SOAP

Autor: Piotr Sobczak

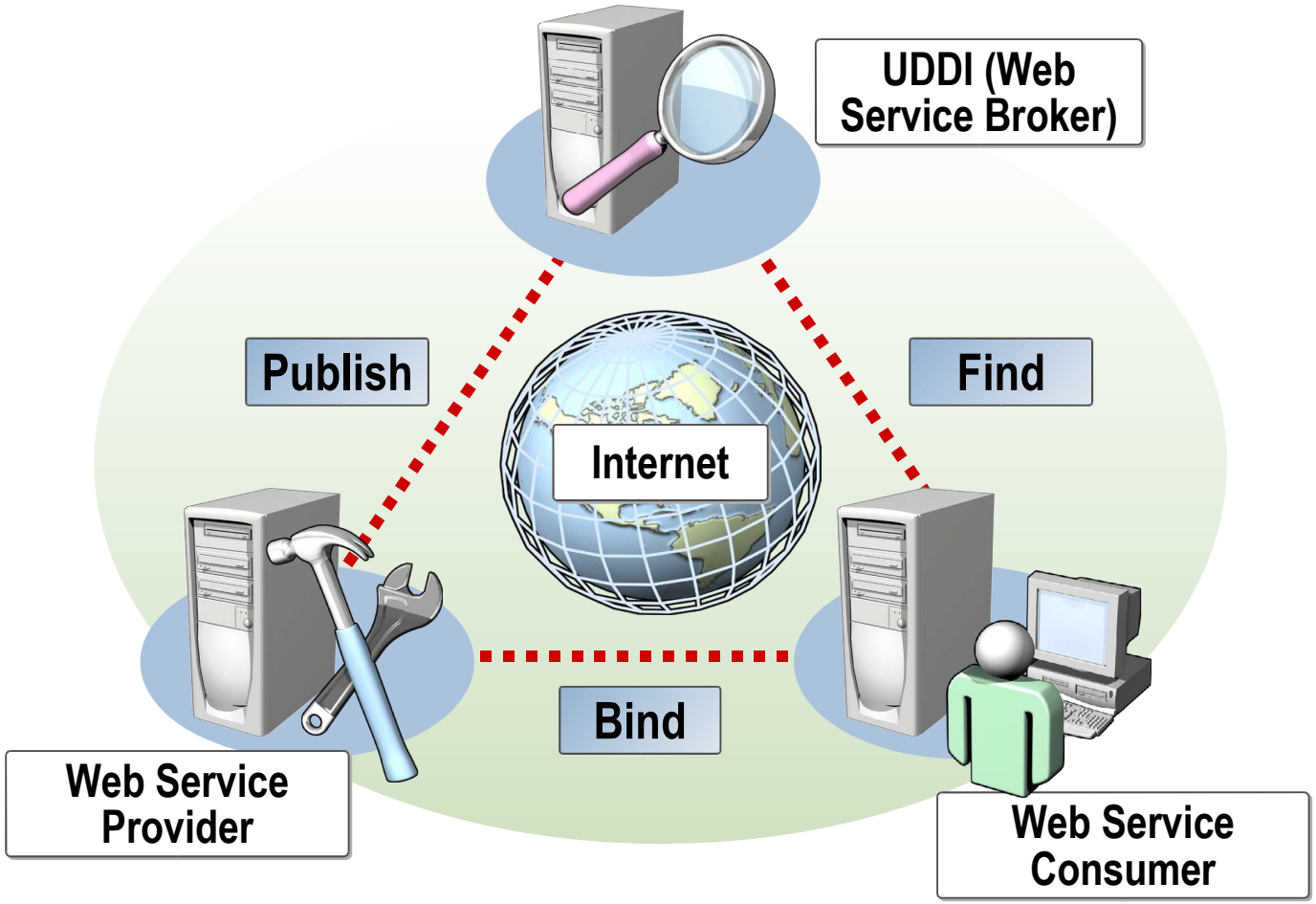
AGENDA:

- Trochę o Web Services
- Wprowadzenie do SOAP
- Anatomia komunikatu SOAP
- Wysyłanie i otrzymywanie komunikatu SOAP oraz API Javy w przykładach
- SOAP z załącznikami
- SOAP-RPC
- Obsługa błędów w SOAP
- Podsumowanie i bibliografia

Web Services 1/2

- UDDI
- WSDL, XML Schema
- **SOAP**
- XML
- HTTP, SMTP,...

Web Services 2/2



źródło: MSDN

SOAP - podstawy

- (Simple Object Access Protocol)
- Definicja: „Oparty na języku XML protokół wymiany informacji w rozproszonym środowisku.”
- Silne wspierany przez wielkie koncerny: Microsoft, IBM, Sun Microsystems.

Anatomia komunikatu SOAP

Cztery najważniejsze składniki to:

- koperta SOAP (envelope)
- wiązanie z protokołem
- zasady kodowania
- mechanizm wywołań zdalnych procedur

Koperta SOAP

- Składa się z nagłówka (header) - opcjonalnego oraz treści (body) – obowiązkowej

```
<?xml version='1.0' encoding='UTF-8' ?>
<soap-env:Envelope
      xmlns:soap-env=http://www.w3.org/2001/06/soapenvelope
        …
      <soap-env:Header>
        …
      </soap-env:Header>

     <soap-env:Body>
        …
      </soap-env:Body>
</soap-env:Envelope>
```

Nagłówek SOAP

- Nagłówek i treść są syntaktycznie podobne.
- Komunikujące się ze sobą strony powinny ustalić zawartość nagłówka.

- ebXML Message Service oparty o SOAP formalizuje użycie nagłówka:

```
<soap-env:header>
  <MessageHeader>
     <From>ME  </From>
      <To> YOU </To>
      <MessageId>9999</MessageId>
           …
  </MessageHeader>
</soap-env:header>
```

- Przy wykorzystaniu SOAP do RPC, nagłówek służy do opisania infrastruktury natomiast treść do wywołania metody i parametrów,

Wiązanie SOAP z protokołem

- Wiązanie z protokołem HTTP:

SOAPAction = ”urn:soaphttpclient-action-uri”
Host = localhost
Content-Type= text/xml; charset=utf-8
Content-Length= 701

```
<?xml version=‘1.0’ encoding=‘UTF-8’ ?>
<soap-env:Envelope
      xmlns:soap-env=http://www.w3.org/2001/06/soapenvelope
        …
</soap-env:Envelope>
```

Przykład – nadajnik SOAP 1/3

```
Import java.org.apache.soap.*;

public class SimpleGenericHTTPSoapClient  {
   …
   public static void main(String args[]) {

      private String hostURL ;
      private String dataFileName;

       try {
           SimpleGenericHTTPSoapClient soapClient =
              new SimpleGenericHTTPSoapClient(hostURL, dataFileName);
           soapClient.sendSOAPMessage();

         } catch(Exception e) {
                        System.out.println(e.getMessage());
          }
   } }
```

Przykład – nadajnik SOAP 2/3

```
public void sendSOAPMessage() {
  try {   FileReader fr = new FileReader (m_dataFileName);

    javax.xml.parsers.DocumentBuilder xdb =
   org.apache.soap.util.xml.XMLParserUtils.getXMLDocBuilder();

     Document doc = xdb.parse (new org.xml.sax.InputSource (fr));
    if (doc == null) { throw new org.apache.soap.SOAPException(); }

    Envelope envelope = new Envelope();

    Vector  headerElements = new Vector();
    Element  headerElement=doc.createElementNS(URI,"jaws:MessageHeader");
     …//dodajemy elementy do naglowka
    headerElements.add(headreElement);

   Header header = new Header();  header.setHeaderEntries(headerElements);
   envelope.setHeader(header);

    Vector bodyElements = new Vector();

  bodyElements.add(doc.getDocumentElement ());
```

```
 Body body = new Body();

 body.setBodyEntries(bodyElements);

envelope.setBody(body);

org.apache.soap.messaging.Message msg = new org.apache.soap.messaging.Message();

 msg.send (new java.net.URL(m_hostURL), URI, envelope);

System.out.println("Sent SOAP Message with Apache HTTP SOAP Client.");
System.out.println("Waiting for response....");

org.apache.soap.transport.SOAPTransport st = msg.getSOAPTransport ();
    BufferedReader br = st.receive ();
    String line = br.readLine();
    if(line == null)  {
        System.out.println("HTTP POST was successful. \n");
    } else  {
       while (line != null) {
            System.out.println (line);
            line = br.readLine();
  } } }
 catch(Exception e)  { e.printStackTrace(); }
```

Serwlet odbiornik- przykład 1/3

```
public class HTTPReceive extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException  {
        System.out.println("Received GET request");
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
    }

public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
  {
      for(Enumeration enum = request.getHeaderNames();
            enum.hasMoreElements(); )   {
                String header = (String)enum.nextElement();
                String value  = request.getHeader(header);
              System.out.println("  " + header + " = " + value);
        }

    if(request.getContentLength() > 0) {
        try{   …
```

Serwlet odbiornik- przykład 2/3

```
java.io.BufferedReader reader = request.getReader();

javax.xml.parsers.DocumentBuilder xdb =
            org.apache.soap.util.xml.XMLParserUtils.getXMLDocBuilder();
Document doc =  xdb.parse (new org.xml.sax.InputSource (reader));
if (doc == null) {
            System.out.println("Doc is null!");
            throw new org.apache.soap.SOAPException
   (org.apache.soap.Constants.FAULT_CODE_CLIENT, "parsing error");
} else  {
      Envelope env = Envelope.unmarshall(doc.getDocumentElement());

       Body body = env.getBody();
       Vector bodyEntries = body.getBodyEntries();
       StringWriter writer = new StringWriter();
```

Serwlet odbiornik- przykład 3/3

```
 for (Enumeration e = bodyEntries.elements(); e.hasMoreElements();)  {

        Element el = (Element) e.nextElement();

        org.apache.soap.util.xml.DOM2Writer.
          serializeAsXML( (Node)el , writer);
  }
System.out.println(writer.toString());
 }
 }    catch(Exception e) {
                              System.out.println(e);
                                  }
}
```

SOAP z załącznikami 1/3

- SOAP with attatchments (SwA)
- Niektóre dane niezbyt odpowiadają językowi XML np. pliki binarne
- SwA łączy SOAP z formatem MIME
- Przyjęty model jest dokładnie taki sam, jak model umieszczania załączników w poczcie.
- Do SwA wykorzystuje się Apache SOAP i zestaw funkcji JavaMail API.

SOAP z załącznikami 2/3

```
 import javax.mail.internet.MimeBodyPart;
if (m_attachment != null)  {
            BufferedReader attachmentReader =
              new BufferedReader(new FileReader(m_attachment));

            StringBuffer buffer = new StringBuffer();
            for (String line = attachmentReader.readLine(); line != null;
                  line = attachmentReader.readLine() )  {
              buffer.append(line);
            }

            MimeBodyPart attachment = new MimeBodyPart();
            attachment.setText(buffer.toString());
            attachment.setHeader("Content-ID", "the-attachment");
            msg.addBodyPart(attachment);
        }

          msg.send (new java.net.URL(m_hostURL), URI, envelope);
```

SOAP z załącznikami 3/3

W przesyłanym dokumencie znajduje się element z identyfikatorem załącznika np:

<attachment href=”cid:theattachment”/>

Kod odbiornika obsługujący załącznik:

```
public void PurchaseOrderWithAttachment(Envelope requestEnvelope,
                                        SOAPContext requestContext,
                                        SOAPContext responseContext)
    Element attachmentEl =
    (Element)  el.getElementsByTagName("attachment").item(0);

  cid = attachmentEl.getAttribute("href").substring(4);

  MimeBodyPart attachment = requestContext.getBodyPart(cid);
```

SOAP-RPC

- Definiuje model reprezentujący wywołania zdalne RPC
- Jest niezależny od używanego protokołu
- W protokole SOAP-RPC treść koperty zawiera nazwę metody i parametry np:

```
<soap-env:Body>
   <ns1:getPrice xmlns:ns1="urn:xmethods-BNPriceCheck">
      <isbn xsi:type="xsd:string"> 12345</isbn>
   </ns1:getPrice>
</soap-env:Body>
```

Typy danych w SOAP-RPC

- typy proste: int, float, bool, string
- typy złożone: tablice, struktury

Nadajnik SOAP-RPC 1/2

```
public static float sendSoapRPCMessage (String url, String isbn)
                        throws Exception   {

  org.apache.soap.rpc.Call call = new org.apache.soap.rpc.Call ();

 String encodingStyleURI =
   org.apache.soap.Constants.NS_URI_SOAP_ENC;

  call.setEncodingStyleURI(encodingStyleURI);

  call.setTargetObjectURI ("urn:xmethods-BNPriceCheck");

  call.setMethodName ("getPrice");

 Vector params = new Vector ();
  params.addElement (new org.apache.soap.rpc.Parameter("isbn",
                                  String.class, isbn, null));
```

Nadajnik SOAP-RPC 2/2

```
call.setParams (params);

org.apache.soap.rpc.Response resp = call.invoke (new java.net.URL
    (url),"");

if (resp.generatedFault ()) {
       org.apache.soap.Fault fault = resp.getFault();
       System.err.println("Generated fault ");
      return 0;
   } else {
       org.apache.soap.rpc.Parameter result = resp.getReturnValue ();
       Float FL = (Float) result.getValue();
       return FL.floatValue();
   }
```

SOAP-RPC na serwerze

Usługę na serwerze opisuje deskryptor wdrożenia:

```
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment"
        id="urn:stock-onhand">
  <isd:provider type="java"
            scope="Application"
            methods="getPrice">
    <isd:java class=„Bank"/>
  </isd:provider>

    <isd:faultListener>org.apache.soap.server.DOMFaultListener</isd:f
    aultListener>
</isd:service>
```

Obsługa błędów w SOAP

- Zwracane przez SOAP błędy są obsługiwane za pomocą wyspecjalizowanej koperty SOAP.

```
<soap-env:Body>
   <soap-env:Fault>
        <faultcode>  Server </faultcode>
       <faultstring> … </faultstring>
       <faultactor>  … </faultactor>
        <detali>
            <stacTrace> … </stacTrace>
        </detail>
   </soap-env:Fault>
</soap-env:Body>
```

Wartości błędów zwracanych przez SOAP

- **VersionMismatch** (nastąpił konflikt wersji, zgodność wersji determinuje przestrzeń nazw koperty SOAP)
- **MustUnderstand** (podelementowi elementu Header nadano taki atrybut=‘true’, a procesor SOAP tego nie rozumie)
- **DTDNotSuported** (koperta nie może zawierać definicji DTD)
- **DataEncodingUnknown** (nieznana wartość atrybutu encodingStyle)
- **Client** (błąd po stronie klienta) **Sender** w SOAP 1.2
- **Server** (błąd po stronie serwera) **Receiver** w SOAP 1.2

PODSUMOWANIE

- SOAP jest prostym uniwersalnym protokołem opartym o XML.
- Stosuje się go do wymiany dokumentów i zdalnych wywołań RPC.
- Jest podstawowym elementem Web Services.
- Jedni uważają, że zaspokaja większość potrzeb, inni że nie zaspokaja nawet minimum.

Materiały

- *David A. Chappell, Tyler Jewell*
  „JAVA. Usługi sieciowe” O’Reilly 2002
- *http://www.w3.org/TR/SOAP*
- *http://java.sun.com/webservices/*

Dziękuję za uwagę.