SOAP Overview (Based on SOAP 1.2)

Sang Shin
Michèle Garoche
www.javapassion.com
"Learn with Passion!"

Agenda

- What is and is not SOAP?
- SOAP Message Structure
- SOAP Terminology
- SOAP Message Exchange
- Document vs. RPC
- SOAP Encoding
- Quick overview of Java API for SOAP
- Changes from SOAP 1.1 to SOAP 1.2

What is SOAP?
What is not SOAP?
Where is SOAP?

What is SOAP? W3C Definition

- SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment
- SOAP uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols
- The framework has been designed to be independent of any particular programming model and other implementation specific semantics

What is SOAP?

- Simple Object Access Protocol
- Wire protocol similar to
  - IIOP for CORBA
  - JRMP for RMI
- XML is used for data encoding
  - “text” based protocol vs. “binary” protocol
- Supports XML-based RPC (Remote Procedure Call)
  - To ease the migration of existing RPC distribution schemes to SOAP based system

Do I Need to know how SOAP works in detail as a Java Developer?

- Yes
  - Understanding it will help you to build better application
  - Ex) Understanding how TCP/IP will help you build better TCP/IP application
- No
  - You will mostly likely use high-level API (JAX-WS) to build Web applications
  - How SOAP works is hidden from developers

What is SOAP?

- Stateless
- One-way message exchange paradigm
  - Applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information
- Silent on the semantics of any application-specific data it conveys

What SOAP is Not

- Not a component model
  - So it will not replace objects and components, i.e. EJB™, JavaBeans™
- Not a programming language
  - So it will not replace Java
- Not a solution for all
  - So it will not replace other distributed computing technologies such as RMI

SOAP Design Goals

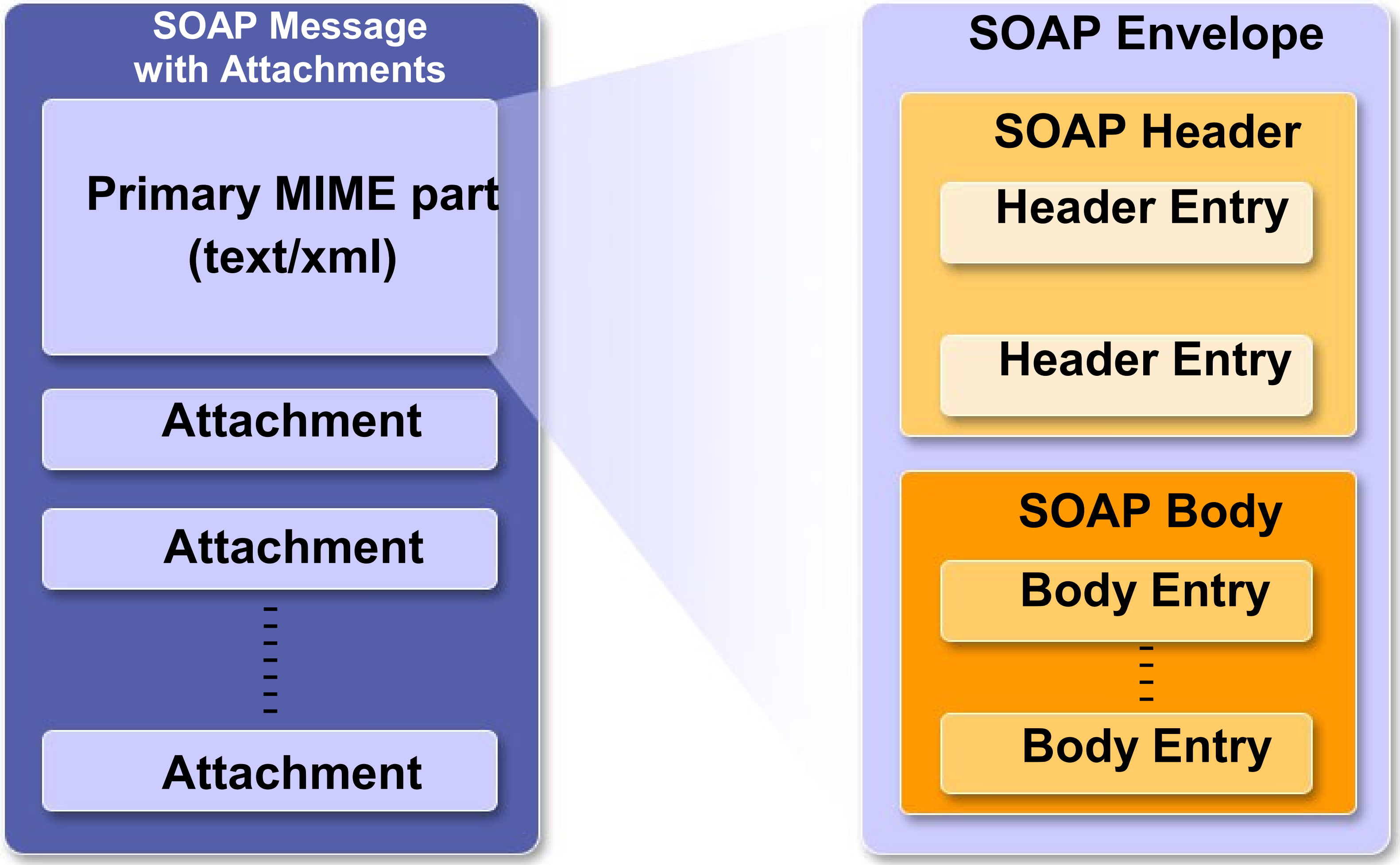
- Simplicity
- Extensibility
  - New standards define new semantics
- Features not supported (by design)
  - Distributed garbage collection
  - Object by reference
  - Activation
  - Message batching

Where is SAOP?

- SOAP 1.2 is W3C recommendation
- SOAP 1.2 Part 1 defines
  - SOAP envelope
  - Protocol binding framework
- SOAP 1.2 Part 2 defines
  - Data model for SOAP
  - Binding to HTTP

Message Structure

SOAP Message Structure



SOAP Message Envelope

- Embedded Information
  - Namespaces
  - Encoding information
- Header
  - Optional
  - Can be handled by intermediaries
- Body
  - Mandatory
  - Handled only by ultimate receiver

SOAP Header (<env:Header>)

- Used for extension
  - Context
  - Authentication
  - Transaction
  - Management
  - Many other system level semantics
- Made of Header blocks (Header entries)
- Most Web services standard activities are basically defining standard header entries for a particular domain, i.e. security

SOAP Header Blocks (Entries)

- Child elements of SOAP Header
- Designed in anticipation of various uses for SOAP by SOAP intermediaries
  - Can be individually targeted at SOAP nodes
  - Allows SOAP intermediaries to provide value-added services
- May be inspected, inserted, deleted or forwarded by SOAP nodes encountered along a SOAP message path

SOAP Body (<env:Body>)

- Made of Body blocks (Body entries)
- Consumed by Ultimate SOAP receiver
- Carry end-to-end information
  - Application data (XML document) (*document* style)
  - RPC method and parameters (*rpc* style)
  - SOAP fault

Fault Message

SOAP Fault (<env:Fault>)

- Used to carry error and/or status information
- Four sub-elements
  - *faultcode*
  - *faultstring*
  - *faultactor*
  - *detail*

Pre-defined SOAP faultcode values

- VersionMismatch
  - Invalid namespace in SOAP envelope
- MustUnderstand
  - Receiver mode cannot handle *mustUnderstand* SOAP header block
- Client
  - Indicates client side error
- Server
  - Indicates server side error

SOAP Fault Example: Cause

```
<env:Envelope
        xmlns:env='http://www.w3.org/2001/06/soap-envelope'>
        <env:Header>
              <abc:Extension1
                      xmlns:abc='http://example.org/2001/06/ext'
                      env:mustUnderstand='1' />
              <def:Extension2
                      xmlns:def='http://example.com/stuff'
                      env:mustUnderstand='1' />
        </env:Header>
        <env:Body>
            . . .
        </env:Body>
</env:Envelope>
```

SOAP Fault Example: Result

```
<env:Envelope xmlns:env='http://www.w3.org/2001/06/soap-envelope'
                    xmlns:f='http://www.w3.org/2001/06/soap-faults' >
        <env:Header>
             <f:Misunderstood qname='abc:Extension1'
                             xmlns:abc='http://example.org/2001/06/ext'/>
             <f:Misunderstood qname='def:Extension2'
                              xmlns:def='http://example.com/stuff'/>
        </env:Header>
        <env:Body>
              <env:Fault>
                    <faultcode>MustUnderstand</faultcode>
                    <faultstring>
                          One or more mandatory headers not understood
                    </faultstring>
              </env:Fault>
        </env:Body>
</env:Envelope>
```

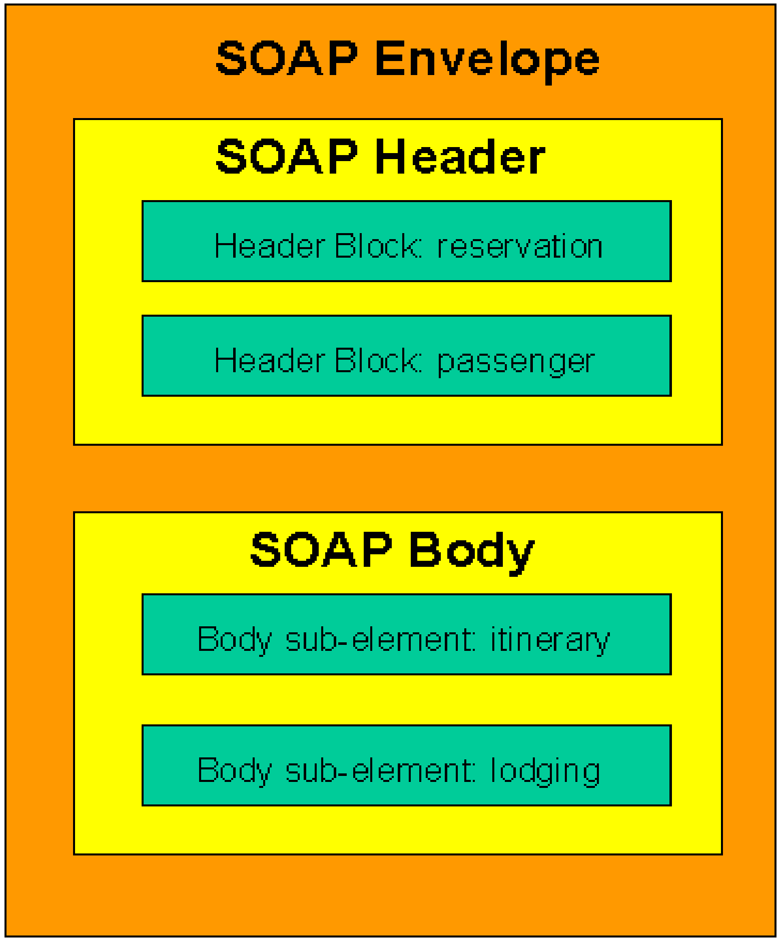
Where do you put your data, in Header block or Body block?

Header Block or Body Block?

- Decisions made at the time of application design
- Header blocks may be targeted at various nodes that might be encountered along a message's path from a sender to the ultimate recipient
  - Intermediate SOAP nodes may provide value-added services based on data in such headers

Example SOAP Messages

SOAP Message Example: Travel Reservation


SOAP Envelope
SOAP Header
Header Block: reservation
Header Block: passenger
SOAP Body
Body sub-element: itinerary
Body sub-element: lodging

Example1: SOAP Message Travel Reservation Request (page 1)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">

 <env:Header>

  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>

  <n:passenger xmlns:n="http://mycompany.example.com/employees"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>

 </env:Header>
```

Example1: SOAP Message Travel Reservation Request (page 2)

```
<env:Body>
 <p:itinerary
  xmlns:p="http://travelcompany.example.org/reservation/travel">
  <p:departure>
    <p:departing>New York</p:departing>
    <p:arriving>Los Angeles</p:arriving>
    <p:departureDate>2001-12-14</p:departureDate>
    <p:departureTime>late afternoon</p:departureTime>
    <p:seatPreference>aisle</p:seatPreference>
  </p:departure>
  <p:return>
    <p:departing>Los Angeles</p:departing>
    <p:arriving>New York</p:arriving>
    <p:departureDate>2001-12-20</p:departureDate>
    <p:departureTime>mid-morning</p:departureTime>
    <p:seatPreference/>
  </p:return>
 </p:itinerary>
 <q:lodging
  xmlns:q="http://travelcompany.example.org/reservation/hotels">
  <q:preference>none</q:preference>
 </q:lodging>
 </env:Body>
</env:Envelope>
```

Quick Namespace Tutorial

XML Namespaces Tutorial

- Used to avoid name collision
- Facilitates grouping of elements
  - i.e.: SOAP application knows which elements belong to which namespace
- Can be used as version control scheme
- Syntax
  - Namespace declaration
  - Elements and attributes

XML Namespaces Declaration

- A prefix is associated with URI
- The association is defined as an attribute within an element
  - *xmlns*:prefix
- *xmlns* is Namespaces keyword, prefix is user- defined

```
<classes xmlns:XMLclass=“ http://www.brandeis.edu/rseg-0151-g”>
      <XMLclass:syllabus>
           ...
      </XMLclass:syllabus>
</ classes>
```

SOAP Namespaces Example

```
<env:Envelope xmlns:env="http://www.w3.org/2001/06/soap-envelope" >
       <env:Body>
             <m:GetLastTradePrice
                  env:encodingStyle="http://www.w3.org/2001/06/soap-encoding"
                  xmlns:m="http://example.org/2001/06/quotes">
                         <symbol>DIS</symbol>
              </m:GetLastTradePrice>
        </env:Body>
 </env:Envelope>
```

- env namespace is defined in SOAP
- m namespace is custom namespace

SOAP 1.1 Namespaces URI's

- Envelope
  - http://www.w3.org/2001/06/soap-envelope
  - Used for "version mismatch" check
- Serialization
  - http://www.w3.org/2001/06/soap-encoding
- mustUnderstand fault
  - http://www.w3.org/2001/06/soap-faults
- Upgrade
  - http://www.w3.org/2001/06/soap-upgrade

SOAP Terminology

Protocol Concepts

- SOAP node
- SOAP role
- SOAP binding
- SOAP feature
  - An extension of the SOAP messaging framework: reliability, security, correlation
- SOAP module
  - Realization of SOAP features
- SOAP message exchange pattern
- SOAP application

Data Encapsulation Concepts

- SOAP message
- SOAP envelope
- SOAP header
- SOAP header block
- SOAP body
- SOAP fault

Message Sender & Receiver Concepts

- SOAP sender
- SOAP receiver
- SOAP message path
- Initial SOAP sender
- SOAP intermediary
- Ultimate SOAP receiver

SOAP Message Exchange

SOAP Exchange Model

- SOAP is a simple messaging framework for transferring information specified in the form of an XML infoset between an initial SOAP sender and an ultimate SOAP receiver
- The more interesting scenarios typically involve multiple message exchanges between these two nodes
  - request and response pattern

Request-Response Pattern

- Conversational message exchange
  - used to exchange XML documents
  - can be multiple message exchange pattern
- RPC (Remote Procedure Call)
  - used when there is a need to model a certain programmatic behavior

Conversational Message Exchange

Example2: SOAP Message Response: Travel Reservation (page 1)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation
xmlns:m="http://travelcompany.example.org/reservation"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
    <m:reference>uuid:093a2da1-q345-739r-ba5d-
pqff98fe8j7d</m:reference>
    <m:dateAndTime>2001-11-29T13:35:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
    <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
```

Reference to request message

Example2:SOAP Message Response Travel Reservation (page 2) - Choices of Airport

```
<env:Body>
 <p:itineraryClarification
  xmlns:p="http://travelcompany.example.org/reservation/travel">
  <p:departure>
   <p:departing>
    <p:airportChoices>
      JFK LGA EWR
    </p:airportChoices>
   </p:departing>
  </p:departure>
  <p:return>
   <p:arriving>
    <p:airportChoices>
     JFK LGA EWR
    </p:airportChoices>
   </p:arriving>
  </p:return>
 </p:itineraryClarification>
 </env:Body>
</env:Envelope>
```

Application defined schema

Example3: SOAP Message Travel Reservation (page 1) Request

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation
     xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
    <m:reference>uuid:093a2da1-q345-739r-ba5d-
pqff98fe8j7d</m:reference>
    <m:dateAndTime>2001-11-29T13:36:50.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
```

Example3: SOAP Message Travel Reservation Reuqest (page 2) - Selection of Airport

```
<env:Body>
  <p:itinerary
   xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>LGA</p:departing>
   </p:departure>
   <p:return>
     <p:arriving>EWR</p:arriving>
   </p:return>
  </p:itinerary>
 </env:Body>
</env:Envelope>
```

SOAP with Attachments

SOAP 1.1 With Attachments

- Submitted to W3C for basis of XMLP
  - http://www.w3.org/TR/SOAP-attachments
- Uses MIME “multipart/related” as a container for:
  - SOAP envelope
  - Arbitrary “attachments”
- SOAP envelope and payload can reference “attachments” via relative URLs (href) in the SOAP envelope

Changes from SOAP 1.1 to SOAP 1.2

Changes: Document Structure

- Document structure
  - SOAP 1.2 has been rewritten in terms of XML infosets, and not as serializations of the form <?xml....?> required by SOAP 1.1

Changes: Additional or Changed Syntax (page 1)

- In the SOAP 1.2 infoset-based description, the env:mustUnderstand attribute in header elements takes the (logical) value "true" or "false" (instead of 1 or 0)
- SOAP 1.2 provides a new fault code DataEncodingUnknown.
- The various namespaces defined by the two protocols are of course different.
- SOAP 1.2 replaces the attribute env:actor with env:role but with essentially the same semantics.

Changes: Additional or Changed Syntax (page 2)

- SOAP 1.2 defines a new attribute, env:relay, for header blocks to indicate if unprocessed header blocks should be forwarded
- SOAP 1.2 defines two new roles, "none" and "ultimateReceiver", together with a more detailed processing model on how these behave
- SOAP 1.2 replaces "client" and "server" fault codes with "Sender" and "Receiver"

Version Handling

- SOAP 1.1 node receiving 1.2 message
  - generates SOAP version mismatch SOAP fault
- SOAP 1.2 node receiving 1.1 message
  - may process the message as a SOAP/1.1 message (if supported)
  - generate a version mismatch SOAP fault based on a SOAP/1.1 message construct
  - SOAP fault SHOULD include an Upgrade SOAP header block

Example: SOAP Version 1.2 node generating a SOAP/1.1 version mismatch fault message

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header>
  <env:Upgrade>
   <env:SupportedEnvelope qname="ns1:Envelope"
           xmlns:ns1="http://www.w3.org/2003/05/soap-envelope"/>
   </env:Upgrade>
  </env:Header>
  <env:Body>
   <env:Fault>
    <faultcode>env:VersionMismatch</faultcode>
    <faultstring>Version Mismatch</faultstring>
   </env:Fault>
 </env:Body>
</env:Envelope>
```

Thank you!

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