Advances in Web Services

Venkat Subramaniam
venkats@agiledeveloper.com

October 2003

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

Abstract

Abstract Web Services is gaining a lot of popularity. Several organizations are beginning to implement serious systems and components using web services. Web services promise greater interoperability across application written in different languages and running on different platforms. However, much concern exists over the practicality of the solution, from the point of view of security, transactions, scalability, performance and infrastructure. This presentation will first introduce the audience to implementation of web services and present details on advances in the areas mentioned. Several working examples will be presented to illustrate the concepts.

Speaker Dr. Venkat Subramaniam, founder of Agile Developer, Inc., has trained and mentored more than 2,500 software developers around the world. He has significant experience in architecture, design, and development of distributed object systems. Venkat is an adjunct professor at the University of Houston and teaches the Professional Software Developer Series at Rice University's Technology Education Center.

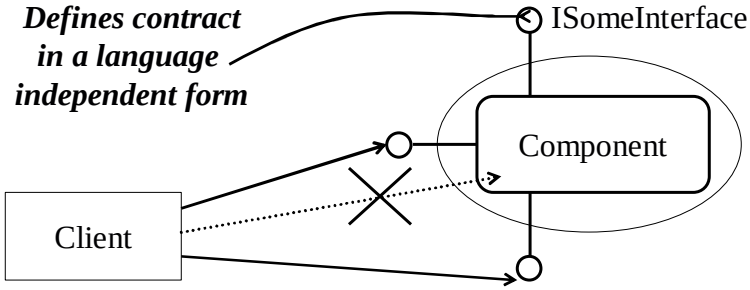
Examples Any page with a  has an example attached
Download from http://www.agiledeveloper.com/download.aspx

Advances in Web Services

- **State of Distributed Computing**
- Application of XML
- SOAP & WSDL
- Using JWSDP
- Using Axis
- Web Service in .NET
- WS-I Basic Profile
- Security and WS-Security
- Attachments: SwA and DIME
- WS-Routing and WS-Referral
- Specifications under consideration
- 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?

Advances in Web Services

- State of Distributed Computing
- **Application of XML**
- SOAP & WSDL
- Using JWSDP
- Using Axis
- Web Service in .NET
- WS-I Basic Profile
- Security and WS-Security
- Attachments: SwA and DIME
- WS-Routing and WS-Referral
- Specifications under consideration
- Conclusion

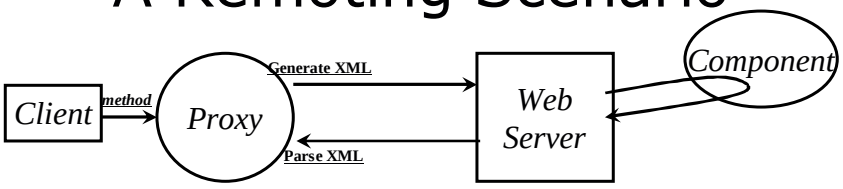
Why XML?

- XML is about extensibility and flexibility
- tags describe and surround the data
- Example:

```
<?xml version = "1.0" ?>
<equipment>
   <pump>
        <name> p01 </name>
        <pressure units="psi">
                32.23 </pressure>
   </pump>
   <pump>
        <name> p02 </name>
        <pressure units="psi">
                22.887 </pressure>
   </pump>
</equipment>
```

- Open, extensible
- Platform independent
- Self describing data
  - Data Exchange
- Supports query and discovery of data
- ***Dynamic Data Exchange***

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

Advances in Web Services

- State of Distributed Computing
- Application of XML
- **SOAP & WSDL**
- Using JWSDP
- Using Axis
- Web Service in .NET
- WS-I Basic Profile
- Security and WS-Security
- Attachments: SwA and DIME
- WS-Routing and WS-Referral
- Specifications under consideration
- Conclusion

Sounds good, how do we do this?

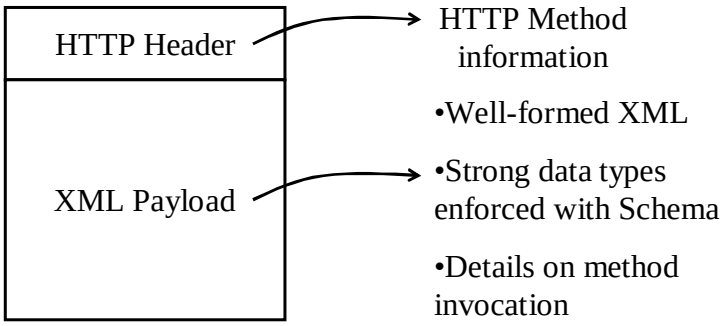
- Each one inventing there own way to do this
  - Leads to duplication of effort
  - Re-inventing the wheel results in higher cost
  - What about efficiency?
- Standardization is essential

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

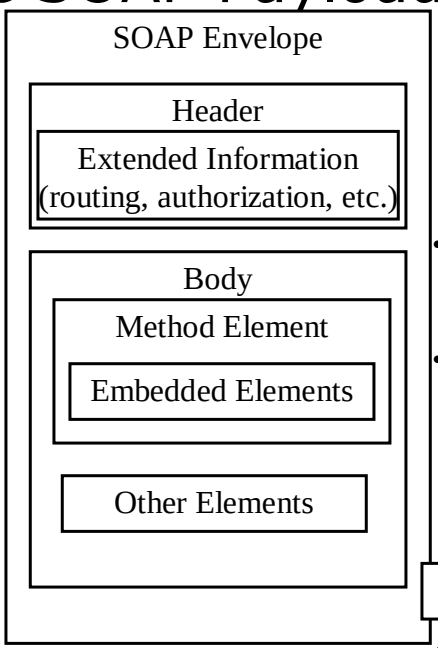
- To use HTTP (or other mechanisms) for transport
- Use XML for data encoding/information exchange
- Keep it simple and extensible



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">                Doc-literal
        <a>4</a>
        <b>8</b>
</add>
```

```
<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>                RPC-Encoded
        <b xsi:type="xsd:int">8</b>
</tns:add>…
```

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>
```

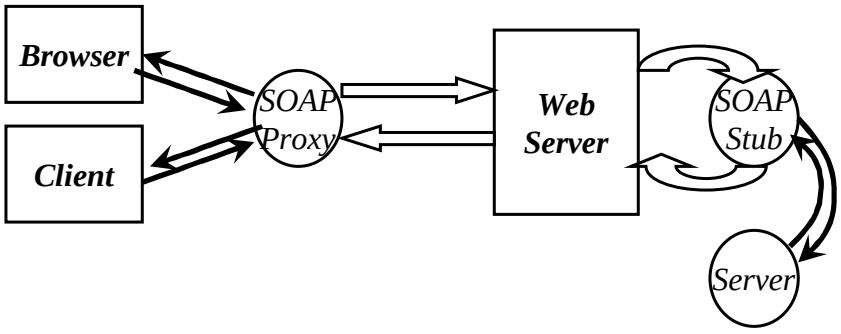
Doc-literal

```
<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

SOAP Applied

- Web Services is simply an application of SOAP



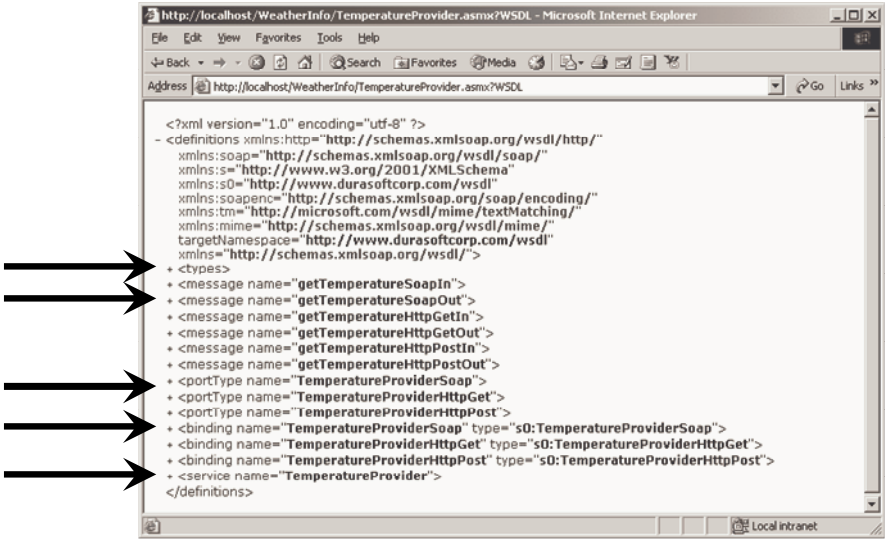
Web Service Infrastructure

- 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)

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

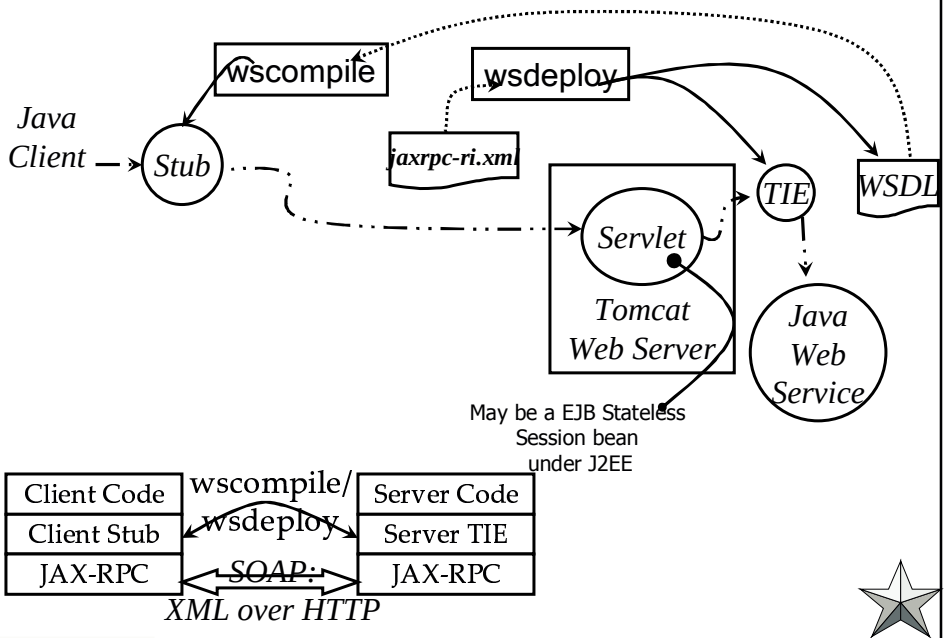
Advances in Web Services

- State of Distributed Computing
- Application of XML
- SOAP & WSDL
- **Using JWSDP**
- Using Axis
- Web Service in .NET
- WS-I Basic Profile
- Security and WS-Security
- Attachments: SwA and DIME
- WS-Routing and WS-Referral
- Specifications under consideration
- Conclusion

JWSDP – Oh Well

- JWSDP is Sun specification
  - + Reference Implementation
- Has taken a RMI based interface approach
- Provides an implementation, *but*
  - Not as easy to use
  - Not much drive behind overall adaptation
- I would look at Apache Axis instead
- Examples provided in this presentation
  - will be handled pretty lightly
  - so we can focus on Axis and .NET
    - two leading implementations

Java Web Service and Clients



JWSDP 1.2 includes

- JSF 1.0 EA4
- XML and Web Services Security 1.0 EA
- JAXB 1.0.1
- SAAJ 1.2 EA
- JAXP 1.2.3
- JAXR 1.0.4
- JAX-RPC 1.1 EA
- JSTL 1.1 EA
- Apache Tomcat 5
- Registry Server 1.0_05

Writing a Java Web Service

- Methods of a Web Service have to be part of a remote interface
- Write a class to implement that interface
- Compile the Service Interface and Class
- Write a configuration file (jaxrpc-ri.xml) that indicates
  - the interface name
  - the class name
  - the service name
  - the target namespace
- Write a web.xml that ties the servlet and the web service
- Run mapping tool wsdeploy to generate server side *tie* class
- Deploy the web service

Information Provider Service

```
package com.agiledeveloper.ws;
public interface InformationProviderIF
      extends java.rmi.Remote {
   public String getRecord(String city)
      throws java.rmi.RemoteException; …
}
```

```
package com.agiledeveloper.ws;
public class InformationProviderImpl
    implements InformationProviderIF,
    javax.xml.rpc.server.ServiceLifecycle {
      public void init(Object context){}
      public void destroy(){}
      public String getRecord(String city)
      {
      // Fetch the information from a database, etc.
      // do any processing and return the result…
      }
}
```

Preparing the Service

jaxrpc-ri.xml

```
<webServices …
  targetNamespaceBase="http://www.agiledevelper.com/wsdl"
  typeNamespaceBase="http://www.agiledevevloper.com/types“>
  <endpoint
    name=“InformationProvider" …
    interface="com.agiledevelopper.ws.InformationProviderIF"
    implementation=
      "com.agiledeveloper.ws.InformationProviderImpl"/>
  <endpointMapping endpointName="InformationProvider"
      urlPattern="/InforamtionProvider"/>
</webServices>
```

```
javac -d InformationProvider\WEB-INF\classes *.java
copy jaxrpc-ri.xml InforamtionProvider\WEB-INF
copy web.xml InformationProvider\WEB-INF
jar –cf Info_predeploy.war WEB-INF (from InformationProvider)
wsdeploy –keep … Info.war Info_predeploy.war
```


Deploying the Service

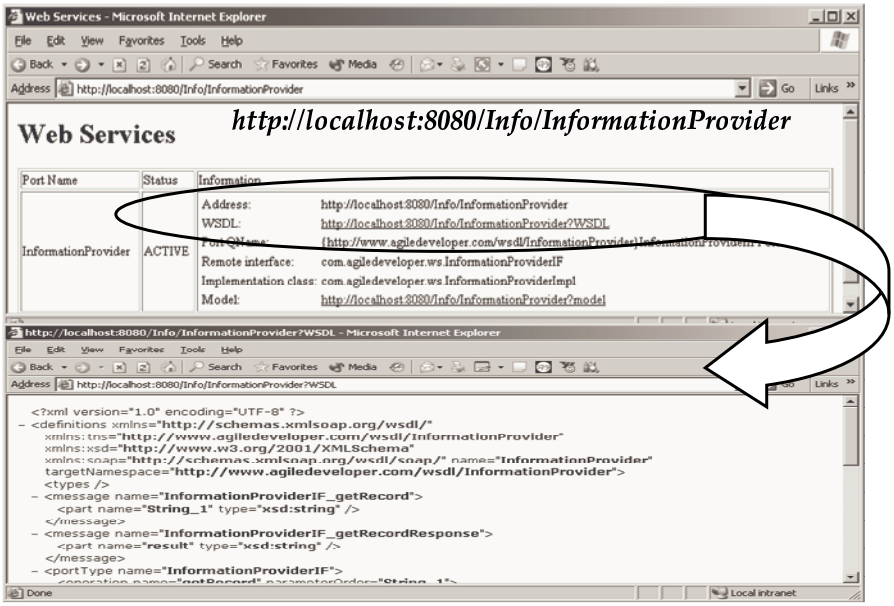
InformationProvider/WEB-INF/web.xml

```
…
<web-app>
  <display-name>
      An Application that provide People Information
  </display-name>
  <description>
      … return information about a person
  </description>
  <session-config>
    <session-timeout>60</session-timeout>
  </session-config>
</web-app>
```

wsdeploy fills in the rest of the details.
View the generated web.xml in the Info.war

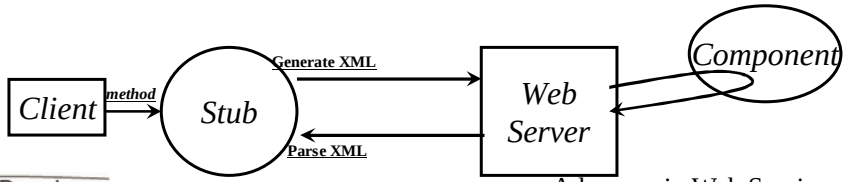
Drop the Info.war into Tomcat's jwsdp-1.2/WebApps

Checking the deployment



Traditional clients

- For non-web based client, you can create proxy
- The web client proxy receives a request, generates XML request & sends it using SOAP
- Receives response, parses it and hands it over to client!
- You can generate a web service proxy using the WSDL compiler.
- You provide a reference to service description using which proxy can be generated



Traditional Java client

Client.java main

```
InformationProviderIF service =
     new InformationProvider_Impl()
          .getInformationProviderIFPort();

String info =
          service.getRecord(personsLastName);
```

config.xml

```
<configuration
  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
  <wsdl
  location=
      "http://localhost:8080/Info/InformationProvider?WSDL"
  packageName="com.durasoftcorp.ws" />
</configuration>
```

```
wscompile –gen:client -keep –s keepDirName -d .  config.xml
javac Client.java
```

Generates stub from the WSDL

C# Client

Client.cs Main

```
Info.InformationProvider service =
      new Info.InformationProvider();


String info = service.getRecord(lastName);
```

```
wsdl /n:Info http://localhost:8080/Info/InformationProvider?WSDL

csc /out:Client.exe Client.cs InformationProvider.cs
```

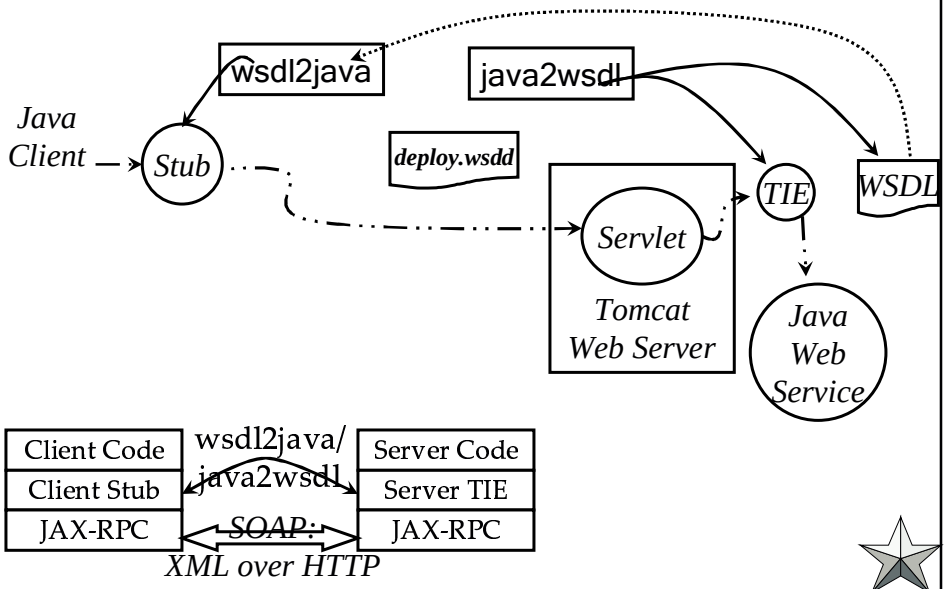
Advances in Web Services

- State of Distributed Computing
- Application of XML
- SOAP & WSDL
- Using JWSDP
- **Using Axis**
- Web Service in .NET
- WS-I Basic Profile
- Security and WS-Security
- Attachments: SwA and DIME
- WS-Routing and WS-Referral
- Specifications under consideration
- 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

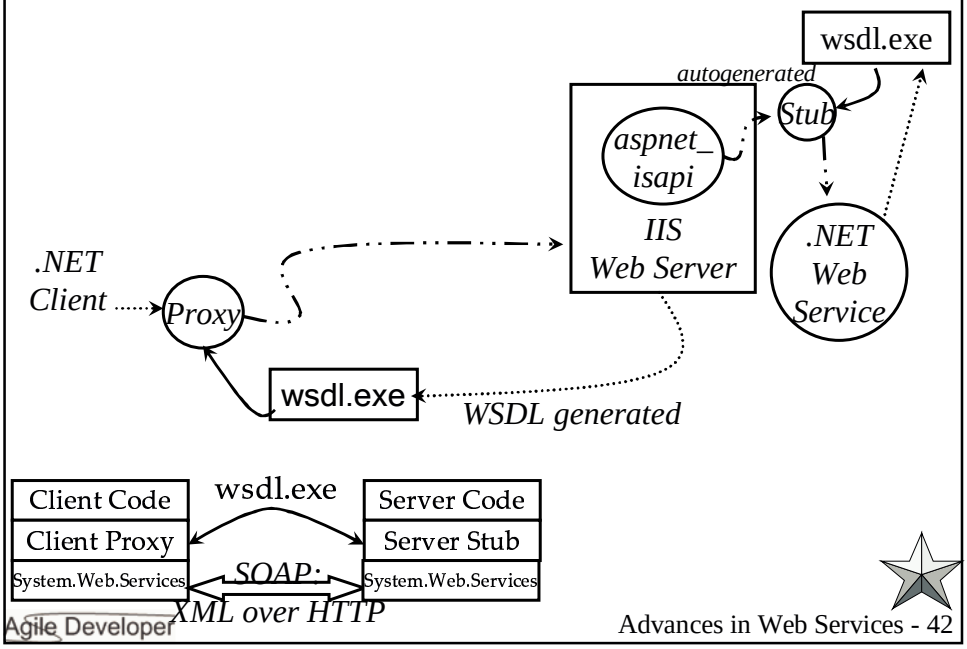
- Keeping the descriptor separate from code has problems
- This was the approach in EJB as well
- Wouldn't it be nice to keep it in synch with the code – like in .NET

- *"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"*

Advances in Web Services

- State of Distributed Computing
- Application of XML
- SOAP & WSDL
- Using JWSDP
- Using Axis
- **Web Service in .NET**
- WS-I Basic Profile
- Security and WS-Security
- Attachments: SwA and DIME
- WS-Routing and WS-Referral
- Specifications under consideration
- Conclusion

.NET Web Service and Clients



Writing a .NET Web Service

- Methods of a Web Service a marked using an attribute [WebMethod] – no separate interface
- Write a class to implement your Web Service methods with an extension asmx
  - No configuration files needed
  - No need to use any server side tools – done automatically
  - No deployment files needed
- Deploy the web service by copying to IIS directory

Writing a .NET Web Service

```
<%@ WebService Language="C#"
      Class= “InformationProvider" %>
using System;
using System.Web.Services;
[WebService(Namespace="http://www.agiledeveloper.com/ws")]
public class InformationProvider
{
    [WebMethod]
    public String getRecord(string lastName)
    {
      …
    }
    …
}
```

Preparing the .NET Web Service

- Nothing to do in this step, the server side stubs are generated automatically
- Optionally, you could use the mapping tool to generate the server side stubs – no need

Deploying .NET Web Service

Drop the InformationProvider.asmx file into inetpub\wwwroot

Developing with Visual Studio

- While creating a asmx page directly is easy, it is not the way to go for serious applications
- Use of an IDE helps
- Creating a project using studio helps us use the code-behind concept fairly easily

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

Quiz Time

Advances in Web Services

- State of Distributed Computing
- Application of XML
- SOAP & WSDL
- Using JWSDP
- Using Axis
- Web Service in .NET
- **WS-I Basic Profile**
- Security and WS-Security
- Attachments: SwA and DIME
- WS-Routing and WS-Referral
- Specifications under consideration
- Conclusion

Interoperability

- Strength of web services relies on interoperability
- Perils:
  - Ambiguity in standards interpretation
  - Differences among specifications
  - Interaction between specs and vendor differences
- RPC-encoded vs. Doc-literal?
- Is the document in UTF-8, UTF-16, etc…?
- Security?
- Attachments?
- Transactions?
- How to keep it manageable?
- Web Services Interoperability Org (WS-I)

WS-I

- Industry commitment
- Not interested in creating standard
- Specifications for interoperability
  - Best practices recommendations
  - Subset to promote interoperability
  - Example:

R0001 *An INSTANCE MUST be described by a WSDL 1.1 service description, by a UDDI binding template, or both.*

- Basic Profile 1.0a (08/2003)

WS-I BP 1.0a Referenced Specifications

- Simple Object Access Protocol (SOAP) 1.1
- Extensible Markup Language (XML) 1.0 (Second Edition)
- RFC2616: Hypertext Transfer Protocol -- HTTP/1.1
- RFC2965: HTTP State Management Mechanism
- Web Services Description Language (WSDL) 1.1
- XML Schema Part 1: Structures
- XML Schema Part 2: Datatypes
- UDDI Version 2.04 API Specification, Dated 19 July 2002
- UDDI Version 2.03 Data Structure Reference, Dated 19 July 2002
- UDDI Version 2 XML Schema
- RFC2818: HTTP Over TLS (Transport Layer Security)
- RFC2246: The TLS Protocol Version 1.0
- The SSL Protocol Version 3.0
- RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile

WS-I BP 1.0a Conformance Claim

- WSDL and Message may hold claims of conformance to WS-I
- This must be carried in the SOAP header (for messages)
- May contain conformance to more than one profile
- …

WS-I BP 1.0a XML Message Representation

- Fault element
  - ***Must not*** have children other than faultcode, faultstring, faultactor and detail
  - Child elements ***must*** be unqualified
  - Receiver ***must*** accept any number of children of detail element
- Message
  - ***must not*** contain DTD or PI
  - Soap:Envelope ***must not*** contain element after Body
- Receiver ***must not*** mandate the use of xsi:type attribute in a message… (encoded is out)
- Http
  - Message ***Should*** be sent using HTTP/1.1 (1.0 allowed)
  - ***Must*** use HTTP POST
- …

WS-I BP 1.0a Service Description

- ***Must*** only use *wsdl:import* statement to import another WSDL document
- *wsdl:import* statement ***must*** precede all other elements except *wsdl:documentation*
- *wsdl:type* statement ***must*** precede all other elements except above mentioned
- Extension ***should not*** contradict other requirements of profile
- wsdl:binding ***must use*** SOAP binding
  - No MIME, HTTP GET/POST binding
- wsdl:binding ***must use*** either *rpc-literal* or *document-literal* binding (*prohibits the use of Encoding*)
- …

Advances in Web Services

- State of Distributed Computing
- Application of XML
- SOAP & WSDL
- Using JWSDP
- Using Axis
- Web Service in .NET
- WS-I Basic Profile
- **Security and WS-Security**
- Attachments: SwA and DIME
- WS-Routing and WS-Referral
- Specifications under consideration
- Conclusion

WS-I Security Recommendation

- Adopts, but does not mandate use of, HTTP secured with either TLS 1.0 or SSL 3.0 (HTTPS)
- May use HTTPS, other counter measures or none at all

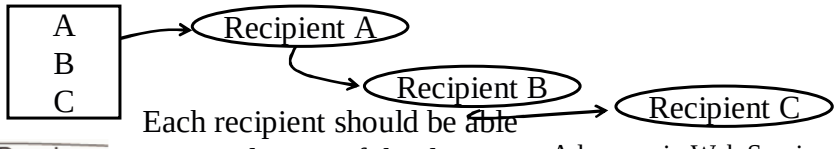
- An instance ***may*** require use of HTTPS
- And ***may*** do so with a requirement of mutual authentication

What do we mean by secure?

- Confidentiality
  - Keep data from being viewed by unauthorized person
- Integrity
  - Data/information has not been altered
- Authentication
  - Confirm that the sender and receiver are whom they claim to be
- Our discussion here will not go into SSL
  - Very popular, extensively used by web servers today
  - Does not address all issues

Why SSL/TLS may not be enough?

- These deal with encrypting/decrypting entire transmission
- What about being able to encrypt
  - Only the data being transmitted (entire XML)
  - Only select elements in XML message
  - Only the value of select attributes or elements
- Multi-recipient documents



Each recipient should be able to view only part of the document

Security: Things to Know

- IETF, W3C efforts
  - XML Canonicalization
    - Helps determine if two XML document (representations) are equivalent
  - XML Signature
    - Helps compute and verify the signature of XML documents and portions of document
  - XML Encryption
    - Helps encrypt full or part of XML message
  - WS-Security
    - Focused on web services security

XML Canonicalization

- Two XML documents may
  - Vary in structure (physical representation) but
  - May be considered equivalent in an application context
- A canonical form is a physical representation that accounts for permissible changes
- Two documents are considered equivalent if they have the same canonical form (canonical XML)
- XML Canonicalization is an algorithm that generates canonical XML for a document

XML Signature

- Useful to verify integrity of message
- Message (or part of it) is signed
- Upon receipt, the sign is verified
- The following is an example from the W3C org web site http://www.w3.org/TR/xmldsig-core/

XML Signature Example

```
<Signature Id="MyFirstSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
                <CanonicalizationMethod
                  Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
                <SignatureMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
                <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
                        <Transforms>
                                <Transform
                                Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
                        </Transforms>
                        <DigestMethod
                              Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
                </Reference>
        </SignedInfo>
        <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
        <KeyInfo>
                <KeyValue>
                        <DSAKeyValue>
                                <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
                        </DSAKeyValue>
                </KeyValue>
        </KeyInfo>
</Signature>
```

Information needed to replicate the signature

XML Encryption

- Consider the following fictitious document

```
<employee xmlns="http://www.agiledeveloper.com">
        <firstName>Venkat</firstName>
        <lastName<Subramaniam</lastName>
        <ssn>123-45-6789</ssn>
        <HireInfo>
                <HireDate>01-01-2003</HireDate>
                <Salary>3012.25</Salary>
        </HireInfo>
</employee>
```

- We may want to encrypt different parts

XML Encryption Granularity

- XML Element

```
<employee xmlns="http://www.agiledeveloper.com">
        <firstName>Venkat</firstName>
        <lastName<Subramaniam</lastName>
        <ssn>123-45-6789</ssn>
        <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
                xmlns='http://www.w3.org/2001/04/xmlenc#'>
                <CipherData>
                        <CipherValue>A12B34C56</CipherValue>
                </CipherData>
        </EncryptedData>
</employee>
```

XML Encryption Granularity…

- Element contents (child elements or character data)

```
<employee xmlns="http://www.agiledeveloper.com">
        <firstName>Venkat</firstName>
        <lastName<Subramaniam</lastName>
        <ssn>123-45-6789</ssn>
        <HireInfo>
                <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
                    Type='http://www.w3.org/2001/04/xmlenc#Content'>
                  <CipherData>
                      <CipherValue>A0123B89</CipherValue>
                  </CipherData>
                </EncryptedData>
        </HireInfo>
</employee>
```


XML Encryption Granularity…

- Entire XML Document

```
<?xml version='1.0'?>
  <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
   MimeType='text/xml'>
    <CipherData>
      <CipherValue>ABC123456</CipherValue>
    </CipherData>
  </EncryptedData>
```

WS-Security

- Leverages existing standards and specifications
  - Kerberos, X.509, XML Canonicalization, XML Encryption, XML Signature
- Defines SOAP header to carry security-related data
- Mechanism to transfer
  - User credential (user name, password) via UsernameToken element
  - Encryption, signing token via BinarySecurityToken

WS-Security Header

- A <Security> element caries information in the header
- Several such elements may be present
- Intended for different intermediaries – identified by actor attribute
- Only one security headers may contain on actor
- No two security header may have the same actor

Quiz Time

Advances in Web Services

- State of Distributed Computing
- Application of XML
- SOAP & WSDL
- Using JWSDP
- Using Axis
- Web Service in .NET
- WS-I Basic Profile
- Security and WS-Security
- **Attachments: SwA and DIME**
- WS-Routing and WS-Referral
- Specifications under consideration
- 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

Advances in Web Services

- State of Distributed Computing
- Application of XML
- SOAP & WSDL
- Using JWSDP
- Using Axis
- Web Service in .NET
- WS-I Basic Profile
- Security and WS-Security
- Attachments: SwA and DIME
- **WS-Routing and WS-Referral**
- Specifications under consideration
- Conclusion

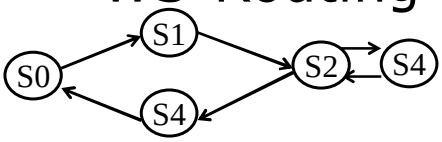
The path and intent

- SOAP lacks the ability to specify the routing needed to send an asynchronous message
  - Consider a service behind a firewall. How do you send a request to a proxy on the firewall which will then transmit your message to the actual machine(s)?
  - Also, how to instruct message to be routed though two or more parties (services)?

WS-Routing

- Stateless protocol that extends SOAP
- Specified ordered route or path from originator, through intermediaries to final destination, for
  - One way messages
  - Two way messages
  - Return paths may be specified as well
- Does not define reliable or secure messaging, however

WS-Routing Header



→ Forward message
← Reverse message

```
<SOAP:Header>
        <wsrp:path xmlns:wsrp="http://schemas.xmlsoap.org/rp/">
                <wsrp:action>What ever it is</wsrp:action>
                <wsrp:to>http://www.S3.com/EndPoint</wsrp:to>
                <wsrp:fwd>
                        <wsrp:via>http://www.S1.com</wsrp:via>
                        <wsrp:via>http://www.S2.com</wsrp:via>
                </wsrp:fwd>
                <wsrp:rev>
                        <wsrp:via>http://www.S2.com</wsrp:via>
                        <wsrp:via>http://www.S4.com</wsrp:via>
                </wsrp:rev>
                <wsrp:from>http://www.S0.com</wsrp:from>
                <wsrp:id>uuid:...</wsrp:id>
        </wsrp:path>
</SOAP:Header>
```

Route information may be dynamically generated by intermediary / programmatically as well

WS-Referral

- WS-Routing allows up to specify path
- How does a node know where to forward to?
- How about being able to dynamically configure the routes
- WS-Referral allows nodes to send and receive route configuration data
- These simply map a logical name to another name; typically of the form
  - For some name, go via some node, if some condition is true (in addition a desc and refid may be provided)

WS-Referral Header

```
<wsr:ref
xmlns:wsr="http://schemas.xmlsoap.org/ws/2001/10/referral">
 <wsr:for>
        <wsr:prefix>http://www.a.com</wsr:prefix>
 </wsr:for>
 <wsr:if>
        <ttl>600</ttl> <!-- time to live in seconds -->
 </wsr:if>
 <wsr:go>
        <wsr:via>http://www.b.com</wsr:via>
 </wsr:go>
 <wsr:refId>...</wsr:refId>
</wsr:ref>
```


Advances in Web Services

- State of Distributed Computing
- Application of XML
- SOAP & WSDL
- Using JWSDP
- Using Axis
- Web Service in .NET
- WS-I Basic Profile
- Security and WS-Security
- Attachments: SwA and DIME
- WS-Routing and WS-Referral
- **Specifications under consideration**
- Conclusion

Specifications under consideration

- These are under evaluation currently
- WS-Policy, WS-PolicyAssertions, WS-PolicyAttachment
  - General purpose model to describe policies for web services
- WS-SecurityPolicy
  - Addendum to WS-Security with policy assertions for WS-Policy which apply to security
- WS-Trust
  - Allows applications to construct trusted SOAP message exchange
- WS-SecureConversation
  - Extends WS-Security to allow constructing security context between applications
- WS-Addressing
- Provides standard format for specifying endpoint references and message information headers

Quiz Time

Advances in Web Services

- State of Distributed Computing
- Application of XML
- SOAP & WSDL
- Using JWSDP
- Using Axis
- Web Service in .NET
- WS-I Basic Profile
- Security and WS-Security
- Attachments: SwA and DIME
- WS-Routing and WS-Referral
- Specifications under consideration

Participating Companies

- WS-I
  - Several companies including IBM and Microsoft
  - Sun's indicates commitment to conform to WS-I recommendations
- WS-Security, WS-Routing, WS-Policy, etc.
  - BEA, IBM, Microsoft, SAP

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

Products Used in this Presentation

- **Sun's Java Web Services Developer Pack *1.2***: http://java.sun.com/webservices/

- **Apache's Axis *1.1***: http://ws.apache.org/axis/

- **Microsoft Visual Studio.NET *2003***: http://msdn.microsoft.com/vstudio/

- **Microsoft Web Services Enhancements *2.0***: http://msdn.microsoft.com/webservices/building/wse/default.aspx

References

1. http://www.w3.org/2002/ws/
2. http://www.w3.org/Signature/
3. http://www.w3.org/Encryption/2001/

4. http://www.ws-i.org/Profiles/Basic/2003-08/BasicProfile-1.0a.htm

5. http://ws.apache.org/axis/

6. http://java.sun.com/webservices/

7. http://msdn.microsoft.com/library/en-us/dnglobspec/html/wsspecsover.asp
9. http://msdn.microsoft.com/webservices/building/wse/default.aspx

10. http://www.agiledeveloper.com/download.aspx

Please fill out your evaluations!