WS-Addressing

Sang Shin
Michèle Garoche
www.javapassion.com
“Learning is fun!”

Agenda

- What is and Why WS-Addressing?
- WS-Addressing and WSDL
- Message Information Headers
- WS-Addressing Annotations
- WS-Addressing and Stateful Web service

What is & Why WS-Addressing?

Motivation for WS-Addressing

- SOAP does not provide a standard way to specify
  - Where a message is going (destination)
  - How to return a response
  - Where to report an error.
- Those details have historically been left up to the transport layer.
  - For example, when a standard SOAP request is sent over HTTP, the URI of the HTTP request serves as the message's destination.
  - The message response is packaged in the HTTP response and received by the client over the HTTP connection.

What is WS-Addressing?

- Provides transport-neutral mechanisms to address Web services and messages
  - The transport-neutrality is achieved by normalizing the information typically shared between transport protocols and messaging systems.
- To serve this purpose, WS-Addressing defines two new constructs
  - Endpoint Reference (EPR)
  - Message Information Headers
- Supported from JAX-WS 2.1

What is WS-Addressing? (Cont.)

- Defines standard ways to route a message over multiple transports or direct a response to a third party.
  - For example, a client application might send a request over JMS and ask to receive the response through e-mail or SMS.

Transport Dependent Addressing

A SOAP 1.2 message, without WS-Addressing, sent over HTTP transport looks like:

```
(001)  POST /fabrikam/Purchasing HTTP 1.1/POST
(002)  Host: example.com
(003)
(004)
(005)  <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
                          xmlns:wombat="http://wombat.org/">
(006)    <S:Header>
(007)      <wombat:MessageID>
(008)        uuid:e197db59-0982-4c9c-9702-4234d204f7f4
(009)      </wombat:MessageID>
(010)    <S:Header>
(011)    <S:Body>
(012)      ...
(013)     </S:Body>
(014)  </S:Envelope>
```

Line (001) - (002) shows the HTTP transport headers. Line (005) - (014) shows the SOAP message in HTTP body.

Transport Dependent Addressing

- The host (example.com), the dispatch method (POST) and the URL to dispatch to (/fabrikam/Purchasing) are in the HTTP transport headers
- If the message is to be sent over an alternate transport, such as SMTP, then the information conveyed in HTTP transport headers need to be mapped to SMTP specific headers. On the server side, to dispatch successfully, a Web service stack has to gather the information from the SMTP (as opposed to HTTP) headers and the SOAP message.

Transport Dependent Addressing

- Also in the above message, there is no standard header to establish the identity of a message.
- In this case, *MessageID* header defined in the namespace bound to *wombat* prefix is used but is application specific and is thus not re-usable.

Transport Independent Addressing via WS-Addressing

A SOAP 1.2 message, with WS-Addressing, sent over HTTP transport

```
(001)  POST /fabrikam/Purchasing HTTP 1.1/POST
(002)  Host: example.com
(003)
(004)
(005)  <S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
                        xmlns:wsa="http://www.w3.org/2005/08/addressing/">
(006)    <S:Header>
(007)      <wsa:MessageID>
(008)        uuid:e197db59-0982-4c9c-9702-4234d204f7f4
(009)      </wsa:MessageID>
(010)      <wsa:To>
(011)        mailto:purchasing@example.com
(012)      </wsa:To>
(013)      <wsa:Action>
(014)        http://example.com/fabrikam/SubmitPO
(015)      </wsa:Action>
(016)    <S:Header>
(017)    <S:Body>
(018)      ...
(019)    </S:Body>
(020)  </S:Envelope>
```

Message Information Headers

Message Information Headers

- Define the full set of addressing information that can be attached to a SOAP message.
- Most of the fields are optional; the only required fields are the *To* and *Action* fields, each of which specifies a URI.

Message Information Headers

<wsa:MessageID> xs:anyURI </wsa:MessageID>

<wsa:RelatesTo RelationshipType="..."?
>xs:anyURI</wsa:RelatesTo>

<wsa:To>xs:anyURI</wsa:To>

<wsa:Action>xs:anyURI</wsa:Action>

<wsa:From>endpoint-reference</wsa:From>

<wsa:ReplyTo>endpoint-reference</wsa:ReplyTo>

<wsa:FaultTo>endpoint-reference</wsa:FaultTo>

To URI & Action URI

- *To* URI specifies the "where" and the *Action* URI specifies the "what":
- *Action* URI specifies service
- The separation of the *To* URI and the *Action* URI allows for lots of flexibility in configuring web service destinations.
  - For example, an e-mail address might receive requests for multiple services, all identified by their *Action* URI values.

Message Addressing Properties

```
<!-- Message 1 -->
<wsa:To>mailto:ws@example.com</wsa:To>
<wsa:Action>http://example.com/aservice</wsa:Action>
// ...

<!-- Message 2 -->
<wsa:To>mailto:ws@example.com</wsa:To>
<wsa:Action>http://example.com/anotherservice</wsa:Action>
// ...
```

Optional Message Addressing Properties

- ReplyTo
  - Must be specified only when the sender expects a response, but it can be used to route that response to any valid endpoint
- FaultTo
  - Routes SOAP fault messages to specified endpoint references
  - Always optional
- From
  - A service consumer identify itself to the service

Separation of Endpoints

- Explicitly separating the message source endpoint, expected reply endpoint, and fault handling endpoint from one another helps WS-Addressing support a variety of messaging models beyond the simple request/reply interactions we typically associate with web services.

Message ID

- When a reply is expected, whether it is expected by the sender or by a third endpoint specified in the *ReplyTo* header, a *MessageId* element must also be present.
- The message ID is a unique URI.
- Because web services can be used over unreliable transports, it is possible that an endpoint will receive duplicate copies of a message.
  - The message ID can be used to avoid processing the same message twice.

Relation Type

- *RelatesTo* element provides a standard way to associate incoming replies with their corresponding requests

WS-Addressing Annotations (in JAX-WS 2.x)

Annotations

- *@javax.xml.ws.soap.Addressing*
  - Is used to enable addressing.
  - With this annotation, the generated WSDL for this service contains *<wsaw:UsingAddressing/>* in the binding section
- *@Action* and *@FaultAction*
  - Are used to specify explicit *wsa:Action* values
  - Are reflected in the *portType* definitions of the generated wsdl
- Without any explicit action values, default values are used as per WS-Addressing specification

Example1: @Action – Default Value

- In the absence of the input and output attributes of *wsa:Action* element, the following pattern is used to construct a default action for inputs and outputs
  - [target namespace]/[port type name]/[input|output name]

Example1: @Action – Default Value

- **Use default values for @Action**

```
@javax.jws.WebService
 public class AddNumbersImpl {
    @javax.ws.addressing.Action
    public int addNumbers(int number1, int number2) {
       return number1 + number2;
    }
 }
```

- **Generated WSDL**

```
<definitions targetNamespace="http://example.com/numbers" ...>
...
  <portType name="AddNumbersPortType">
    <operation name="AddNumbers">
      <input message="tns:AddNumbersInput" name="Parameters"
 wsa:Action="http://example.com/numbers/AddNumbersPortType/
 Parameters"/>
     <output message="tns:AddNumbersOutput" name="Result"
 wsa:Action="http://example.com/numbers/AddNumbersPortType/
 Result"/>
    </operation>
  <portType>
...
<definitions>
```

Example2: @Action – input

- **input attribute is specified and output attribute is derived from input**

```
@javax.jws.WebService
 public class AddNumbersImpl {
    @javax.ws.addressing.Action(input="http://example.com/numbers/add")
    public int addNumbers(int number1, int number2) {
       return number1 + number2;
    }
 }
```

- **Generated WSDL**

```
<definitions targetNamespace="http://example.com/numbers" ...>
...
  <portType name="AddNumbersPortType">
    <operation name="AddNumbers">
      <input message="tns:AddNumbersInput" name="Parameters"
        wsa:Action="http://example.com/numbers/add"/>
      <output message="tns:AddNumbersOutput" name="Result"
        wsa:Action="http://example.com/numbers/addResponse"/>
    </operation>
  <portType>
...
<definitions>
```

Example3: Annotations in Java Code

```
@Addressing
@WebService
public class AddNumbersImpl {

    @Action(
            input = "http://example.com/input",
            output = "http://example.com/output")
    public int addNumbers(int number1, int number2) throws
    AddNumbersException {
        return impl(number1, number2);
    }
    @Action(
            input = "http://example.com/input3",
            output = "http://example.com/output3",
            fault = {
            @FaultAction(className = AddNumbersException.class,
    value = "http://example.com/fault3")
                    }
    )
    public int addNumbers3(int number1, int number2) throws
    AddNumbersException {
        return impl(number1, number2);
    }
```

Example3: WSDL Document

```
<portType name="AddNumbersImpl">
    <operation name="addNumbers">
        <input wsaw:Action="http://example.com/input"
  message="tns:addNumbers"></input>
        <output wsaw:Action="http://example.com/output"
  message="tns:addNumbersResponse"></output>
        <fault message="tns:AddNumbersException"
  name="AddNumbersException"></fault>
    </operation>

    <operation name="addNumbers3">
        <input wsaw:Action="http://example.com/input3"
  message="tns:addNumbers3"></input>
        <output wsaw:Action="http://example.com/output3"
  message="tns:addNumbers3Response"></output>
        <fault message="tns:AddNumbersException"
  name="AddNumbersException"
  wsaw:Action="http://example.com/fault3"></fault>
    </operation>
</portType>
```

Example: SOAP Request Message

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Header>
      <To
   xmlns="http://www.w3.org/2005/08/addressing">http://localhost:4040/jax
   ws-fromjava-wsaddressing/addnumbers</To>
      <Action
   xmlns="http://www.w3.org/2005/08/addressing">http://server.fromjava_w
   saddressing/AddNumbersImpl/addNumbers2Request</Action>
      <ReplyTo xmlns="http://www.w3.org/2005/08/addressing">

   <Address>http://www.w3.org/2005/08/addressing/anonymous</Address>
      </ReplyTo>
      <MessageID
   xmlns="http://www.w3.org/2005/08/addressing">uuid:b734fc16-1cbb-
   4201-a944-7d593babf0f3</MessageID>
   </S:Header>
   <S:Body>
      <ns2:addNumbers2 xmlns:ns2="http://server.fromjava_wsaddressing/">
         <arg0>10</arg0>
         <arg1>10</arg1>
      </ns2:addNumbers2>
   </S:Body>
</S:Envelope>
```

Example: SOAP Response Message

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Header>
      <To
   xmlns="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/
   08/addressing/anonymous</To>
      <Action
   xmlns="http://www.w3.org/2005/08/addressing">http://server.fromjava_w
   saddressing/AddNumbersImpl/addNumbers2Response</Action>
      <MessageID
   xmlns="http://www.w3.org/2005/08/addressing">uuid:9d395f31-40a3-
   4c47-a396-cd68564d674f</MessageID>
      <RelatesTo
   xmlns="http://www.w3.org/2005/08/addressing">uuid:b734fc16-1cbb-
   4201-a944-7d593babf0f3</RelatesTo>
   </S:Header>
   <S:Body>
      <ns2:addNumbers2Response
   xmlns:ns2="http://server.fromjava_wsaddressing/">
         <return>20</return>
      </ns2:addNumbers2Response>
   </S:Body>
</S:Envelope>
```

WS-Addressing & Stateful Web Service

WS-Addressing and Stateful Service

- WS-Addressing also defines a standard for including service-specific attributes within an address
  - These attributes are particularly useful for the creation of stateful web services, which are services that can receive a series of requests from a particular client and remember some state information between requests.

Thank you!

Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps
"Learn with Passion!"