Computer Science E-259

XML with Java

Lecture 11:
Web Services, SOAP 1.2, and WSDL 1.1

10 December 2007

David J. Malan

malan@post.harvard.edu

Last Time

XML Schema (Second Edition), Continued

- XML Schema (Second Edition), Continued

Last Time

Simple Types

```
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="year" type="xsd:integer"/>
<xsd:element name="lastmod" type="xsd:date"/>


<xsd:attribute name="country" type="xsd:string"/>
```

Last Time

Complex Types: Simple Content

```
<xsd:simpleType name="size">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="small" />
    <xsd:enumeration value="medium" />
    <xsd:enumeration value="large" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="jeans">
  <xsd:simpleContent>
    <xsd:extension base="size">
      <xsd:attribute name="sex">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="male" />
            <xsd:enumeration value="female" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Adapted from http://www.w3schools.com/schema/el_extension.asp.

Last Time

Complex Types: Element-Only Content

```
<name>
  <first>John</first>
  <last>Harvard</last>
</name>

<xsd:element name="name">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="first" type="xsd:string"/>
      <xsd:element name="last" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Last Time

Complex Types: Mixed Content

```
<letter>
Dear Mr.<name>John Smith</name>.
Your order <orderid>1032</orderid>
will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

```
<xsd:element name="letter">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="orderid" type="xsd:positiveInteger"/>
      <xsd:element name="shipdate" type="xsd:date"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Adapted from http://www.w3schools.com/schema/schema_complex_mixed.asp.

Last Time

Complex Types: Empty Content

```
<foo bar="baz" />

<xsd:element name="foo">
  <xsd:complexType>
    <xsd:attribute name="bar" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

This Time

Agenda

- Web Services
- RPCs
- SOAP 1.2
- WSDL 1.1
- Axis 1.4

Web Services

History

- SOAP in 1998
- XML-RPC
  - Dave Winer of UserLand Software, 1998
  - `http://www.xml-rpc.com/spec`
- SOAP since 1999
  - Based on XML-RPC
  - Version 0.9 released in late 1999 by Microsoft, DevelopMentor, and Dave Winer
  - Version 1.0 followed soon thereafter
  - Version 1.1 submitted as a W3C note May 2000 by DevelopMentor, IBM, Lotus, Microsoft, and UserLand Software
  - IBM and Microsoft release toolkits. IBM donates toolkit to Apache
  - Many vendors implement SOAP

Web Services

Architecture

- The web services architecture is an evolution of existing technologies
  - The Internet enables hosts to communicate and information to be published and retrieved
  - Distributed computing platforms allow programmatic components to communicate
  - XML closes the barriers between platforms and technologies

RPCs

Remote Procedure Calls

- Two pieces of code (client and server) talk over the network, generally using TCP/IP sockets
- The client need not be aware that it is not using a local class
  - Client-side *stub* implements the service interface and takes care of serializing/marshalling the arguments for transmission over the network. The stub makes the network call and hands the response back to the caller
- The remote object (server) need not be aware that it is not called by a local class
  - Server-side *skeleton* handles deserializing/unmarshalling the arguments and calling the local class, sending back the result to the remote client

SOAP 1.2

What

- SOAP (Simple Object Access Protocol) is used to serialize a remote procedure call (RPC) across the network
- Has its roots in distributed computing technologies
  - DCOM
  - CORBA
  - Java RMI

SOAP 1.2

What

- SOAP is a lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML
- SOAP is an XML-based protocol that consists of three parts
  - An envelope that defines a framework for describing what is in a message and how to process it
  - A set of encoding rules for expressing instances of application-defined datatypes (*i.e.*, how to serialize data structures)
  - A convention for representing remote procedure calls and responses

SOAP 1.2

HTTP Request

```
POST /warehouse/services/Purchasing HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.3
Host: 127.0.0.1
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 412

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <processPO
   soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <arg0 xsi:type="xsd:string">&lt;PO/&gt;</arg0>
  </processPO>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP 1.2

HTTP Response

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/xml;charset=utf-8
Date: Wed, 13 Apr 2005 14:09:02 GMT
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <processPOResponse
   soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <processPOReturn xsi:type="xsd:string">&lt;PO-ACK/&gt;</processPOReturn>
  </processPOResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP 1.2

Envelope

- `Envelope` is the top-level element. It defines the various namespaces used in the message
  - `Header` is an optional element used for carrying extra information about authentication, transactions, *etc*.
  - `Body` is the element containing the payload of the message

SOAP 1.2

Encoding Rules

- The SOAP specification describes how to serialize application-specific data-types into and out of an XML representation
  - XML Schema primitive types (`int`, `byte`, `short`, `boolean`, `string`, `float`, `double`, `date`, `time`, and `URL`) are sent as-is
  - More complicated objects (*e.g.*, Java classes) must have a matching schema and a mechanism for serializing into/out of schema

SOAP 1.2

SOAP Router

- A SOAP router
  - Listens on the appropriate protocol
  - Receives SOAP request
  - Has a binding between service's URN and implementing class
  - Calls appropriate class to handle request
  - Returns response to sender
- Apache provides a SOAP router called “Axis” that is an HTTP servlet that can be deployed in any webserver or servlet container

WSDL 1.1

What

- WSDL (Web Services Description Language) describes
  - what a web service can do
  - where it resides
  - how to invoke it
- WSDL is usually used with SOAP as a transport protocol, although it can work with other protocols as well

WSDL 1.1

definitions

```
<definitions name="CurrencyExchangeService"
             targetNamespace="http://www.xmethods.net/sd/CurrencyExchangeService.wsdl"
             xmlns:tns="http://www.xmethods.net/sd/CurrencyExchangeService.wsdl"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap"
             xmlns="http://schemas.xmlsoap.org/wsdl/">

  <message name="getRateRequest">...</message>
  <message name="getRateResponse">...</message>

  <portType name="CurrencyExchangePortType">
    ...
  </portType>

  <binding name="CurrencyExchangeBinding" type="CurrencyExchangePortType">
    ...
  </binding>

  <service name="CurrencyExchangeService">
    ...
  </service>

</definitions>
```

what operations does this service provide?

how are those operations invoked?

where is the service?

Adapted from http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl.

WSDL 1.1

message

```
<message name="getRateRequest">
  <part name="country1" type="xsd:string"/>
  <part name="country2" type="xsd:string"/>
</message>
<message name="getRateResponse">
  <part name="return" type="xsd:float"/>
</message>
```

A **message** is a single transmission of information going between the two parties; it has multiple parts that have either simple types (primitives) or more complicated types (defined in a schema)

Excerpted from http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl.

WSDL 1.1

portType

```
<portType name="CurrencyExchangePortType">
  <operation name="getRate">
    <input message="getRateRequest"
           name="tns:getRate"/>
    <output message="getRateResponse"
            name="tns:getRateResponse"/>
  </operation>
</portType>
```

A **portType** corresponds to a set of one or more operations, where each operation defines a specific input/output sequence; corresponds to the programmatic notion of an interface

Excerpted from http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl.

WSDL 1.1

binding

```
<binding name="CurrencyExchangeBinding" type="CurrencyExchangePortType">
  <soap:binding style="rpc"
                transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getRate">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="encoded"
       namespace="urn:xmethods-CurrencyExchange"
       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
      <soap:body use="encoded"
       namespace="urn:xmethods-CurrencyExchange"
       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
  </operation>
</binding>
```

A **binding** describes how a **portType** is implemented over a particular protocol; the soap message body is created using the type encoding specified by the soap specification. In this case, the protocol is RPC-style SOAP

Excerpted from http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl.

WSDL 1.1

service

```
<service name="CurrencyExchangeService">

  <documentation>Returns the exchange rate between the two
   currencies</documentation>

  <port name="CurrencyExchangePort"
        binding="tns:CurrencyExchangeBinding">
    <soap:address
     location="http://services.xmethods.net:80/soap"/>
  </port>

</service>
```

A **service** is a collection of related endpoints, described by **documentation**; a **port** describes the availability of a particular binding at an endpoint

Excerpted from http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl.

WSDL 1.1

types

- When a WSDL document declares operations that take more complicated types, XML Schema type definitions are included beneath the WSDL document's `definitions` element

```
<types>
  <xsd:schema targetNamespace="urn:AddressFetcher2"
   xmlns:xsd="http://www.w3.org/1999/XMLSchema">
    <xsd:simpleType name="stateType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="TX"/>
        <xsd:enumeration value="IN"/>
        <xsd:enumeration value="OH"/>
      </xsd:restriction>
    </xsd:simpleType>
    ...
</types>
```

Example excerpted from http://www.fas.harvard.edu/~cscie259/distribution/lectures/11/examples11/AddressBook/AddressBook.wsdl.

WSDL 1.1

Document Style

- Web service designers/users often think of the XML representation merely as the "wire format"
- They want RPC-style invocation, with binding to programmatic objects on both sides via serialization
- But sometimes, we want to just use WSDL/SOAP to send an XML document, without the RPC semantics
- To do this, we specify `<soap:binding style="document">` in the WSDL file

WSDL 1.1

Tools

- Toolkits help automate
  - Generating client code from a WSDL file for invoking the web service it describes
  - Generating a WSDL file from an object (Java, COM, Visual Basic class)

Axis 1.4

TaxService

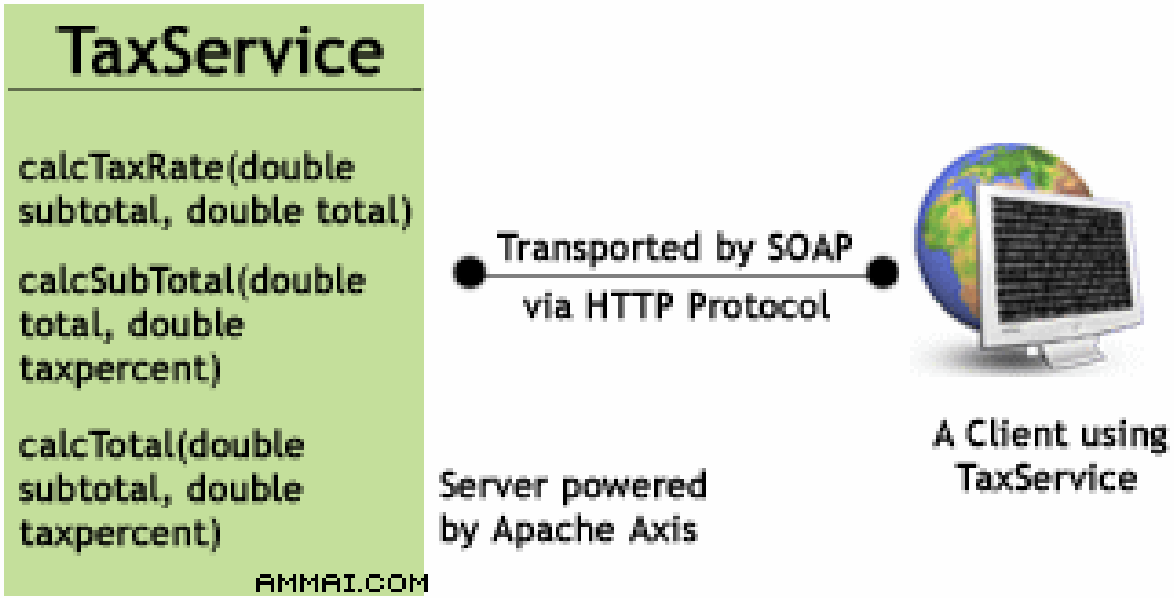
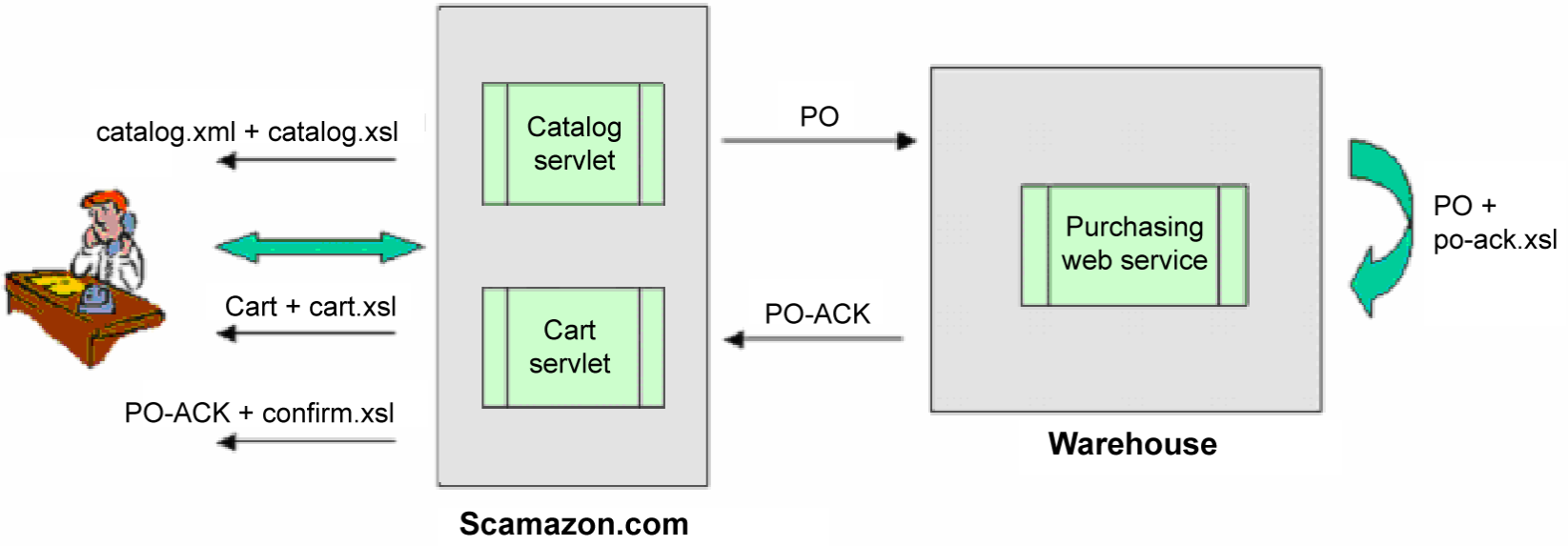


Image from http://www.ammai.com/modules.php?op=modload&name=Sections&file=index&req=viewarticle&artid=4&page=4.

Axis 1.4

Warehouse



Axis 1.4

AWS

Computer Science E-259

XML with Java

Lecture 11:
Web Services, SOAP 1.2, and WSDL 1.1

10 December 2007

David J. Malan

malan@post.harvard.edu