

SOAP

Kayartaya Vinod

SOAP

- Simple Object Access Protocol (SOAP 1.1 and just SOAP in SOAP 1.2)
- W3C recommendation for message encoding.
- Provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized distributed environment.
- All SOAP messages are encoded in XML.
- The protocol is independent of runtime environments, so it works well in a heterogeneous environment for data exchange.

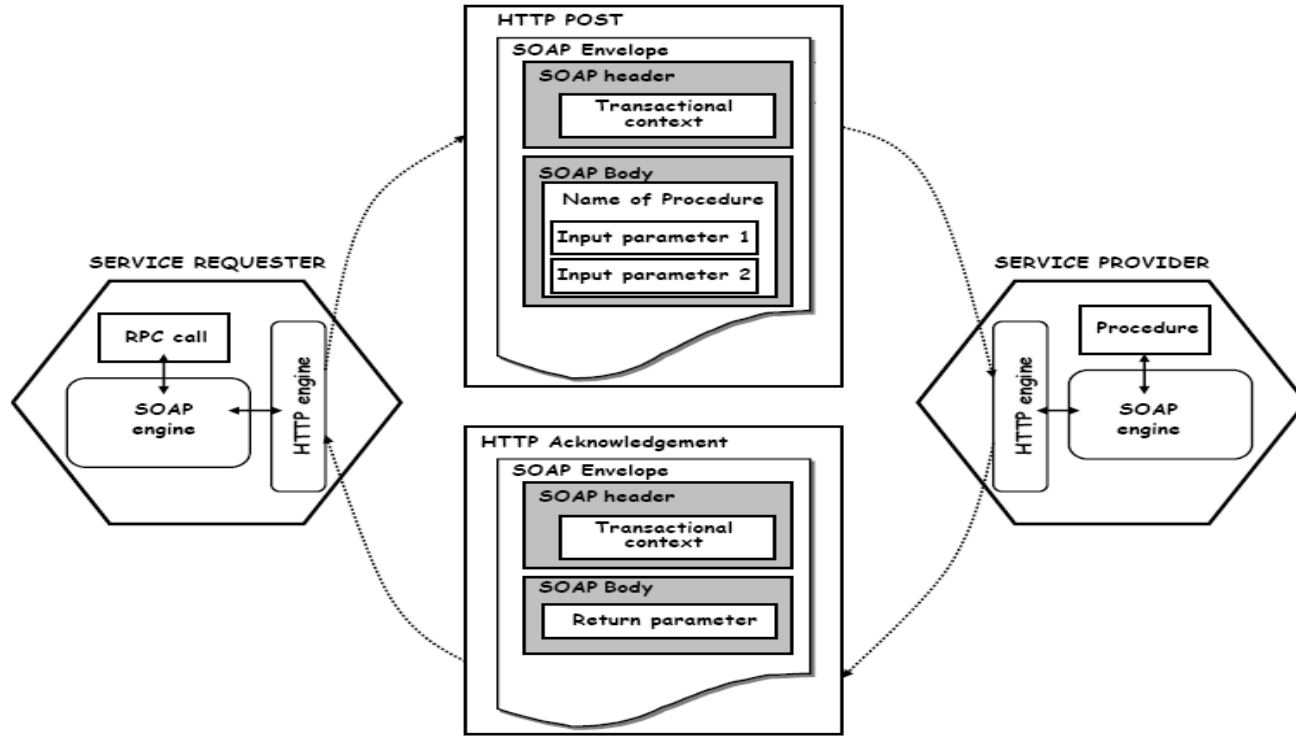
SOAP

- SOAP covers the following four main areas:
 - A message format for one-way communication describing how a message can be packed into an XML document
 - A description of how (the XML document that makes up) a SOAP message should be transported through the Web (using HTTP) or e-mail (using SMTP).

SOAP

- A set of rules that must be followed when processing a SOAP message and a simple classification of the entities involved in that processing. It also specifies what parts of the messages should be read by whom and how to react in case the content is not understood.
- A set of conventions on how to turn a RPC call into a SOAP message and back as well as how to implement the RPC style of interaction (how the client side RPC call is translated into a SOAP message, forwarded, turned into a server side RPC call, the reply converted into a SOAP message and returned to the client)

SOAP IN ACTION: ALL TOGETHER



SOAP MESSAGES

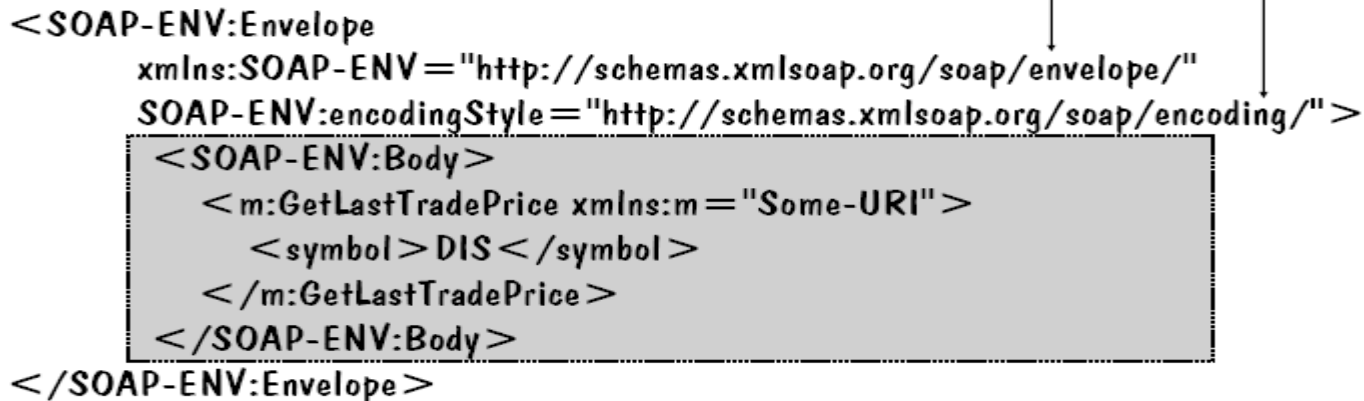
- SOAP is based on message exchanges
- Messages are seen as envelopes where the application encloses the data to be sent
- A message has two main parts; header and body, which both can be divided into blocks
- SOAP does not specify what to do the body, it only states that the header is optional and the body is mandatory
- The use of header and body, however, is implicit.
 - The body is for application level data. The header is for infrastructure level data (security, transaction, etc)

FOR XML FANS (SOAP, BODY ONLY)

XML name space identifier for SOAP serialization

XML name space identifier for SOAP envelope

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" >
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A diagram showing XML namespace identifiers pointing to specific parts of the SOAP XML structure. Two lines from the text 'XML name space identifier for SOAP serialization' point to the 'xmlns:SOAP-ENV' attribute and the 'SOAP-ENV:encodingStyle' attribute. Two lines from the text 'XML name space identifier for SOAP envelope' point to the 'SOAP-ENV:Envelope' element and the 'SOAP-ENV:Body' element. The 'SOAP-ENV:Body' element and its contents are highlighted with a grey background.

SOAP EXAMPLE, HEADER AND BODY

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Header>
    <t:Transaction
      xmlns:t="some-URI"
      SOAP-ENV:mustUnderstand="1">
      5
    </t:Transaction>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DEF</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

THE SOAP HEADER

- The header is intended as a generic place holder for information that is not necessarily application dependent (the application may not even be aware that a header was attached to the message).
- Typical uses of the header are: coordination information, identifiers (for, e.g., transactions) and security information (e.g., certificates)

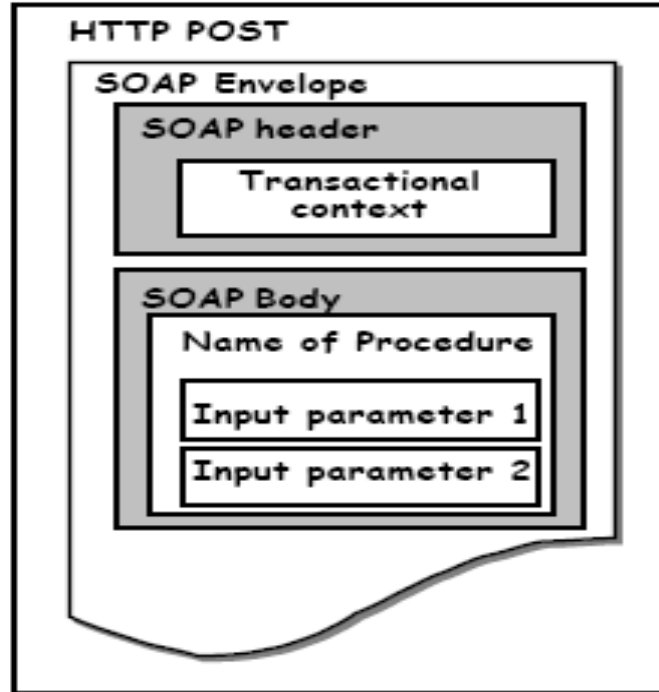
THE SOAP BODY

- The body is intended for the application specific data contained in the message
- Typically, it contains information about the name of the operation, parameter values

SOAP AND HTTP

- A binding of SOAP to a transport protocol is a description of how a SOAP message is to be sent using that transport protocol
- The typical binding for SOAP is HTTP
- SOAP uses the same error and status codes as those used in HTTP so that HTTP responses can be directly interpreted by a SOAP module

SOAP AND HTTP



HTTP REQUEST (IN XML)

POST /StockQuote HTTP/1.1

Host: www.stockquoteserver.com

Content-Type: text/xml; charset="utf-8"

Content-Length: nnnn

SOAPAction: "Some-URI"

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

HTTP RESPONSE (IN XML)

HTTP/1.1 200 OK

Content-Type: text/xml; charset="utf-8"

Content-Length: nnnn

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
  <SOAP-ENV:Body>
    <m:GetLastTradePriceResponse xmlns:m="Some-URI">
      <Price>34.5</Price>
    </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Questions?

Messaging Clients

Using JAXM and SAAJ

MESSAGEFACTORY

- JAXM implementation
 - Represents an object to create SOAP-message objects
- `MessageFactory mf=MessageFactory.newInstance();`
- Creates an instance based on the implementation (eg., Weblogic, Apache)

SOAPMESSAGE

- Represents the entire SOAP Message

```
SOAPMessage request=mf.createMessage();
```

- Creates a blank SOAP message.
- Contains :
 - a SOAP-part
 - an Attachment-part

SOAPPART AND SOAPENVELOPE

- A logical container for an envelope

```
SOAPPart part=request.getSOAPPart();
```

```
SOAPEnvelope env=part.getEnvelope();
```

- Envelope has an optional header and a mandatory body

SOAPBody

```
SOAPBody body=env.getBody();
```

- A placeholder for the actual SOAP message
 - Can contain an RPC (in XML format)
 - Or an XML document (representing info)

CONSTRUCTING RPC ELEMENTS

```
Name name = env.createName("toDollor");  
SOAPElement elem = body.addChildElement(name);  
elem.addNamespaceDeclaration(  
    "m", "http://kvinod.com");
```

CONSTRUCTING RPC ELEMENTS

```
SOAPElement param =  
    elem.addChildElement("doubleValue");
```

```
param.addTextNode("123").addAttribute(  
    env.createName("xsi:type"), "xsd:double");
```

SOAPCONNECTION

- Now that the message is ready, create a connection object to send the “request” over the net using HTTP (or any other protocol)

```
SOAPConnectionFactory factory=  
    SOAPConnectionFactory.newInstance();
```

```
SOAPConnection conn = factory1.createConnection();
```

THE ACTUAL CALL

```
SOAPMessage response = conn.call(msg,  
"http://localhost:9999/myapp/cs");
```

```
response.writeTo(System.out);
```