Another Random Developer Blog

Under development by Siegfried Bolz

Mar 08 2008

How to modify JAX-WS SOAP-Messages

Published by Siegfried Bolz at 1:02 am under Java, NetBeans, web services

This blog is depending on my previous blog Creating SOAP Web Services with NetBeans 6. You can download the needed sourcecode there.

Here i will show how you can modify the JAX-WS SOAP-Message before the outgoing SOAP-Response will pass the specified Web Service Operation. This is sometimes necessary, when some recipients want to receive their own special SOAP-Headers. You can also modify the incoming Request to read special informations, which are included in the SOAP-Header, but this is not part of this posting (maybe i will show it in an other upcoming blog).

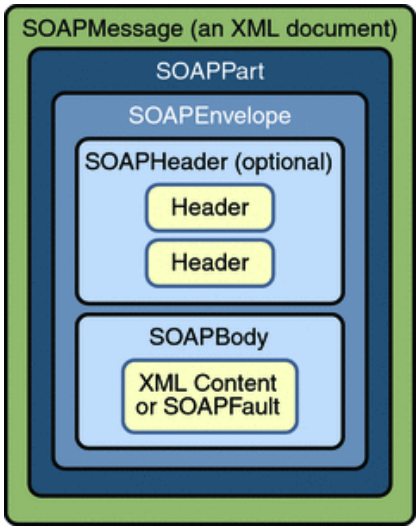
For this example i am using JAX-WS 2.1 with NetBeans 6.0 .

new ServerSOAPHandler.java

This is the core-class of this example. Before the Request/Response pass the Web Service Operation, it can be handled here in the method

```
view plain  copy to clipboard  print
01. public boolean handleMessage(SOAPMessageContext context)
```

The outbound-part (line 101) adds the SOAP-Header with four additional elements and one element for the SOAP-Body. The generated SOAP-Message is dumped to the console at runtime. Read the included comments to see how it works.



Create the following file:

ServerSOAPHandler.java

view plain copy to clipboard print

```
01. package eu.jdevelop.soapwebservices.handler;
02.
03. import javax.xml.ws.handler.soap.SOAPMessageContext;
04. import javax.xml.ws.handler.soap.SOAPHandler;
05. import javax.xml.ws.handler.MessageContext;
06. import javax.xml.soap.SOAPMessage;
07. import javax.xml.soap.SOAPBody;
08. import javax.xml.soap.SOAPElement;
09. import javax.xml.soap.SOAPException;
10. import javax.xml.namespace.QName;
11. import java.util.Set;
12. import java.io.ByteArrayOutputStream;
13. import javax.annotation.PostConstruct;
14. import javax.annotation.PreDestroy;
15. import javax.xml.soap.SOAPBodyElement;
16. import javax.xml.soap.SOAPEnvelope;
17. import javax.xml.soap.SOAPPart;
18.
19. /**
20.  * This class handles the SOAP-Requests before they reach the
21.  * Web Service Operation. It is possible to read and manipulate
22.  * the SOAP-Message.
23.  *
24.  * @author Siegfried Bolz
25.  */
26. public class ServerSOAPHandler implements SOAPHandler<soapmessageCor
27.
28.     /**
29.      * Is called after constructing the handler and before executing
30.      */
31.     @PostConstruct
32.     public void init() {
33.     }
34.
35.     /**
36.      * Returns the <code>Set</code> of supported SOAP headers
37.      */
38.     public Set<qname> getHeaders() {
39.         return null;
40.     }
41.
42.     /**
43.      * Returns the message encoding (e.g. utf-8)
44.      *
45.      * @param msg
46.      * @return
47.      * @throws javax.xml.soap.SOAPException
48.      */
49.     private String getMessageEncoding(SOAPMessage msg) throws SOAPE
50.         String encoding = "utf-8";
51.         if (msg.getProperty(SOAPMessage.CHARACTER_SET_ENCODING) != n
52.             encoding = msg.getProperty(SOAPMessage.CHARACTER_SET_ENC
53.         }
54.         return encoding;
```

```
55.     }
56.
57.     /**
58.      * Dump SOAP Message to console
59.      *
60.      * @param msg
61.      */
62.     private void dumpSOAPMessage(SOAPMessage msg) {
63.         if (msg == null) {
64.             System.out.println("SOAP Message is null");
65.             return;
66.         }
67.         System.out.println("");
68.         System.out.println("--------------------");
69.         System.out.println("DUMP OF SOAP MESSAGE");
70.         System.out.println("--------------------");
71.         try {
72.             ByteArrayOutputStream baos = new ByteArrayOutputStream()
73.             msg.writeTo(baos);
74.             System.out.println(baos.toString(getMessageEncoding(msg)
75.
76.             // show included values
77.             String values = msg.getSOAPBody().getTextContent();
78.             System.out.println("Included values:" + values);
79.         } catch (Exception e) {
80.             e.printStackTrace();
81.         }
82.     }
83.
84.     /**
85.      * This method handles the incoming and outgoing SOAP-Message.
86.      * It's an excellent point to manipulate the SOAP.
87.      *
88.      * @param SOAPMessageContext
89.      * @return boolean
90.      */
91.     public boolean handleMessage(SOAPMessageContext context) {
92.
93.         //Inquire incoming or outgoing message.
94.         boolean outbound = (Boolean) context.get(MessageContext.MESS
95.
96.         try {
97.
98.             if (outbound) {
99.                 // OUTBOUND
100.
101.                System.out.println("Direction=outbound (handleMessag
102.                SOAPMessage msg = ((SOAPMessageContext) context).get
103.
104.                // get SOAP-Part
105.                SOAPPart sp = msg.getSOAPPart();
106.
107.                //edit Envelope
108.                SOAPEnvelope env = sp.getEnvelope();
109.
110.                // add namespaces
111.                env.addNamespaceDeclaration("blog", "http://blog.jde
112.                env.addNamespaceDeclaration("xsd", "http://www.w3.or
113.                env.addNamespaceDeclaration("xsi", "http://www.w3.or
114.                env.addNamespaceDeclaration("soap", "http://schemas.
115.
116.                // add the Header with additional Elements
117.                SOAPElement soapElement1 = env.addHeader().addHeader
118.                soapElement1.addTextNode("1");
119.
120.                SOAPElement soapElement2 = env.getHeader().addHeader
121.                soapElement2.addTextNode("gzip");
122.
123.                SOAPElement soapElement3 = env.getHeader().addHeader
124.                soapElement3.addTextNode("jdevelop");
125.
126.                SOAPElement soapElement4 = env.getHeader().addHeader
127.                soapElement4.addTextNode("blog");
128.
129.                // get the SOAP-Body
130.                SOAPBody body = env.getBody();
131.
132.                // add an additional Element to the SOAP-Body
133.                SOAPBodyElement bodyElement = body.addBodyElement(ne
134.                bodyElement.addTextNode("Security through obscurity'
135.
136.                // print SOAP-Message
137.                dumpSOAPMessage(msg);
138.
139.            } else {
140.                // INBOUND
```

```
141.
142.                 System.out.println("Direction=inbound (handleMessage
143.                 SOAPMessage msg = ((SOAPMessageContext) context).get
144.                 dumpSOAPMessage(msg);
145.
146.              }
147.
148.         } catch (Exception e) {
149.
150.             //All other unhandled problems.
151.             e.printStackTrace();
152.         }
153.         return true;
154.     }
155.
156.     /**
157.      * Handles SOAP-Errors.
158.      *
159.      * @param context
160.      * @return
161.      */
162.     public boolean handleFault(SOAPMessageContext context) {
163.         System.out.println("ServerSOAPHandler.handleFault");
164.         boolean outbound = (Boolean) context.get(MessageContext.MESS
165.         if (outbound) {
166.             System.out.println("Direction=outbound (handleFault)");
167.         } else {
168.             System.out.println("Direction=inbound (handleFault)");
169.         }
170.         if (!outbound) {
171.             try {
172.                 SOAPMessage msg = ((SOAPMessageContext) context).get
173.                 dumpSOAPMessage(msg);
174.                 if (context.getMessage().getSOAPBody().getFault() !=
175.                     String detailName = null;
176.                     try {
177.                         detailName = context.getMessage().getSOAPBod
178.                         System.out.println("detailName=" + detailNam
179.                     } catch (Exception e) {
180.                     }
181.                 }
182.             } catch (SOAPException e) {
183.                 e.printStackTrace();
184.             }
185.         }
186.         return true;
187.     }
188.
189.     public void close(MessageContext messageContext) {
190.     }
191.
192.     /**
193.      * Is executed before this handler is being destroyed -
194.      * means after close() has been executed.
195.      */
196.     @PreDestroy
197.     public void destroy() {
198.     }
199.
200. } // .EOF
```

new serversoaphandler.xml

This is the configuration-file for the handler-chain. Create it at the source-root.

serversoaphandler.xml

view plain copy to clipboard print

```
01. <?xml version="1.0" encoding="UTF-8"?>
02. <jws:handler-chains xmlns:jws="http://java.sun.com/xml/ns/javaee">
03.
04.   <jws:handler-chain>
05.     <jws:handler>
06.       <jws:handler-name>ServerSOAPHandler</jws:handler-name>
07.       <jws:handler-class>eu.jdevelop.soapwebservices.handler.ServerS
08.     </jws:handler>
09.   </jws:handler-chain>
10.
```

```
11.  </jws:handler-chains>
```

ServiceImpl.java

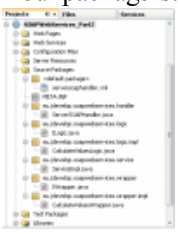
To load the configuration-file, just add this line of code under the **"@WebService"** -annotation.

```
view plain  copy to clipboard  print
01.  @HandlerChain(name = "ServerSOAPHandler", file = "serversoaphandler.
```

ServiceImpl.java

```
view plain  copy to clipboard  print
01.  package eu.jdevelop.soapwebservices.service;
02.
03.  import eu.jdevelop.soapwebservices.CalculateValues;
04.  import eu.jdevelop.soapwebservices.CalculateValuesResponse;
05.  import eu.jdevelop.soapwebservices.SOAPWebServices;
06.  import eu.jdevelop.soapwebservices.wrapper.impl.CalculateValuesWrapp
07.  import javax.jws.WebService;
08.  import javax.jws.WebService;
09.  import javax.jws.HandlerChain;
10.
11.  /**
12.   * This is the Service-Implementation of the Web Service. Here are t
13.   * operations which can be called from web clients.
14.   *
15.   * @author Siegfried Bolz
16.   */
17.  @WebService(serviceName = "SOAPService", portName = "WebServices", e
18.  @HandlerChain(name = "ServerSOAPHandler", file = "serversoaphandler.
19.  public class ServiceImpl implements SOAPWebServices {
20.
21.      public CalculateValuesResponse getCalculateValues(CalculateValue
22.
23.          try {
24.              CalculateValuesWrapper wrapper = new CalculateValuesWrap
25.              return wrapper.getResult(calculateValues);
26.
27.          } catch (Exception x) {
28.              throw new IllegalStateException(x);
29.          }
30.      }
31.
32.  } // .EOF
```

Your package-structure should now look like this:



Testing

Build, deploy and start the application. Use **soapui** to submit a request.

You will receive the following SOAP-Message:

view plain copy to clipboard print

```
<s:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlr
   <s:Header>
      <security xmlns="http://blog.jdevelop.eu/securityprotocol">1</
      <compression xmlns="http://blog.jdevelop.eu/securityprotocol">
      <username xmlns="http://blog.jdevelop.eu/securityprotocol">jde
      <password xmlns="http://blog.jdevelop.eu/securityprotocol">blo
   </s:Header>
   <s:Body>
      <ns2:calculateValuesResponse xmlns:ns2="soapwebservices.jdevel
         <result>15.33</result>
      </ns2:calculateValuesResponse>
      <message xmlns="http://blog.jdevelop.eu/securitymessage">Secur
   </s:Body>
</s:Envelope>
```

For comparison, the response without using the soaphandler looks like this:

view plain copy to clipboard print

```
<s:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlr
   <s:Body>
      <ns2:calculateValuesResponse xmlns:ns2="soapwebservices.jdevel
         <result>15.33</result>
      </ns2:calculateValuesResponse>
   </s:Body>
</s:Envelope>
```

And here the request:

view plain copy to clipboard print

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/env
<soapenv:Header>
    <security xmlns:soap="http://blog.jdevelop.eu/securityprotocol"
    <username xmlns:soap="http://blog.jdevelop.eu/securityprotocol"
    <password xmlns:soap="http://blog.jdevelop.eu/securityprotocol"
</soapenv:Header>
   <soapenv:Body>
      <soap:calculateValues>
         <value1>10</value1>
         <value2>5.33</value2>
      </soap:calculateValues>
   </soapenv:Body>
</soapenv:Envelope>
```

Download

NetBeans 6 Project: download

Technorati Tags: Java, jax-ws, NetBeans, soap, soaphandler, web services

Tags: Java, jax-ws, NetBeans, soap, soaphandler, web services

13 Responses to “How to modify JAX-WS SOAP-Messages”

1. # *Salfordon 17 Apr 2008 at 4:06 pm*

   I have followed this but the new added header dont display, but body ones displays, what might be the cause

2. # *Siegfried Bolzon 17 Apr 2008 at 6:24 pm*

   Please download my example and try it to see if everything is working fine.

3. # *Dieteron 06 Feb 2009 at 9:16 am*

Hi Siegfried,

vielen Dank, Dein Beispiel ist aufschlussreich, was das zippen in Jax-Ws angeht. Ich habe aber noch ein anderes Problem zu lösen. Ich muß für einen .net Webservice mit jax-ws eine SOAP-Message in der Version 1.1 anstatt Standard 1.2 erzeugen. Kann ich da was im apt einstellen oder hilft mir die Implementierung des Handlers mit Bezug auf eine SOAP-Version dabei?

Vielen Dank und Grüsse aus Essen

Dieter

4. # *Siegfried Bolzon 07 Feb 2009 at 8:23 pm*

   Hi Dieter,
   es müsste möglich sein die SOAP-Message so zu manipulieren das daraus eine Version 1.1 entsteht, allerdings gibt es da große Unterschiede die du beachten must (siehe Links unten). Mit dem ServerSoapHandler müsste diese Manipulation möglich sein.

   http://www.w3.org/2003/06/soap11-soap12.html
   http://www.idealliance.org/papers/xmle02/dx_xmle02/papers/02-02-02/02-02-02.html

   Viele Grüße
   Siegfried

5. # *Diegoon 16 Feb 2009 at 2:53 pm*

   Hi, thanks a lot for this part of the code:

   ByteArrayOutputStream baos = new ByteArrayOutputStream();
   msg.writeTo(baos);

   I was trying to write the content of the message to a log. Previously i was using msg.writeTo(System.out), and when I changed my System.out logs to log4J, it was driving me crazy (I haven't even think about dropping the content to an Output Stream variable!).

   Now with
   logger.info("Message: " + baos.toString) i can log my messages again.

   Best regards,
   Diego.

6. # *Siddharthaon 25 Jun 2009 at 2:02 pm*

   Hi Siegfried,

   This code is able to handle the request & response
   but when the webservice is throwing an exception it's not able to handle it.

   My WS Code.

   @WebService(name = "TestWS",targetNamespace="http://ws.test")
   @SOAPBinding(style = SOAPBinding.Style.DOCUMENT,
   use = SOAPBinding.Use.LITERAL)
   @HandlerChain(name="ServerSOAPHandler",file="../../handler/serversoaphandler.xml")
   public class TestWS {
   public String testWs(String str) throws Exception{
   if(some condition){
   throw new Exception("Sidd Exp");
   }
   return str;
   }
   }

7. # *Rajivon 29 Jun 2009 at 8:43 pm*

   I'm trying to get the exception handling to work. I'm trying to get the below-

   ABC

xyz
blah

I created my own exception, and I'm able to set the faultstring. but when I use @HandlerChain(file='xxx.xml') the faultstring is changed to

Exception raised from invocation of public java.lang.String …..

could you advise why this could be happening.

Thanks in advace,
Rajiv.

8. # *Rajivon 29 Jun 2009 at 8:48 pm*

   oops the xml tags were ripped in the above -

   <SOAP-ENV:Fault>
   <faultcode>ABC</faultcode>
   <faultstring>xyz</faultstring>
   </SOAP-ENV:Fault>

9. # *SSon 09 Feb 2010 at 8:15 am*

   Suppose I want to change SOAP response Envelope attribute from to standard format like using handler class. Could you please tell me what changes need to be done here?

   Thanks in Advance

10. # *SSon 09 Feb 2010 at 10:29 am*

    More over , is there any way to identify the service name from Soap Fault in the Handler

11. # *Davidon 12 Apr 2011 at 11:08 am*

    Hi a good article

    I try to modify a soap incoming using a Soap Handler

    private void logToSystemOut(SOAPMessageContext smc) {

    try {

    Boolean outboundProperty = (Boolean) smc.get (MessageContext.MESSAGE_OUTBOUND_PROPERTY);
    if (outboundProperty.booleanValue()) {
    HttpServletRequest req = (HttpServletRequest) smc.get("javax.xml.ws.servlet.request");

    SOAPMessage msg = ((SOAPMessageContext) smc).getMessage();
    //SOAPMessageContext smc = (SOAPMessageContext) context;
    //SOAPMessage msg = smc.getMessage();
    SOAPPart sp = msg.getSOAPPart();

    SOAPEnvelope se = sp.getEnvelope();

    raise a next exception
    error:Unable to create envelope from given source
    by
    Caused by: javax.xml.transform.TransformerException: org.w3c.dom.DOMException: NAMESPACE_ERR: An attempt is made to create or change an object in a way which is incorrect with regard to namespaces.

    Caused by: org.w3c.dom.DOMException: NAMESPACE_ERR: An attempt is made to create or change an object in a way which is incorrect with regard to namespaces.

    may i help me

I am using metro 1.5 , java 6

12. # *viralon 24 Nov 2011 at 2:23 pm*

    I am looking for logging soap request and response xml.

    I tried this but it doesn't work for me.

    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    msg.writeTo(baos);

    it prints blank but webservice works fine and gives me result

    anyone have any other solution or api to convert soapmessage to XML using java?

13. # *Andrewon 29 May 2012 at 4:54 pm*

    Hi,
    love your work! I'm trying to add an attribute to an element and hoped your approach would help. I have an element thus:

    and I want it the element in the response to be have attribute values as below:

    Any clues as to what to set to arrive at this?

    Any advice is much appreciated,

    Andrew

Trackback URI | Comments RSS

Leave a Reply

Name (required)

Mail (hidden) (required)

Website

Security code (Required)*
To prove that you're not a bot, enter this code

Note: To submit your comment you have to preview it at first!

Preview

- Search for:
  Search

- # Recent Posts

  - SBQueue - An Objective-C Queue for iOS 4.3

- How to create a Grails Web Service with a Selenium Test
- Simple Validation-Framework for UITextField & UITextView -Fields
- Read some interesting data from your iDevice with iOS 4.3
- Use Generators to create boilerplate code in GWT 2.0
- Experiences with the migration from GWT 1.7.1 to 2.0
- Fixed the Plugin "EclipseMode" for IntelliJ IDEA 9
- Create a GWT Application from Scratch
- Solve the circular reference problem with Guice and Spring
- NetBeans RSS Reader - My first iPhone application released !

Categories

Select Category

Google Friend Connect

Listed on