BAI3110 SOAP

from http://www.w3schools.com/soap/default.asp

What is SOAP?

- SOAP stands for Simple Object Access Protocol
- SOAP is a communication protocol
- SOAP is for communication **between applications**
- SOAP is a format for **sending messages**
- SOAP is designed to communicate **via Internet**
- SOAP is platform independent
- SOAP is language independent
- SOAP is based on XML
- SOAP is simple and extensible
- SOAP allows you to **get around firewalls**
- SOAP will be developed as a **W3C standard**

Why SOAP?

It is important for application development to allow Internet communication between programs.

Today's applications communicate using Remote Procedure Calls (RPC) between objects like DCOM and CORBA, but HTTP was not designed for this. RPC represents a compatibility and security problem; firewalls and proxy servers will normally block this kind of traffic.

A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

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

Microsoft and SOAP

SOAP is a key element of Microsoft's .NET architecture for future Internet application development.

SOAP 1.1 was Proposed to W3C

UserLand, Ariba, Commerce One, Compaq, Developmentor, HP, IBM, IONA, Lotus, Microsoft, and SAP proposed to W3C, in May 2000, the SOAP Internet protocol that they hope will revolutionize application development by connecting graphic user interface desktop applications to powerful Internet servers using the standards of the Internet; HTTP and XML.

W3C is Working on SOAP 1.2

The first public Working Draft on SOAP was published from W3C in December 2001. To read more about the SOAP activities at W3C please visit our W3C School.

SOAP Building Blocks

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

A required Envelope element that identifies the XML document as a SOAP message
An optional Header element that contains header information
A required Body element that contains call and response information
An optional Fault element that provides information about errors that occurred while processing the message

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

http://www.w3.org/2001/12/soap-envelope

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

http://www.w3.org/2001/12/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 use the SOAP Encoding 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/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
  ...
  ...
</soap:Header>
<soap:Body>
  ...
  ...
  <soap:Fault>
    ...
    ...
  </soap:Fault>
</soap:Body>
</soap:Envelope>
```

SOAP Envelope Element

The mandatory SOAP Envelope element is the root element of a SOAP message.

The SOAP Envelope Element

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

Note the use of the xmlns:soap namespace. It should always have the value of:

http://www.w3.org/2001/12/soap-envelope

and it defines the Envelope as a SOAP Envelope:

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

The xmlns:soap Namespace

A SOAP message must always have an Envelope element associated with the "http://www.w3.org/2001/12/soap-envelope" namespace.

If a different namespace is used, the application must generate an error and discard the message.

The encodingStyle Attribute

The SOAP 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"
```

Example

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

SOAP Header Element

The optional SOAP Header element contains header information.

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/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
<m:Trans
xmlns:m="http://www.w3schools.com/transaction/"
soap:mustUnderstand="1">234</m:Trans>
</soap:Header>
...
...
</soap:Envelope>
```

The example above contains a header with a "Trans" element, a "mustUnderstand" attribute value of "1", and a value of 234.

SOAP defines three attributes in the default namespace ("http://www.w3.org/2001/12/soap-envelope"). These attributes are: actor, mustUnderstand, and encodingStyle. The attributes defined in the SOAP Header defines how a recipient should process the SOAP message.

The actor Attribute

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

The SOAP actor attribute may be used to address the Header element to a particular endpoint.

Syntax

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

Example

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
<m:Trans
xmlns:m="http://www.w3schools.com/transaction/"
soap:actor="http://www.w3schools.com/appml/">
234
</m:Trans>
</soap:Header>
...
...
</soap:Envelope>
```

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 must fail when processing the Header.

Syntax

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

Example

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
<m:Trans
xmlns:m="http://www.w3schools.com/transaction/"
soap:mustUnderstand="1">
234
</m:Trans>
</soap:Header>
...
...
</soap:Envelope>
```

The encodingStyle Attribute

The SOAP encodingStyle attribute is explained in the previous chapter.

SOAP Body Element

The mandatory SOAP Body element contains the actual SOAP message.

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. SOAP defines one element inside the Body element in the default namespace ("http://www.w3.org/2001/12/soap-envelope"). This is the SOAP Fault element, which is used to indicate error messages.

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

A SOAP response could look something like this:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body>
   <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
      <m:Price>1.90</m:Price>
   </m:GetPriceResponse>
</soap:Body>
</soap:Envelope>
```

SOAP Fault Element

The optional SOAP Fault element is used to hold error and status information for a SOAP message.

The SOAP Fault Element

An error message from a SOAP message is carried inside a Fault element.

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

SOAP HTTP Binding

The HTTP Protocol

HTTP communicates over TCP/IP. An HTTP client connects to an HTTP server using TCP. After establishing a connection, the client can send an HTTP request message to the server:

```
POST /item HTTP/1.1
Host: 189.123.345.239
Content-Type: text/plain
Content-Length: 200
```

The server then processes the request and sends an HTTP response back to the client. The response contains a status code that indicates the status of the request:

```
200 OK
Content-Type: text/plain
Content-Length: 200
```

In the example above, the server returned a status code of 200. This is the standard success code for HTTP.

If the server could not decode the request, it could have returned something like this:

```
400 Bad Request
Content-Length: 0
```

SOAP HTTP Binding

A SOAP method is an HTTP request/response that complies with the SOAP encoding rules.

HTTP + XML = SOAP

A SOAP request could be an HTTP POST or an HTTP GET request.

The HTTP POST request specifies at least two HTTP headers: Content-Type and Content-Length.

Content-Type

The Content-Type header for a SOAP request and response defines the MIME type for the message and the character encoding (optional) used for the XML body of the request or response.

Syntax

```
Content-Type: MIMEType; charset=character-encoding
```

Example

```
POST /item HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
```

Content-Length

The Content-Length header for a SOAP request and response specifies the number of bytes in the body of the request or response.

Syntax

```
Content-Length: bytes
```

Example

```
POST /item HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 250
```

SOAP Example

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 will be returned in the response. The namespace for the function is defined in "http://www.stock.org/stock" address.

The SOAP request:

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

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.stock.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

A SOAP response:

```
HTTP/1.1 200 OK
Content-Type: application/soap; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.stock.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```

Introduction to WSDL

WSDL is an XML-based language for describing Web services and how to access them.

What You Should Already Know

Before you study the WSDL tutorial, you should have a basic understanding of XML Namespaces and XML Schema.

If you want to study these subjects first, please visit our XML tutorial, and our Schema tutorial.

What is WSDL?

WSDL stands for Web Services Description Language
WSDL is written in XML
WSDL is an XML document
WSDL is used to describe Web services
WSDL is also used to locate Web services
WSDL is not yet a W3C standard

WSDL Describes Web Services

WSDL stands for Web Services Description Language.

WSDL is a document written in XML. The document describes a Web service. It specifies the location of the service and the operations (or methods) the service exposes.

WSDL Development History at W3C

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.

(a W3C Note is made available by the W3C for discussion only. Publication of a Note by W3C indicates no endorsement by W3C or the W3C Team, or any W3C Members)

The first Working Draft of WSDL 1.2 was released by W3C in July 2002.

WSDL Documents

A WSDL document is just a simple XML document.

It contains set of definitions to define a web service.

The WSDL Document Structure

A WSDL document defines a web service using these major elements:

Element	Defines
<portType>	The operations performed by the web service
<message>	The messages used by the web service
<types>	The data types used by the web service
<binding>	The communication protocols used by the web service

The main structure of a WSDL document looks like this:

```
<definitions>
   <types>
      definition of types........
   </types>
   <message>
      definition of a message....
   </message>
   <portType>
      definition of a port.......
   </portType>
   <binding>
      definition of a binding....
   </binding>
</definitions>
```

A WSDL document can also contain other elements, like extension elements and a service element that makes it possible to group together the definitions of several web services in one single WSDL document.

For a complete syntax overview go to the chapter WSDL Syntax.

WSDL Ports

The **<portType>** element is the most important WSDL element.

It defines a web service, the operations that can be performed, and the messages that are involved.

The <portType> element can be compared to a function library (or a module, or a class) in a traditional programming language.

WSDL Messages

The **<message>** element defines the data elements of an operation.

Each messages can consist of one or more parts. The parts can be compared to the parameters of a function call in a traditional programming language.

WSDL Types

The **<types>** element defines the data type that are used by the web service.

For maximum platform neutrality, WSDL uses XML Schema syntax to define data types.

WSDL Bindings

The **<binding>** element defines the message format and protocol details for each port.

WSDL Example

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" and an **output message** called "getTermResponse".

The **message** elements defines the **parts** of each message and the associated data types.

Compared to traditional programming, glossaryTerms is a function library, "getTerm" is a function with "getTermRequest" as the input parameter and getTermResponse as the return parameter.

WSDL Ports

A WSDL port describes the interfaces (legal operations) exposed by a web service.

WSDL Ports

The **<portType>** element is the most important WSDL element.

It defines **a web service**, the **operations** that can be performed, and the **messages** that are involved.

The port defines the connection point to a web service. It can be compared to a function library (or a module, or a class) in a traditional programming language. Each operation can be compared to a function in a traditional programming language.

Operation Types

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

One-Way Operation

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 this example the port "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.

Request-Response Operation

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>
```

In this example the port "glossaryTerms" defines a request-response operation called "getTerm".

The "getTerm" operation requires an input message called "getTermRequest" with a parameter called "term", and will return an output message called "getTermResponse" with a parameter called "value".

WSDL Bindings

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

Binding to SOAP

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 - the name attribute and the type attribute.

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 - the style attribute and the transport attribute.

The style attribute can be "rpc" or "document". In this case we use document. The transport attribute defines the SOAP protocol to use. In this case we use HTTP.

The **operation** element defines each operation that the port 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".

WSDL and UDDI

Universal Description, Discovery and Integration (UDDI) is a directory service where businesses can register and search for Web services.

What is UDDI

UDDI is a platform-independent framework for describing services, discovering businesses, and integrating business services by using the Internet.

UDDI stands for Universal Description, Discovery and Integration
UDDI is a directory for storing information about web services
UDDI is a directory of web service interfaces described by WSDL
UDDI communicates via SOAP
UDDI is built into the Microsoft .NET platform

What is UDDI Based On?

UDDI uses World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) Internet standards such as XML, HTTP, and DNS protocols.

UDDI uses WSDL to describe interfaces to web services

Additionally, cross platform programming features are addressed by adopting SOAP, known as XML Protocol messaging specifications found at the W3C Web site.

UDDI Benefits

Any industry or businesses of all sizes can benefit from UDDI

Before UDDI, there was no Internet standard for businesses to reach their customers and partners with information about their products and services. Nor was there a method of how to integrate into each other's systems and processes.

Problems the UDDI specification can help to solve:

Making it possible to discover the right business from the millions currently online
Defining how to enable commerce once the preferred business is discovered
Reaching new customers and increasing access to current customers
Expanding offerings and extending market reach
Solving customer-driven need to remove barriers to allow for rapid participation in the global Internet economy
Describing services and business processes programmatically in a single, open, and secure environment

How can UDDI be Used

If the industry published an UDDI standard for flight rate checking and reservation, airlines could register their services into an UDDI directory. Travel agencies could then search the UDDI directory to find the airline's reservation interface. When the interface is found, the travel agency can communicate with the service immediately because it uses a well-defined reservation interface.

Who is Supporting UDDI?

UDDI is a cross-industry effort driven by all major platform and software providers like Dell, Fujitsu, HP, Hitachi, IBM, Intel, Microsoft, Oracle, SAP, and Sun, as well as a large community of marketplace operators, and e-business leaders.

Over 220 companies are members of the UDDI community.

The Full WSDL Syntax

The full WSDL 1.2 syntax as described in the W3C Working Draft is listed below.

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri">
    <import namespace="uri" location="uri"/> *
    <wsdl:documentation .... /> ?
    <wsdl:types> ?
```

```
        <wsdl:documentation .... /> ?
        <xsd:schema .... /> *
    </wsdl:types>
    <wsdl:message name="ncname"> *
        <wsdl:documentation .... /> ?
        <part name="ncname" element="qname"? type="qname"?/> *
    </wsdl:message>
    <wsdl:portType name="ncname"> *
        <wsdl:documentation .... /> ?
        <wsdl:operation name="ncname"> *
            <wsdl:documentation .... /> ?
            <wsdl:input message="qname"> ?
                <wsdl:documentation .... /> ?
            </wsdl:input>
            <wsdl:output message="qname"> ?
                <wsdl:documentation .... /> ?
            </wsdl:output>
            <wsdl:fault name="ncname" message="qname"> *
                <wsdl:documentation .... /> ?
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:serviceType name="ncname"> *
        <wsdl:portType name="qname"/> +
    </wsdl:serviceType>
    <wsdl:binding name="ncname" type="qname"> *
        <wsdl:documentation .... /> ?
        <-- binding details --> *
        <wsdl:operation name="ncname"> *
            <wsdl:documentation .... /> ?
            <-- binding details --> *
            <wsdl:input> ?
                <wsdl:documentation .... /> ?
                <-- binding details -->
            </wsdl:input>
            <wsdl:output> ?
                <wsdl:documentation .... /> ?
                <-- binding details --> *
            </wsdl:output>
            <wsdl:fault name="ncname"> *
                <wsdl:documentation .... /> ?
                <-- binding details --> *
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="ncname" serviceType="qname"> *
        <wsdl:documentation .... /> ?
        <wsdl:port name="ncname" binding="qname"> *
            <wsdl:documentation .... /> ?
            <-- address details -->
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```