Building Web Services with Axis

Venkat Subramaniam
venkats@agiledeveloper.com

Presentation and examples can be downloaded from
http://www.agiledeveloper.com/download.aspx

Abstract

Abstract Web services promise greater interoperability across application written in different languages and running on different platforms. Axis is the Apache open source implementation that provides an infrastructure to develop and deploy web services. This talk will start with a very short introduction of web services and show you how to create web services using Axis, how you can exchange binary data, and help you understand the object lifetime issues. The speaker will also make some recommendations as to how to build effective web services. Several coding examples will be presented.

Speaker Dr. Venkat Subramaniam (venkats@agiledeveloper.com), founder of Agile Developer, Inc., has trained and mentored more than 3000 software developers in the US and Europe. Venkat is also an adjunct professor at the University of Houston and teaches the Professional Software Developer Series at Rice University's Technology Education Center. He speaks regularly at software development conferences.

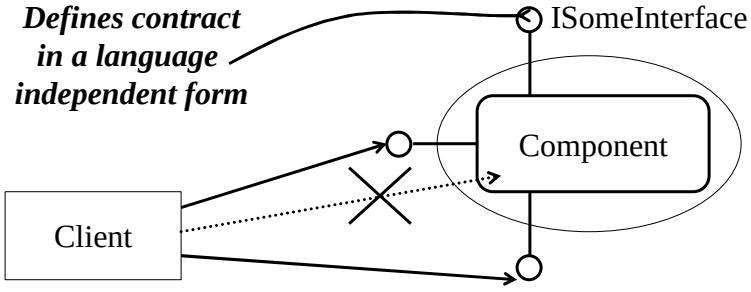
Examples Any page with a  has an example attached
Download from http://www.agiledeveloper.com/download.aspx

Building Web Services With Axis

- **State of Distributed Computing**
- Understanding SOAP
- Understanding WSDL
- Web Service with Axis
- .NET Client
- Attachments: SwA and DIME
- Object Lifetime
- How to build a web service?
- Conclusion

Distributed Computing With Interfaces

- DCOM, CORBA, RMI architectures

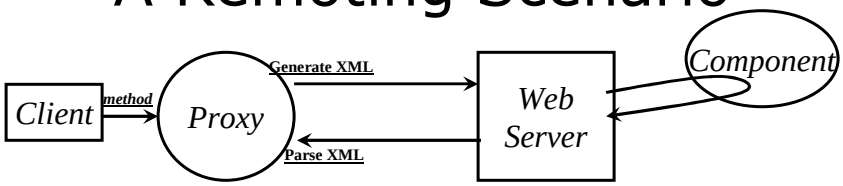


Encapsulation
Flexibility to change implementation at will
Polymorphism and substitutability
Language interoperability
Platform interoperability(?)
Process/processor independence

Distributed Computing & XML

- Problems with DCOM & CORBA
  - Interfaces based clients bind to the interface IDL
  - DCOM works only among select platforms
    - Predominantly on Windows NT, 2000, etc.
  - Client side ORB is required in case of CORBA
  - **These technologies are for intranet and not internet**
    - firewall will block the requests from a truly remote client!
- XML
  - provides cross platform transfer mechanism
  - has multi-language / multi-vendor support
  - So why not use XML to transport method invocation rather than simply data?

A Remoting Scenario



- The Payload
  - XML document could simply present method names
    - as elements
  - data may be carried by sub-elements
- The receiving system could
  - parse the XML document
  - Generate an XML response and send it back to the client
- The client waits for response, receives & parses it

Building Web Services With Axis

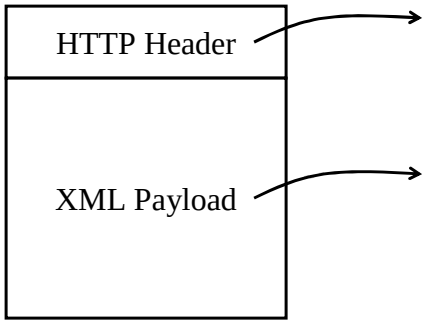
- State of Distributed Computing
- **Understanding SOAP**
- Understanding WSDL
- Web Service with Axis
- .NET Client
- Attachments: SwA and DIME
- Object Lifetime
- How to build a web service?
- Conclusion

SOAP

- A well-formed XML may be used as a wire protocol
- XML documents can be easily transported to remote systems
  - Most convenient to use HTTP (walks through firewall)
  - Other mechanisms like SMTP
- Mechanism is addressed by SOAP
  - **Simple Object Access Protocol** (SOAP)
- SOAP relies on the XML schema definition and elements
  - It uses the schema definition
  - The elements are used to communicate information
  - attributes are used to communicate SOAP specific meta-information

SOAP's approach

- To use HTTP (or other mechanisms) for transport
- Use XML for data encoding/information exchange
- Keep it simple and extensible



HTTP Method information

•Well-formed XML

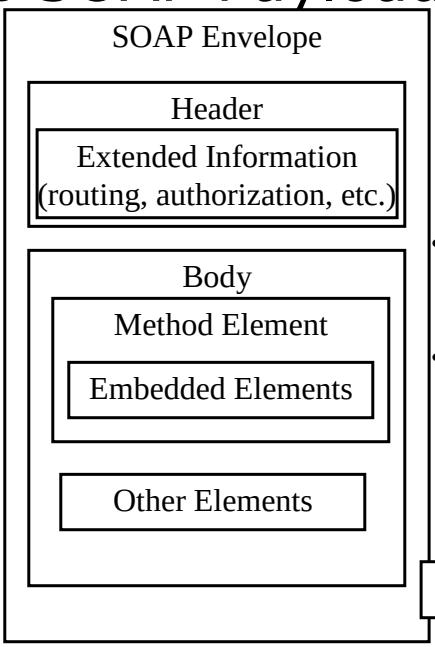
•Strong data types enforced with Schema

•Details on method invocation

Why SOAP?

- Why not use existing transport mechanisms to achieve greater distribution across platforms?
- SOAP simply uses two existing and very popular standards
  - HTTP for transport protocol
  - XML for data interchange
- SOAP simply provides a interoperable wire protocol
- All you need is a HTTP server and an XML parse
  - Almost

Basic SOAP Payload Structure



- How is body element formatted? *RPC or Document*
- How are parameters formatted? *Encoded or literal*

Message Encoding

- RPC Encoded
  - Defined by SOAP specification (Section 5, 7)
  - All parameters within single element with service method name
  - Parameter element named after parameter
  - Default in Axis
- Document Literal
  - Defined by XML schema for each parameter
  - More flexible
  - Default in .NET

RPC Encoded vs. Doc-literal

- Consider int add(int a, int b)

```
<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>
<add xmlns="http://www.agiledeveloper.com">
        <a>4</a>
        <b>8</b>
</add>
```

Doc-literal

```
<soap:Envelope xmlns:xsi=…  xmlns:xsd=…
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://www.agiledeveloper.com"
xmlns:types="http://www.agiledeveloper.com/encodedTypes"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<tns:add>
        <a xsi:type="xsd:int">4</a>
        <b xsi:type="xsd:int">8</b>
</tns:add>…
```

RPC-Encoded

RPC Encoded vs. Doc-literal…

- Consider bool shipTo(Address mailingAddress)

```
<soap:Body>
<shipTo xmlns="http://www.agiledeveloper.com">
<mailingAddress>
        <street>101 Main St.</street>
        <city>Sugar Land</city>
        <state>Texas</state>
        <zip>77478</zip>
</mailingAddress>
```

Doc-literal

```
<soap:Body soap:encodingStyle=…>
<tns:shipTo>
        <mailingAddress href="#a1" />
</tns:shipTo>
<types:Address id="a1" xsi:type="types:Address">
        <street xsi:type="xsd:string">101 Main St.</street>
        <city xsi:type="xsd:string">Sugar Land</city>
        <state xsi:type="xsd:string">Texas</state>…
```

RPC-Encoded

Web Service Infrastructure

- Web Service Wire Format
  - Protocol to allow open communication between various entities independent of platform
  - SOAP
- Web Service Description
  - Allows for finding the details of web services, its methods, arguments, types, etc
  - Web Service Description Language (WSDL)
- Web Service Discovery
  - Allows for a web service to make its presence known and for a client to find it
  - SOAP Discovery (DISCO)
  - Universal Description, Discovery and Integration (UDDI)

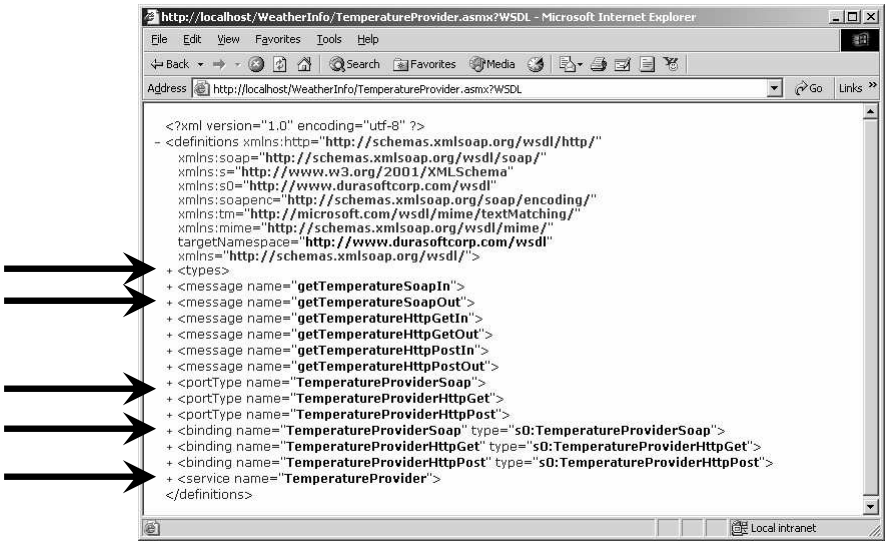
Building Web Services With Axis

- State of Distributed Computing
- Understanding SOAP
- **Understanding WSDL**
- Web Service with Axis
- .NET Client
- Attachments: SwA and DIME
- Object Lifetime
- How to build a web service?
- Conclusion

WSDL

- Web Service Description Language
- Describes
  - the web service
  - the methods published by the web service
  - the arguments and results of methods
- Used to create the client proxy
- Description in the form of XML Schema
- Automatically generated using reflected metadata
  - kept in synch with the code changes
- You may view it by visiting your web service with a ?WSDL request
  - **http://localhost/WeatherInfo/TemperatureProvider.asmx?WSDL**

The WSDL contents



Quiz Time

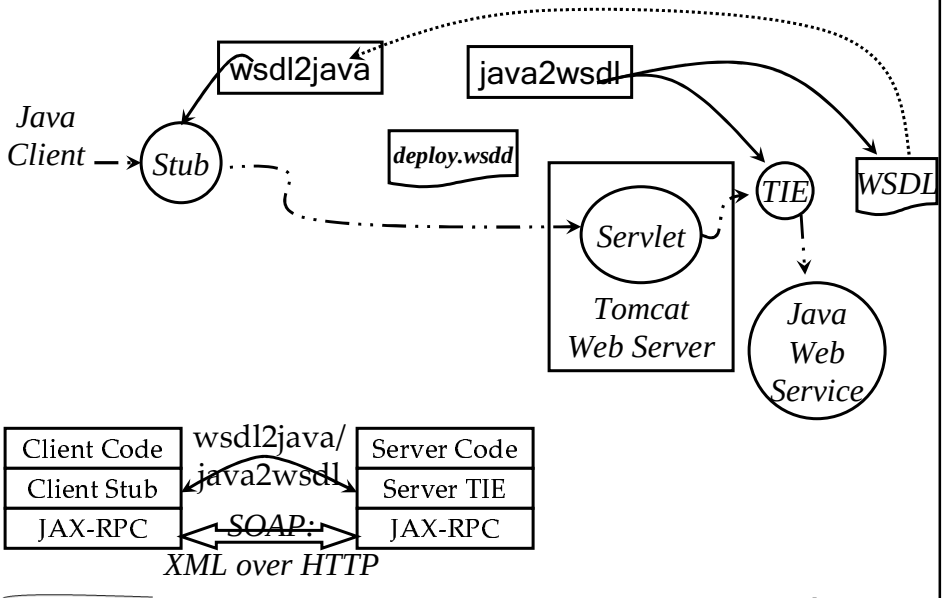
Building Web Services With Axis

- State of Distributed Computing
- Understanding SOAP
- Understanding WSDL
- **Web Service with Axis**
- .NET Client
- Attachments: SwA and DIME
- Object Lifetime
- How to build a web service?
- Conclusion

Apache's Axis

- Remember the Apache SOAP toolkit?
  - IBM's SOAP4J
- Axis is simply a SOAP Engine
- Promotes easy "drop in" deployment
  - Must easier than JWSDP, but limited use
- Deployed using a deployment descriptor
  - Allows flexibility, can provide Handlers (interceptors) for your message

Axis Web Service and Clients



Drop-in Deployment

- jws extension to your java files
- Simply copy to webapps\axis directory
- Your code is compiled on the fly (like JSP)

- Limitations
  - All public methods are exposed as web service (unlike in .NET where you can selectively declare web methods)
  - Intended for simple services only
  - No packages allowed
  - **Production quality services should not use this**

Custom Deployment (WSDD)

- Custom deployment allows you to
  - Configure the service
  - Push legacy code as web service (even if you do not have source code)
- Web Service Deployment Descriptor

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
        xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
        <service name="Info" provider="java:RPC">
                <parameter name="className"
                        value="com.agiledeveloper.ws.InformationProvider"/>
                <parameter name="allowedMethods"
                        value="getRecord, getFullName"/>
        </service>
</deployment>
```

May be *

Specifies handler or provider:
org.apache.axis.providers.java.RPCProvider

Descriptor separate from code

- Keeping the descriptor separate from code has problems
- This was the approach in EJB as well
- Wouldn't it be nice to keep it in synch with the code – like in .NET

- *"Note for the future : the Axis team, and the Java SOAP community at large, are thinking about ways to be able to embed this sort of metadata into your source files if desired - stay tuned"*

Building Web Services With Axis

- State of Distributed Computing
- Understanding SOAP
- Understanding WSDL
- Web Service with Axis
- **.NET Client**
- Attachments: SwA and DIME
- Object Lifetime
- How to build a web service?
- Conclusion

.NET Client

- Wsdl.exe reads WSDL to generate proxy
- Figures out RPC-Encoded or Doc-Literal
  - .NET by default uses Doc-Literal
- Invoke method on proxy as if making a local method call

Building Web Services With Axis

- State of Distributed Computing
- Understanding SOAP
- Understanding WSDL
- Web Service with Axis
- .NET Client
- **Attachments: SwA and DIME**
- Object Lifetime
- How to build a web service?
- Conclusion

Attachments

- Sending data using XML message is limiting
  - Limited to character content
  - Limited to well-formed syntax
- What about binary data, images, etc?
- You may encode binary data as base64 char
  - Not efficient ☹
- Need to send efficiently
  - SOAP with Attachment (Swa) based on MIME
  - Direct Internet Message Exchange (Microsoft)

SwA

- Pretty much based on MIME
- Attachment follow SOAP Envelop
- Content-Type: Multipart/Related MIME header appears as part of HTTP header
- MIME headers can't appear as part of HTTP header (however, that's allowed in case of SMPT headers)
- Supported by Sun's JWSDP
  - Through SOAP with Attachment API for Java (SAAJ)
- Originally was developed with Microsoft
- Microsoft now favors DIME, however

SwA...

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml; …
Content-Description: This is the optional message description.
--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: …
<?xml version='1.0' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
…
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
--MIME_boundary
Content-Type: image/gif
Content-Transfer-Encoding: binary
Content-ID: …
...binary image...
--MIME_boundary--
```

DIME

- A message is a series of records
- Each record has fixed-length (128 bits) header that determines length (up to 4GB – multiples of 32 bit) and type of record
- Design for efficient transmission over HTTP and TCP as well
- First record is SOAP message, rest are attachments
- Attachments have ID – refer to these in SOAP message

SwA vs. DIME

- SwA is basically MIME
- MIME is great for sender
- Receiver suffers from performance
- Have to scan for boundary string to figure out next attachments
- Especially inefficient if receiver interested only in select attachments
- DIME provide for more efficiency by communicating the length of each record being transmitted

Building Web Services With Axis

- State of Distributed Computing
- Understanding SOAP
- Understanding WSDL
- Web Service with Axis
- .NET Client
- Attachments: SwA and DIME
- **Object Lifetime**
- How to build a web service?
- Conclusion

Object Lifetime

- Different implementations deal with object life time differently
  - In JWSDP, one object shared by all requests
  - In .NET, each request (even from same client) gets different object
  - Axis gives the most flexibility
    - You may configure using a parameter name="scope" whose value is Application, Session or Request
    - By default, behaves like .NET does

Building Web Services With Axis

- State of Distributed Computing
- Understanding SOAP
- Understanding WSDL
- Web Service with Axis
- .NET Client
- Attachments: SwA and DIME
- Object Lifetime
- **How to build a web service?**
- Conclusion

Some Guidelines

- Do not think WS as remote method call
- Keep the interaction coarse grain
- Do not put business logic in your WS
- Do not call WS from within a web app
  - Unless it is a third party WS you are calling
- Do not use WS if the two end points of your application are fully under control
- Use your judgment while using the above guidelines
  - Never say never, right?

Quiz Time

Building Web Services With Axis

- State of Distributed Computing
- Understanding SOAP
- Understanding WSDL
- Web Service with Axis
- .NET Client
- Attachments: SwA and DIME
- Object Lifetime
- How to build a web service?
- **Conclusion**

Conclusion

- Web Services has potential
- Beginning to gain serious considerations

- Vendor support varies
- Security is one of the big concerns
- Still issues related to security, attachments, etc. not fully resolved
- Still no one standard that has established
- Some early adopter implementations out there – we have seen some of them here
- Will be a while before the dust settles and the path is clear