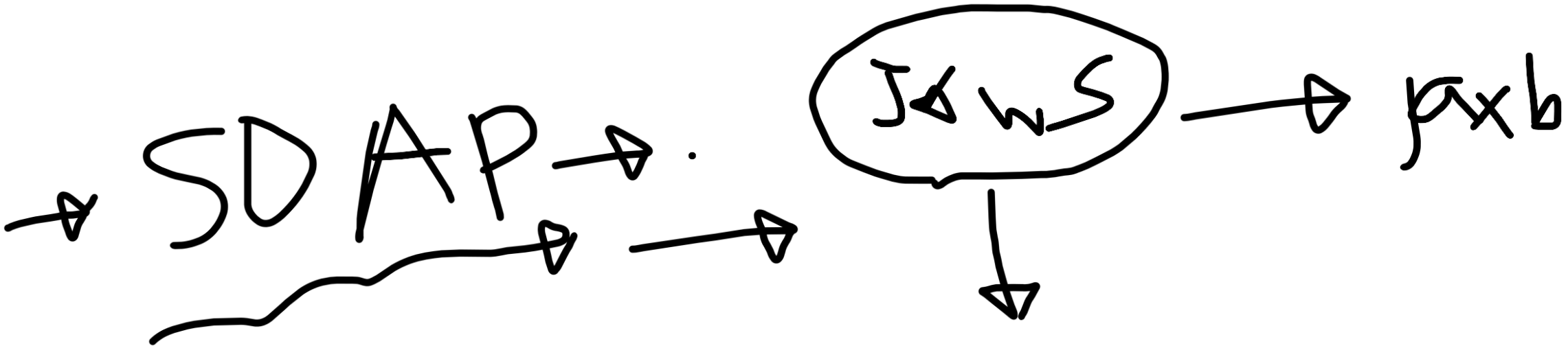
SOAP Web Service


SOAP
JAWS
JAXB

SOAP

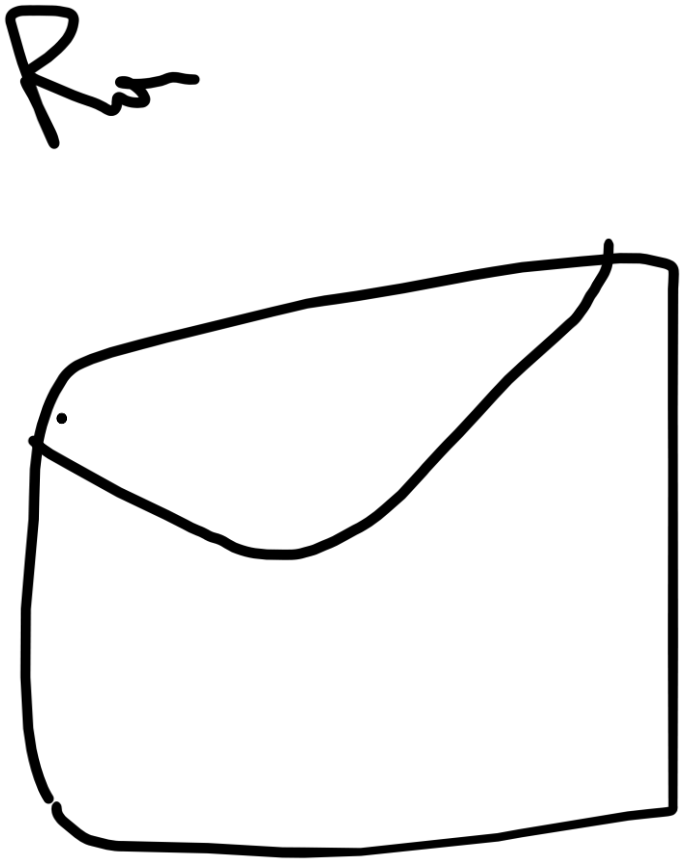
- SOAP is an acronym for Simple Object Access Protocol.

SOAP

- SOAP is a communication protocol designed to communicate via Internet.
- SOAP can extend HTTP for XML messaging.
- SOAP provides data transport for **Web services**.
- SOAP can exchange complete **documents** or call a **remote procedure**.
- SOAP can be used for **broadcasting** a message.
- SOAP is **platform-** and **language-independent**.
- SOAP is the **XML** way of defining what information is sent and how.
- SOAP enables **client** applications to easily connect to **remote services** and invoke **remote methods**.

SOAP containing the following elements

- Envelope
- Header
- Body
- Fault



SOAP envelope indicates the start and the end of the message

- Every SOAP message has a **root** Envelope element.
- Envelope is a **mandatory** part of SOAP message.
- Every Envelope element must contain exactly **one Body** element.
- If an Envelope contains a Header element, it must contain **no more than one**, and it must appear as the **first child** of the Envelope, before the Body.
- The envelope changes when SOAP versions change.
- The SOAP envelope is specified using the ***ENV*** namespace **prefix** and the Envelope element.

v1.2-Compliant SOAP Message

```
<?xml version = "1.0"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
  SOAP-ENV:encodingStyle = " http://www.w3.org/2001/12/soap-encoding">
  ...
  Message information goes here
  ...
</SOAP-ENV:Envelope>
```

SOAP with HTTP POST

```
POST /OrderEntry HTTP/1.1
Host: www.example.com
Content-Type: application/soap;  charset="utf-8"
Content-Length: nnnn

<?xml version = "1.0"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
  SOAP-ENV:encodingStyle = " http://www.w3.org/2001/12/soap-encoding">
  ...
  Message information goes here
  ...
</SOAP-ENV:Envelope>
```

SOAP Header

```
<?xml version = "1.0"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV = " http://www.w3.org/2001/12/soap-envelope"
  SOAP-ENV:encodingStyle = " http://www.w3.org/2001/12/soap-encoding">

  <SOAP-ENV:Header>
    <t:Transaction
      xmlns:t = "http://www.example.com/transaction/"
      SOAP-ENV:mustUnderstand = "true">5
    </t:Transaction>
  </SOAP-ENV:Header>
  ...
  ...
</SOAP-ENV:Envelope>
```

SOAP Header

- It is an optional part of a SOAP message.
- Header elements can occur multiple times.
- Headers are intended to add new features and functionality.
- The SOAP header contains header entries defined in a **namespace**.
- The header is encoded as the first immediate child element of the SOAP envelope.
- When multiple headers are defined, all **immediate child elements** of the SOAP header are interpreted as **SOAP header blocks**.

SOAP - Body

- The body contains mandatory information intended for the ultimate receiver of the message.

```
<?xml version = "1.0"?>
<SOAP-ENV:Envelope>
  ........
  <SOAP-ENV:Body>
    <m:GetQuotation xmlns:m = "http://www.example.com/Quotation">
      <m:Item>Computers</m:Item>
    </m:GetQuotation>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP - Fault

If an error occurs during processing, the response to a SOAP message is a SOAP fault element in the body of the message, and the fault is returned to the sender of the SOAP message.

- The SOAP fault mechanism returns specific information about the error, including a predefined **code**, a **description**, and the **address** of the SOAP processor that generated the fault.

SOAP FAULT

- A SOAP message can carry only one fault block.
- Fault is an optional part of a SOAP message.
- For HTTP binding, a successful response is linked to the 200 to 299 range of status codes.
- SOAP Fault is linked to the 500 to 599 range of status codes.

Sub-elements of Fault

Sr.No	Sub-element & Description
1	<faultCode> It is a text code used to indicate a class of errors. See the next Table for a listing of predefined fault codes.
2	<faultString> It is a text message explaining the error.
3	<faultActor> It is a text string indicating who caused the fault. It is useful if the SOAP message travels through several nodes in the SOAP message path, and the client needs to know which node caused the error. A node that does not act as the ultimate destination must include a faultActor element.
4	<detail> It is an element used to carry application-specific error messages. The detail element can contain child elements called detail entries.

SOAP Fault Codes

- SOAP-ENV:VersionMismatch
- SOAP-ENV:MustUnderstand
- SOAP-ENV:Client
- SOAP-ENV:Server

SOAP Fault Example

- The following code is a sample Fault. The client has requested a method named *ValidateCreditCard*, but the service does not support such a method. This represents a client request error, and the server returns the following SOAP response

SOAP Fault Example

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi = "http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd = "http://www.w3.org/1999/XMLSchema">

  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode xsi:type = "xsd:string">SOAP-ENV:Client</faultcode>
      <faultstring xsi:type = "xsd:string">
        Failed to locate method (ValidateCreditCard) in class (examplesCreditCard) at
          /usr/local/ActivePerl-5.6/lib/site_perl/5.6.0/SOAP/Lite.pm line 1555.
      </faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Encoding

- SOAP includes a built-in set of rules for encoding data types. It enables the SOAP message to indicate specific data types, such as integers, floats, doubles, or arrays.

SOAP Encoding

- SOAP data types are divided into two broad categories – scalar types and compound types.
  - Scalar types contain exactly one value such as a last name, price, or product description.
  - Compound types contain multiple values such as a purchase order or a list of stock quotes.
  - Compound types are further subdivided into arrays and structs.

SOAP Encoding

- The encoding style for a SOAP message is set via the *SOAP-ENV:encodingStyle* attribute.
- To use SOAP 1.1 encoding, use the value http://schemas.xmlsoap.org/soap/encoding/
- To use SOAP 1.2 encoding, use the value http://www.w3.org/2001/12/soap-encoding

SOAP Encoding

- Latest SOAP specification adopts all the built-in types defined by XML Schema. Still, SOAP maintains its own convention for defining constructs not standardized by XML Schema, such as arrays and references.

Scalar Types

- For scalar types, SOAP adopts all the built-in simple types specified by the XML Schema specification. This includes strings, floats, doubles, and integers.

Scalar Types

Simple Types Built-In to XML Schema	
Simple Type	**Example(s)**
string	Confirm this is electric.
boolean	true, false, 1, 0.
float	-INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN.
double	-INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN.
decimal	-1.23, 0, 123.4, 1000.00.
binary	100010
integer	-126789, -1, 0, 1, 126789.
nonPositiveInteger	-126789, -1, 0.
negativeInteger	-126789, -1.
long	-1, 12678967543233
int	-1, 126789675
short	-1, 12678
byte	-1, 126
nonNegativeInteger	0, 1, 126789
unsignedLong	0, 12678967543233
unsignedInt	0, 1267896754
unsignedShort	0, 12678
unsignedByte	0, 126
positiveInteger	1, 126789.
date	1999-05-31, ---05.
time	13:20:00.000, 13:20:00.000-05:00

SOAP response with a double data type

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <SOAP-ENV:Body>
      <ns1:getPriceResponse
         xmlns:ns1 = "urn:examples:priceservice"
         SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding">
         <return xsi:type = "xsd:double">54.99</return>
      </ns1:getPriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Compound Types

- SOAP arrays have a very specific set of rules, which require that you specify both the element type and array size. SOAP also supports multidimensional arrays, but not all SOAP implementations support multidimensional functionality.
- To create an array, you must specify it as an *xsi:type* of array. The array must also include an *arrayType* attribute. This attribute is required to specify the data type for the contained elements and the dimension(s) of the array.

Compound Types

- For example, the following attribute specifies an array of 10 double values −
- arrayType = "xsd:double[10]“

- In contrast, the following attribute specifies a two-dimensional array of strings −

- arrayType = "xsd:string[5,5]"

Sample SOAP response with an array of double values

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <SOAP-ENV:Body>
      <ns1:getPriceListResponse
         xmlns:ns1 = "urn:examples:pricelistservice"
         SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding">

         <return xmlns:ns2 = "http://www.w3.org/2001/09/soap-encoding"
            xsi:type = "ns2:Array" ns2:arrayType = "xsd:double[2]">
            <item xsi:type = "xsd:double">54.99</item>
            <item xsi:type = "xsd:double">19.99</item>
         </return>
      </ns1:getPriceListResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- Structs contain multiple values, but each element is specified with a unique accessor element.

- **For example**, consider an item within a **product catalog**. In this case, the struct might contain a product **SKU**, **product name**, **description**, and **price**. Here is how such a struct would be represented in a SOAP message −

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <SOAP-ENV:Body>
      <ns1:getProductResponse
         xmlns:ns1 = "urn:examples:productservice"
         SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding">

         <return xmlns:ns2 = "urn:examples" xsi:type = "ns2:product">
            <name xsi:type = "xsd:string">Red Hat Linux</name>
            <price xsi:type = "xsd:double">54.99</price>
            <description xsi:type = "xsd:string">
               Red Hat Linux Operating System
            </description>
            <SKU xsi:type = "xsd:string">A358185</SKU>
         </return>
      </ns1:getProductResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Transport

- SOAP is not tied to any transport protocol. SOAP can be transported via SMTP, FTP, IBM's MQSeries, or Microsoft Message Queuing (MSMQ).

- SOAP specification includes details on HTTP only. HTTP remains the most popular SOAP transport protocol.

SOAP via HTTP

- Quite logically, SOAP requests are sent via an HTTP request and SOAP responses are returned within the content of the HTTP response. While SOAP requests can be sent via an HTTP GET, the specification includes details on HTTP POST only.
- Additionally, both HTTP requests and responses are required to set their content type to text/xml.

SOAP via HTTP

- The SOAP specification mandates that the client must provide a SOAPAction header, but the actual value of the SOAPAction header is dependent on the SOAP server implementation.

- For example, to access the AltaVista BabelFish Translation service, hosted by XMethods, you must specify the following as a SOAPAction header.

- urn:xmethodsBabelFish#BabelFish

Here is a sample request sent via HTTP to the XMethods Babelfish Translation service –

```
POST /perl/soaplite.cgi HTTP/1.0
Host: services.xmethods.com
Content-Type: text/xml; charset = utf-8
Content-Length: 538
SOAPAction: "urn:xmethodsBabelFish#BabelFish"

<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi = "http://www.w3.org/1999/XMLSchema-instance"
   xmlns:xsd = "http://www.w3.org/1999/XMLSchema">

   <SOAP-ENV:Body>
      <ns1:BabelFish
         xmlns:ns1 = "urn:xmethodsBabelFish"
         SOAP-ENV:encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/">
         <translationmode xsi:type = "xsd:string">en_fr</translationmode>
         <sourcedata xsi:type = "xsd:string">Hello, world!</sourcedata>
      </ns1:BabelFish>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- Note the content type and the SOAPAction header. Also note that the BabelFish method requires two String parameters. The translation mode en_fr translates from English to French.
- Here is the response from XMethods

```
HTTP/1.1 200 OK
Date: Sat, 09 Jun 2001 15:01:55 GMT
Server: Apache/1.3.14 (Unix) tomcat/1.0 PHP/4.0.1pl2
SOAPServer: SOAP::Lite/Perl/0.50
Cache-Control: s-maxage = 60, proxy-revalidate
Content-Length: 539
Content-Type: text/xml

<?xml version = "1.0" encoding = "UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENC = "http://schemas.xmlsoap.org/soap/encoding/"
   SOAP-ENV:encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsi = "http://www.w3.org/1999/XMLSchema-instance"
   xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd = "http://www.w3.org/1999/XMLSchema">

    <SOAP-ENV:Body>
       <namesp1:BabelFishResponse xmlns:namesp1 = "urn:xmethodsBabelFish">
          <return xsi:type = "xsd:string">Bonjour, monde!</return>
       </namesp1:BabelFishResponse>
    </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

SOAP Examples

- In the example , a *GetQuotation* request is sent to a SOAP Server over HTTP. The request has a *QuotationName* parameter, and a Quotation will be returned in the response.
- The namespace for the function is defined in http://www.xyz.org/quotation address.

SOAP Example – The Request

```
POST /Quotation HTTP/1.0
Host: www.xyz.org
Content-Type: text/xml; charset = utf-8
Content-Length: nnn

<?xml version = "1.0"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
  SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding">

  <SOAP-ENV:Body xmlns:m = "http://www.xyz.org/quotations">
    <m:GetQuotation>
      <m:QuotationsName>MicroSoft</m:QuotationsName>
    </m:GetQuotation>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Example – The Response

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset = utf-8
Content-Length: nnn

<?xml version = "1.0"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
  SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding">

  <SOAP-ENV:Body xmlns:m = "http://www.xyz.org/quotation">
    <m:GetQuotationResponse>
      <m:Quotation>Here is the quotation</m:Quotation>
    </m:GetQuotationResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Standards

- SOAP 1.1 was originally submitted to the W3C in May 2000. Official submitters included large companies such as Microsoft, IBM, and Ariba, and smaller companies such as UserLand Software and DevelopMentor.
- In July 2001, the XML Protocol Working Group released a "working draft" of SOAP 1.2. Within the W3C, this document is officially a work in progress, meaning that the document is likely to be updated many times before it is finalized.

- SOAP Version 1.1 is available online

  at http://www.w3.org/TR/SOAP/

- The working draft of SOAP Version 1.2 is available

  at http://www.w3.org/TR/soap12/

WSDL

WSDL

- WSDL stands for Web Services Description Language. It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM.

Features of WSDL

- WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.
- WSDL definitions describe how to access a web service and what operations it will perform.
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.
- WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'.

WSDL Usage

- WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

History of WSDL

- WSDL 1.1 was submitted as a W3C Note by Ariba, IBM, and Microsoft for describing services for the W3C XML Activity on XML Protocols in March 2001.
- WSDL 1.1 has not been endorsed by the World Wide Web Consortium (W3C), however it has just released a draft for version 2.0 that will be a recommendation (an official standard), and thus endorsed by the W3C.

WSDL Elements

- WSDL breaks down web services into three specific, identifiable elements that can be combined or reused once defined.
- The three major elements of WSDL that can be defined separately are –
  - Types
  - Operations
  - Binding
- A WSDL document has various elements, but they are contained within these three main elements, which can be developed as separate documents and then they can be combined or reused to form complete WSDL files.

WSDL Elements

- **Definition** − It is the root element of all WSDL documents. It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements described here.
- **Data types** − The data types to be used in the messages are in the form of XML schemas.
- **Message** − It is an abstract definition of the data, in the form of a message presented either as an entire document or as arguments to be mapped to a method invocation.
- **Operation** − It is the abstract definition of the operation for a message, such as naming a method, message queue, or business process, that will accept and process the message.

WSDL Elements

- **Port type** − It is an abstract set of operations mapped to one or more end-points, defining the collection of operations for a binding; the collection of operations, as it is abstract, can be mapped to multiple transports through various bindings.
- **Binding** − It is the concrete protocol and data formats for the operations and messages defined for a particular port type.
- **Port** − It is a combination of a binding and a network address, providing the target address of the service communication.
- **Service** − It is a collection of related end-points encompassing the service definitions in the file; the services map the binding to the port and include any extensibility definitions.

helloService.wsdl

```
<definitions name = "HelloService"
   targetNamespace = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns = "http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <message name = "SayHelloRequest">
      <part name = "firstName" type = "xsd:string"/>
   </message>

   <message name = "SayHelloResponse">
      <part name = "greeting" type = "xsd:string"/>
   </message>

   <portType name = "Hello_PortType">
      <operation name = "sayHello">
         <input message = "tns:SayHelloRequest"/>
         <output message = "tns:SayHelloResponse"/>
      </operation>
   </portType>

   <binding name = "Hello_Binding" type = "tns:Hello_PortType">
      <soap:binding style = "rpc"
         transport = "http://schemas.xmlsoap.org/soap/http"/>
      <operation name = "sayHello">
         <soap:operation soapAction = "sayHello"/>
         <input>
```

```
29              <soap:body
30                 encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
31                 namespace = "urn:examples:helloservice"
32                 use = "encoded"/>
33           </input>
34
35           <output>
36              <soap:body
37                 encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
38                 namespace = "urn:examples:helloservice"
39                 use = "encoded"/>
40           </output>
41        </operation>
42     </binding>
43
44     <service name = "Hello_Service">
45        <documentation>WSDL File for HelloService</documentation>
46        <port binding = "tns:Hello_Binding" name = "Hello_Port">
47           <soap:address
48              location = "http://www.examples.com/SayHello/" />
49        </port>
50     </service>
51  </definitions>
```