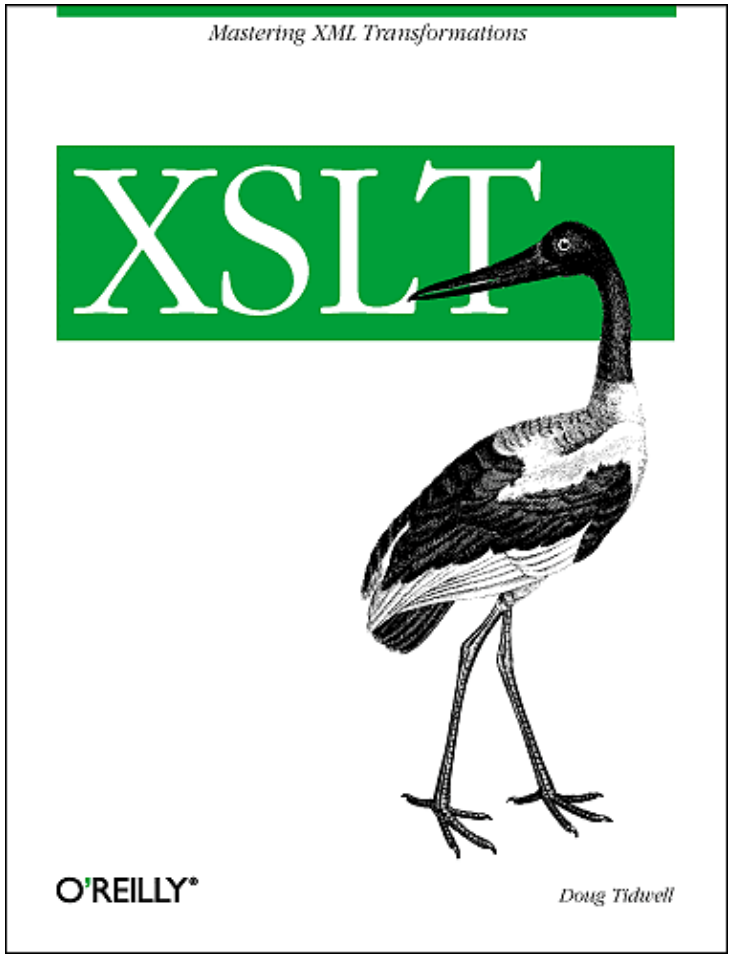
Co-Written with James Snell (**`soap-wrc.org`**) and Paul Kulchenko.

Available at amazon.com.

ibm.com/developerWorks/webservices

o

Shameless – Part 2



ISBN 0596000537

Order your copy at amazon.com today!

Makes a great holiday gift!

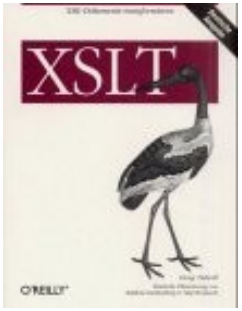
Show your loved ones

how much you care
by showing them the power of XSLT!

ibm.com/developerWorks/webservices

o

Schamlos – Teil drei



Wer XML bereits kennt und ein Problem schnell lösen möchte oder noch nach einem Problem sucht, für den ist XSLT genau richtig. Doug Tidwell richtet sich an XML-erfahrene Entwickler, die schnell in die komplexen Sphären von XSL eintauchen möchten und Antworten auf ihre eigenen Problemstellungen suchen. – Norbert Hartl

See www.oreilly.de/catalog/xsltger.

ibm.com/developerWorks/webservices

These slides and samples...

- ...are available at

  www.ibm.com/developerworks/speakers/dtidwell.

ibm.com/developerWorks/webservices

o

Thanks for coming!

Doug Tidwell
IBM developer**Works**
dtidwell@us.ibm.com