XML Web Services

Web services are web application components.

A Web service is a method of communication between two electronic devices over a network. It is a software function provided at a network address over the Web with the service always-on.

SOAP

- SOAP stands for Simple Object Access Protocol
- SOAP is an XML based protocol for accessing Web Services.
- SOAP is based on XML
- SOAP is a W3C recommendation

WSDL

- WSDL stands for Web Services Description Language
- WSDL is an XML-based language for describing Web services.
- WSDL is a W3C recommendation

XML WSDL

An WSDL document describes a web service.

An WSDL specifies the location of the service, and the methods of the service, using these major elements:

Element	Description
<types>	Defines the (XML Schema) data types used by the web service
<message>	Defines the data elements for each operation
<portType>	Describes the operations that can be performed and the messages involved.
<binding>	Defines the protocol and data format for each port type

The main structure of a WSDL document looks like this:

```
<definitions>
  <types>
    Data type definitions…
  </types>

  <message>
    Definition of the data being communicated…
  </message>

  <portType>
    Set of operations…
  </portType>

  <binding>
    Protocol and data format specification…
  </binding>
</definitions>
```

This is a simplified fraction of a WSDL document:

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

In this example the <portType> element defines "glossaryTerms" as the name of a port, and "getTerm" as the name of an operation.

The "getTerm" operation has…

- an input message called "getTermRequest";
- an output message called "getTermResponse".

The <message> elements define the parts of each message and the associated data types.

The <portType> Element

The <portType> element defines

1. A web service;
2. The operations that can be performed;
3. The messages that are involved.

The request-response type is the most common operation type, but WSDL defines four types:

Type	Definition
One-way	The operation can receive a message but will not return a response
Request-response	The operation can receive a request and will return a response
Solicit-response	The operation can send a request and will wait for a response
Notification	The operation can send a message but will not wait for a response

A one-way operation example:

```
<message name="newTermValues">
  <part name="term" type="xs:string"/>
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
```

```
  <operation name="setTerm">
    <input name="newTerm" message="newTermValues"/>
  </operation>
</portType >
```

In the example above, the portType "glossaryTerms" defines a one-way operation called "setTerm".

The "setTerm" operation allows input of new glossary terms messages using a "newTermValues" message with the input parameters "term" and "value". However, no output is defined for the operation.

WSDL Binding to SOAP

WSDL bindings define the message format and protocol details for a web service.

A request-response operation example:

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>

<binding type="glossaryTerms" name="b1">
   <soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http" />
   <operation>
     <soap:operation soapAction="http://example.com/getTerm"/>
     <input><soap:body use="literal"/></input>
     <output><soap:body use="literal"/></output>
  </operation>
</binding>
```

The “binding” element has two attributes:

1. Name;
2. Type.

The “name” attribute (you can use any name you want) defines the name of the binding, and the type attribute points to the port for the binding, in this case the "glossaryTerms" port.

The “soap:binding” element has two attributes:

1. Style;

   The “style” attribute can be "rpc" or "document". In this case we use document.

2. Transport.

   The “transport” attribute defines the SOAP protocol to use. In this case we use HTTP.

The “operation” element defines each operation that the portType exposes.

For each operation the corresponding SOAP action has to be defined. You must also specify how the input and output are encoded. In this case we use "literal".

XML SOAP

SOAP Simple Object Access Protocol is platform independent application communication protocol.

It is important for web applications to be able to communicate over the Internet.

The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers.

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

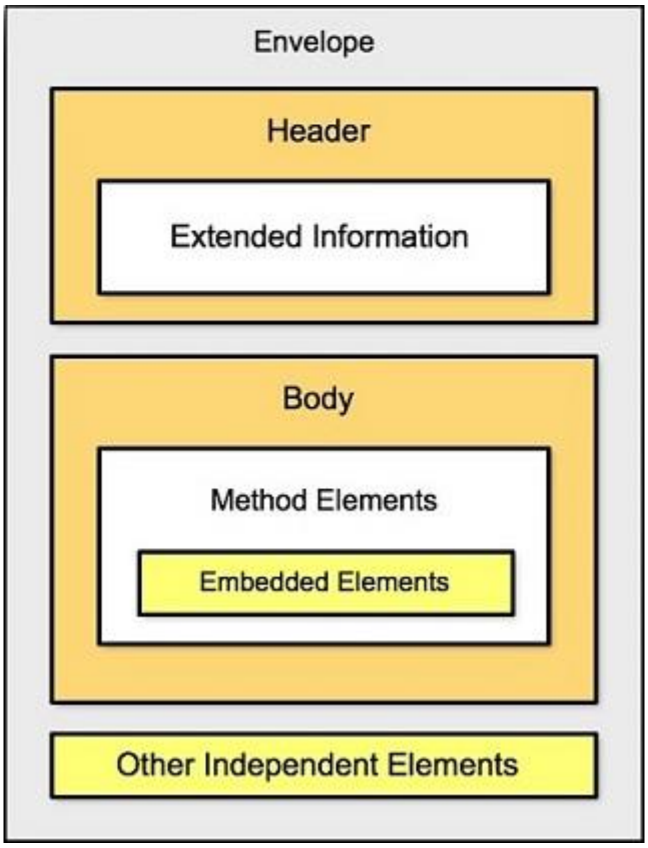
- An Envelope element that identifies the XML document as a SOAP message;
- A Header element that contains header information;
- A Body element that contains call and response information;
- A Fault element containing errors and status information.

All the elements above are declared in the default namespace for the SOAP envelope:

http://www.w3.org/2003/05/soap-envelope/

… and the default namespace for SOAP encoding and data types is:

http://www.w3.org/2003/05/soap-encoding



Syntax Rules

Here are some important syntax rules:

- A SOAP message MUST be encoded using XML;
- A SOAP message MUST use the SOAP Envelope namespace;

- A SOAP message must NOT contain a DTD reference;
- A SOAP message must NOT contain XML Processing Instructions.

Skeleton SOAP Message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

SOAP Binding

The SOAP specification defines the structure of the SOAP messages, not how they are exchanged. This gap is filled by what is called "SOAP Bindings". SOAP bindings are mechanisms which allow SOAP messages to be effectively exchanged using a transport protocol.

Most SOAP implementations provide bindings for common transport protocols, such as HTTP or SMTP:

- HTTP is synchronous and widely used. A SOAP HTTP request specifies at least two HTTP headers: Content-Type and Content-Length;
- SMTP is asynchronous and is used in last resort or particular cases.

Java implementations of SOAP usually provide a specific binding for the JMS protocol.

A SOAP Example

In the example below, a GetStockPrice request is sent to a server. The request has a StockName parameter, and a Price parameter that will be returned in the response. The namespace for the function is defined in "http://www.example.org/stock".

A SOAP request:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
```

```
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

The SOAP response:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

The SOAP Envelope Element

The required SOAP Envelope element is the root element of a SOAP message. This element defines the XML document as a SOAP message.

Example:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  ...
  Message information goes here
  ...
</soap:Envelope>
```

The xmlns:soap Namespace

Notice the xmlns:soap namespace in the example above. It should always have the value of: "http://www.w3.org/2003/05/soap-envelope/".

The namespace defines the Envelope as a SOAP Envelope.

If a different namespace is used, the application generates an error and discards the message.

The encodingStyle Attribute

The encodingStyle attribute is used to define the data types used in the document. This attribute may appear on any SOAP element, and applies to the element's contents and all child elements.

A SOAP message has no default encoding

Syntax:

```
soap:encodingStyle="URI"
```

The SOAP Header Element

The optional SOAP Header element contains application-specific information (like authentication, payment, etc) about the SOAP message.

If the Header element is present, it must be the first child element of the Envelope element.

Note: All immediate child elements of the Header element must be namespace-qualified.

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

  <soap:Header>
    <m:Trans xmlns:m="https://www.w3schools.com/transaction/"
    soap:mustUnderstand="1">234
    </m:Trans>
  </soap:Header>
  …
  …
</soap:Envelope>
```

The example above contains a header with

1. … a "Trans" element with a value of 234;
2. … a "mustUnderstand" attribute with a value of 1.

SOAP defines three attributes in the default namespace. These attributes are:

1. mustUnderstand;
2. actor;
3. encodingStyle.

The attributes defined in the SOAP Header defines how a recipient should process the SOAP message.

The mustUnderstand Attribute

The SOAP “mustUnderstand” attribute can be used to indicate whether a header entry is mandatory or optional for the recipient to process:

- If you add mustUnderstand="1" to a child element of the Header element it indicates that the receiver processing the Header must recognize the element;
- If the receiver does not recognize the element it will fail when processing the Header.

Syntax:

```
soap:mustUnderstand="0|1"
```

The actor Attribute

A SOAP message may travel from a sender to a receiver by passing different endpoints along the message path. However, not all parts of a SOAP message may be intended for the ultimate endpoint, instead, it may be intended for one or more of the endpoints on the message path.

The SOAP actor attribute is used to address the Header element to a specific endpoint.

Syntax:

```
soap:actor="URI"
```

Example:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Header>
  <m:Trans xmlns:m="https://www.w3schools.com/transaction/"
  soap:actor="https://www.w3schools.com/code/">234
  </m:Trans>
</soap:Header>
  …
  …
</soap:Envelope>
```

The encodingStyle Attribute

The encodingStyle attribute is used to define the data types used in the document. This attribute may appear on any SOAP element, and it will apply to that element's contents and all child elements.

A SOAP message has no default encoding.

Syntax:

```
soap:encodingStyle="URI"
```

The SOAP Body Element

The required SOAP Body element contains the actual SOAP message intended for the ultimate endpoint of the message.

Immediate child elements of the SOAP Body element may be namespace-qualified.

Example:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="https://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

The example above requests the price of apples. Note that the m:GetPrice and the Item elements above are application-specific elements. They are not a part of the SOAP namespace.

A SOAP response could look something like this:

```xml
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="https://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

The SOAP Fault Element

The optional SOAP Fault element is used to indicate error messages.

The SOAP Fault element holds errors and status information for a SOAP message.

If a Fault element is present, it must appear as a child element of the Body element. A Fault element can only appear once in a SOAP message.

The SOAP Fault element has the following sub elements:

Sub Element	Description
<faultcode>	A code for identifying the fault
<faultstring>	A human readable explanation of the fault
<faultactor>	Information about who caused the fault to happen
<detail>	Holds application specific error information related to the Body element

SOAP Fault Codes

The faultcode values defined below must be used in the faultcode element when describing faults:

Error	Description
VersionMismatch	Found an invalid namespace for the SOAP Envelope element
MustUnderstand	An immediate child element of the Header element, with the mustUnderstand attribute set to "1", was not understood
Client	The message was incorrectly formed or contained incorrect information
Server	There was a problem with the server so the message could not proceed