Lecture 15: Web Services

Wendy Liu
CSC309F – Fall 2007

Outline
- Web Services
- SOAP
- WSDL
- Web Services Stack
  - Axis2

Web Services
- A web service is a network accessible interface to application functionality, built using standard Internet technologies
- Web-based applications that use open, XML-based standards and transport protocols to exchange data with each other

Web Application Architecture
- Web Service
  - Option 1: Native application
  - Option 2: Web application

How do SOAP and WSDL fit in these pictures?
What is SOAP

- A messaging framework that specifies
  - Message format (XML)
  - Message processing
  - Message transport on underlying protocols (mainly HTTP)
- Human behaviour: how to extend SOAP with new features
- An attempt to enable software agents to do business on the Web; failed for the most part
  - Because software had no money to spend, could not be trusted with real business decisions
  - Was often used as “Web RPC” but too complex for that

SOAP Message Format

- SOAP Envelope • Required • Outermost element
- SOAP Header • Optional
- Header blocks
- SOAP Body • Required
- Message body • Application-specific (any XML payload) • Or a SOAP Fault

SOAP Message Processing

- SOAP message travels along a path of nodes
  - One-way transmission; like a Unix pipeline
  - Request-Response treated as 2 messages: “Message Exchange Pattern” (MEP)
  - Routing is not specified by SOAP
- SOAP nodes
  - Initial sender, intermediaries, ultimate receiver
  - Process and/or relay message
  - Assume a SOAP role towards message
- Header blocks control processing
  - Every header block is targeted at a SOAP role
  - A node in that role must perform message processing implied by header block, and delete header block before relaying onward
- Header blocks may be used to:
  - Authenticate
  - Route the message
  - Indicate preference to service broker
**SOAP over HTTP**

- 2 nodes exchanging 2 messages
- SOAP features:
  - Web-Method (GET, PUT, POST, DELETE)
  - Action (URI that identifies intent of message)
  - Response MEP (client sends no envelope; GET)
  - Request-Response MEP (client sends envelope; POST)
- Content-Type: application/soap+xml
  - Action is an optional parameter of Content-Type (like charset)
- HTTP status code indicates success/failure of SOAP processing
  - 2xx = success
  - 400 / 500 used in conjunction with a SOAP Fault in Body

**Remote Procedure Call (RPC)**

- Not a new idea. Review:
  - Identify remote component to call
  - Create local proxy with same interface
  - Call proxy
  - Parameters values are marshalled, sent to remote node, unmarshalled, and passed to component
  - Component’s return value is marshalled, sent back, unmarshalled, and returned by proxy
  - Programmer is barely aware the call was remote

**SOAP RPC**

- New idea:
  - Invocation over HTTP (port 80 open worldwide)
  - Marshalling data as XML (interoperable)
- For example:
  - Invoke Java method: int divide(int dividend, int divisor);
  - SOAP request body contains (namespace qualifiers omitted):
    ```xml
    <divide>
      <dividend>24</dividend>
      <divisor>7</divisor>
    </divide>
    ```
  - SOAP response body contains:
    ```xml
    <divideResponse>
      <result>3</result>
    </divideResponse>
    ```
  - On error, SOAP response body contains:
    ```xml
    <S:Fault>
      <S:Reason><S:Text>java.lang.ArithmeticException</S:Text></S:Reason>
    </S:Fault>
    ```

**Example: Calculator**

```java
public class Calculator {
    public int add(int i1, int i2) {
        return i1 + i2;
    }
    public int subtract(int i1, int i2) {
        return i1 - i2;
    }
}
```
Example: SOAP Request

POST /axis/Calculator HTTP/1.1
Host: localhost
Content-Type: text/xml
Content-Length: 586
SOAPAction: ""

<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>
    <calc:add xmlns:calc="Calculator">
      <calc:op1 xsi:type="xsd:int">50</calc:op1>
      <calc:op2 xsi:type="xsd:int">6</calc:op2>
    </calc:add>
  </soapenv:Body>
</soapenv:Envelope>

Example: SOAP Response

HTTP/1.1 200 OK
Set-Cookie: JSESSIONID=7B0EBF2F233CED633E6E4D172BC0631; Path=/axis
Content-Type: text/xml;charset=utf-8
Date: Tue, 23 Nov 2004 03:42:37 GMT
Server: Apache-Coyote/1.1
Connection: close

<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:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
>
  <soapenv:Body>
    <ns1:addResponse xmlns:ns1="Calculator">
      <addReturn xsi:type="xsd:int" />56</addReturn>
    </ns1:addResponse>
  </soapenv:Body>
</soapenv:Envelope>

Example: SOAP Error

HTTP/1.1 500 Internal Server Error
Content-Type: text/xml;charset=utf-8
Date: Tue, 23 Nov 2004 03:59:35 GMT
Server: Apache-Coyote/1.1
Connection: close

<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>
    <soapenv:Fault>
      <faultcode>soapenv:Server.userException</faultcode>
      <faultstring>java.io.FileNotFoundException: /Calculato.jws</faultstring>
      <detail>
        <ns1:hostname xmlns:ns1="http://xml.apache.org/axis/"/>simon
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>

WSDL

Web Service Description Language

http://www.w3.org/TR/wsdl
Web Service Description Language

Definition of an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages
- Service location
- Service methods
- Parameters and return types
- Data types

Elements of WSDL (1)

- Types
  - a container for data type definitions using some type system (e.g. XSD)
  - provide data type definitions used to describe the messages exchanged
- Message
  - an abstract, typed definition of the data being communicated
  - a message consists of logical parts, each of which is associated with a definition within some type system
- Operation
  - an abstract description of an action supported by the service

Elements of WSDL (2)

- Port Type
  - an abstract set of operations supported by one or more endpoints
  - each operation refers to an input message and output messages
- Binding
  - a concrete protocol and data format specification for a particular port type
  - specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType
- Port
  - a single endpoint defined as a combination of a binding and a network address
  - specifies an address for a binding, thus defining a single communication endpoint
- Service
  - a collection of related endpoints
  - used to aggregate a set of related ports

WSDL Document

- Define services as collections of network endpoints, or ports
  - A port is defined by associating a network address with a reusable binding
  - A collection of ports define a service
- Allow reuse of abstract definitions
  - messages
    - Abstract descriptions of the data being exchanged
  - port types
    - Abstract collections of operations
- Binding mechanism
  - Attach a specific protocol or data format or structure to an abstract message, operation, or endpoint
  - It allows the reuse of abstract definitions
WSDL Document Layout

```xml
<definitions>
  <types> … </types><message> … </message><portType>
    <operation> … </operation>
  …
</portType>
<binding>
  <operation> … </operation>
  …
</binding>
<service> … </service>
</definitions>
```

Example: Calculator

```java
public class Calculator {
  public int add(int i1, int i2) {
    return i1 + i2;
  }
  public int subtract(int i1, int i2) {
    return i1 - i2;
  }
}
```

<definitions>

- Root of the document
- Includes name space definitions
- Target name space for service definitions

```xml
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:ns0="http://ws.apache.org/axis2"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  targetNamespace="http://ws.apache.org/axis2">
```

<types>

- Provide data type definitions used to describe the messages exchanged

```xml
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    attributeFormDefault="qualified" elementFormDefault="qualified"
    targetNamespace="http://ws.apache.org/axis2">
    <xs:element name="add">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" name="param0" type="xs:int"/>
          <xs:element minOccurs="0" name="param1" type="xs:int"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
```
<message>

- Abstract and typed definitions of the data being communicated
  - `<wsdl:message name="addRequest">
      <wsdl:part name="parameters" element="ns0:add"/>
    </wsdl:message>`
  - `<wsdl:message name="addResponse">
      <wsdl:part name="parameters" element="ns0:addResponse"/>
    </wsdl:message>`

</message>

<portType>

- An abstract set of operations supported by the service
  - `<operation>
      - An abstract description of an action supported by the service
        - `<wsdl:operation name="add">
            - `<wsdl:input message="ns0:addRequest" wsaw:Action="urn:add"/>
            - `<wsdl:output message="ns0:addResponse" wsaw:Action="urn:addResponse"/>
          </wsdl:operation>`
        - `<wsdl:operation name="subtract">
            - `<wsdl:input message="ns0:subtractRequest" wsaw:Action="urn:subtract"/>
            - `<wsdl:output message="ns0:subtractResponse" wsaw:Action="urn:subtractResponse"/>
          </wsdl:operation>`
    </wsdl:operation>`

</portType>

<binding>

- A concrete protocol and data format specification for a particular portType
  - `<wsdl:binding name="CalculatorSOAP12Binding" type="ns0:CalculatorPortType">
      - `<wsdl:operation name="add">
          - `<soap12:operation soapAction="urn:add" style="document"/>
          - `<wsdl:input>
              - `<soap12:body use="literal"/>
          </wsdl:input>
          - `<wsdl:output>
              - `<soap12:body use="literal"/>
          </wsdl:output>
        </wsdl:operation>`
      - `<wsdl:operation name="subtract">
          - `<soap12:operation soapAction="urn:subtract" style="document"/>
          - `<wsdl:input>
              - `<soap12:body use="literal"/>
          </wsdl:input>
          - `<wsdl:output>
              - `<soap12:body use="literal"/>
          </wsdl:output>
        </wsdl:operation>`
    </wsdl:binding>`

</binding>

<service>

- A collection of related endpoints
  - `<port>
      - A single endpoint defined as a combination of a binding and a network address
        - `<wsdl:port name="CalculatorSOAP12port_http" binding="ns0:CalculatorSOAP12Binding">
            - `<soap12:address location="http://localhost:8080/axis2/services/Calculator"/>
          </wsdl:port>`
  
    ... 

</service>
Axis2 / Java

Apache Web Services Stack (Tool)

Creating Web Services

- Create service from skeleton code
- Create service from Java classes
- Create service from WSDL

Creating WS Clients

- Standalone client
  - Access the web service directly
- Web client
  - Access the web service via browser
- Both must make use of the client-side proxy of the web service (port)

Disadvantages of Axis

- Client stub is not a real proxy
  - Do not reflect the true signatures of the services
- Not following standard specifications
  - JAX-WS (Java API for XML-Based Web Services)
- Incomplete data binding
  - Rely on Java’s reflection mechanism
The Big Picture

So far ...

How Do the Topics Relate?

- **Client side**
  - XHTML, CSS
  - JavaScript, DOM
  - Content manipulation
- **Server side**
  - CGI, Java Servlets/JSP, and JDBC (database)
  - Web content provision
- **Communication protocol**
  - HTTP
- **Others**
  - XML
  - Concurrency control
  - Transaction
  - Web usability
- **Tools**
  - Tomcat
  - Derby
  - Eclipse

Web Application Architecture I

- **Static content**

```
Browser (HTTP) ➔ HTTP Server
```

- **Dynamic content**

```
Browser (HTTP) ➔ Servlet/CGI Engine ➔ JDBC ➔ DB
```

Web Application Architecture II

- **Web Service**
  - **Option 1: Native application**
    ```
    Client (SOAP/HTTP) ➔ Web Service
    ```
  - **Option 2: Web application**
    ```
    Browser (HTTP) ➔ Servlet/CGI Engine ➔ JDBC ➔ DB ➔ SOAP/HTTP ➔ Web Service
    ```