**SOAP**

|  |  |  |
| --- | --- | --- |
| SOAP |  | **SOAP is a simple and open standard XML-based protocol for exchanging information between computers.**  **In this tutorial you will learn what is SOAP and Why and How to use it.**  **SOAP is very easy to learn and to use and in demand too.** |

**SOAP is an XML-based protocol for exchanging information between computers.**

**SOAP is XML. That is, SOAP is an application of the XML specification.**

**All statements are TRUE for SOAP**

* SOAP is acronym for Simple Object Access Protocol
* SOAP is a communication protocol
* SOAP is 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 gets sent and how

Although SOAP can be used in a variety of messaging systems and can be delivered via a variety of transport protocols, the initial focus of SOAP is remote procedure calls transported via HTTP.

SOAP enables client applications to easily connect to remote services and invoke remote methods.

Other frameworks, including CORBA, DCOM, and Java RMI, provide similar functionality to SOAP, but SOAP messages are written entirely in XML and are therefore uniquely platform- and language-independent.

## SOAP - Recommended Knowledge

It is recommended that before you proceed further you should be familiar with XML and XML namespace.

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

* **Envelope:**(Mandatory)  
  Defines the start and the end of the message.
* **Header:**(Optional)  
  Contains any optional attributes of the message used in processing the message, either at an intermediary point or at the ultimate end point.
* **Body:**(Mandatory)  
  Contains the XML data comprising the message being sent.
* **Fault:**(Optional)  
  An optional Fault element that provides information about errors that occurred while processing the message

## A SOAP Message Structure

|  |
| --- |
| <?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>  ...  ...  </SOAP-ENV:Header>  <SOAP-ENV:Body>  ...  ...  <SOAP-ENV:Fault>  ...  ...  </SOAP-ENV:Fault>  </SOAP-ENV:Body>  </SOAP\_ENV:Envelope> |

|  |  |  |
| --- | --- | --- |
| The SOAP envelope indicates the start and the end of the message so that the receiver knows when an entire message has been received. The SOAP envelope solves the problem of knowing when you're done receiving a message and are ready to process it. The SOAP envelope is therefore basic ally a packaging mechanism  SOAP Envelope element can be explained as:   * Every SOAP message has a root Envelope element. * Envelope element is 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, beforethe Body. * The envelope changes when SOAP versions change. * The SOAP envelope is specified using the *ENV* namespace prefix and the *Envelope* element. * The optional SOAP encoding is also specified using a namespace name and the optional *encodingStyle* element, which could also point to an encoding style other than the SOAP one. * A v1.1-compliant SOAP processor will generate a fault when receiving a message containing the v1.2 envelope namespace. * A v1.2- compliant SOAP processor generates a *VersionMismatch* fault if it receives a message that does not include the v1.2 envelope namespace.   Example for v1.2 is given below   |  | | --- | | <?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> |   Following example illustrates the use of a SOAP message within an HTTP POST operation, which sends the message to the server. It shows the namespaces for the envelope schema definition and for the schema definition of the encoding rules. The *OrderEntry* reference in the HTTP header is the name of the program to be invoked at the tutorialspoint.com Web site.   |  | | --- | | POST /OrderEntry HTTP/1.1  Host: www.tutorialspoint.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> |   **NOTE:** The HTTP binding specifies the location of the service. |

SOAP HEADER

The optional Header element offers a flexible framework for specifying additional application-level requirements. For example, the Header element can be used to specify a digital signature for password-protected services; likewise, it can be used to specify an account number for pay-per-use SOAP services.

SOAP Header element can be explained as:

* Header elements are optional part of SOAP messages.
* 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 more than one header is defined, all immediate child elements of the SOAP header are interpreted as SOAP header blocks.

SOAP Header element can have following two attributes

* **Actorattribute:**  
  The SOAP protocol defines a message path as a list of SOAP service nodes. Each of these intermediate nodes can perform some processing and then forward the message to the next node in the chain. By setting the Actor attribute, the client can specify the recipient of the SOAP header.
* **MustUnderstandattribute**  
  Indicates whether a Header element is optional or mandatory. If set to true ie. 1 the recipient must understand and process the Header attribute according to its defined semantics, or return a fault.

Following example shows how to use a Header in the 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">  <SOAP-ENV:Header>  <t:Transaction  xmlns:t="http://www.tutorialspoint.com/transaction/"  SOAP-ENV:mustUnderstand="true">5</t:Transaction>  </SOAP-ENV:Header>  ...  ...  </SOAP-ENV:Envelope> |

SOAP BODY

|  |  |  |
| --- | --- | --- |
| The SOAP body is a mandatory element which contains the application-defined XML data being exchanged in the SOAP message. The body must be contained within the envelope and must follow any headers that might be defined for the message. The body is defined as a child element of the envelope, and the semantics for the body are defined in the associated SOAP schema.  The body contains mandatory information intended for the ultimate receiver of the message. For example:   |  | | --- | | <?xml version="1.0"?>  <SOAP-ENV:Envelope  ........  <SOAP-ENV:Body>  <m:GetQuotation xmlns:m="http://www.tp.com/Quotation">  <m:Item>Computers</m:Item>  </m:GetQuotation>  </SOAP-ENV:Body>  </SOAP-ENV:Envelope> |   The example above requests the quotation of computer sets. Note that the m:GetQuotation and the Item elements above are application-specific elements. They are not a part of the SOAP standard.  Here is the response of above query:   |  | | --- | | <?xml version="1.0"?>  <SOAP-ENV:Envelope  ........  <SOAP-ENV:Body>  <m:GetQuotationResponse xmlns:m="http://www.tp.com/Quotation">  <m:Quotation>This is Qutation</m:Quotation>  </m:GetQuotationResponse>  </SOAP-ENV:Body>  </SOAP-ENV:Envelope> |   Normally, the application also defines a schema to contain semantics associated with the request and response elements.  The *Quotation* service might be implemented using an EJB running in an application server; if so, the SOAP processor would be responsible for mapping the body information as parameters into and out of the EJB implementation of the *GetQuotationResponse* service. The SOAP processor could also be mapping the body information to a .NET object, a CORBA object, a COBOL program, and so on. |

SOAP FAULT

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| When 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, the address of the SOAP processor that generated   * A SOAP Message can carry only one fault block * Fault element is an optional part of SOAP Message * For the 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.   The SOAP Fault element has the following sub elements:   |  |  | | --- | --- | | **Sub Element** | **Description** | | <faultCode> | A text code used to indicate a class of errors. See the next Table for a listing of predefined fault codes. | | <faultString> | A text message explaning the error | | <faultActor> | A text string indicating who caused the fault. This 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. | | <detail> | An element used to carry application-specific error messages. The detail element can contain child elements, called detail entries. |  SOAP Fault Codes The faultCode values defined below must be used in the faultcode element when describing faults   |  |  | | --- | --- | | **Error** | **Description** | | SOAP-ENV:VersionMismatch | Found an invalid namespace for the SOAP Envelope element | | SOAP-ENV:MustUnderstand | An immediate child element of the Header element, with the mustUnderstand attribute set to "1", was not understood | | SOAP-ENV:Client | The message was incorrectly formed or contained incorrect information | | SOAP-ENV:Server | There was a problem with the server so the message could not proceed |  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:   |  | | --- | | <?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.This enables the SOAP message to indicate specific data types, such as integers, floats, doubles, or arrays.

* 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.
* 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>
* 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.

|  |  |  |
| --- | --- | --- |
| **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 |  |

For example, here is a SOAP response with a double data type:

|  |
| --- |
| <?xml version='1.0' encoding='UTF-8'?\*gt;  <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.

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]" |

Here is a sample SOAP response with an array of double values:

|  |
| --- |
| <?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: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> |

**NOTE:** Please you take care of proper indentation while you write your SOAP code.

Each element in a struct is specified with a unique accessor name. For example, the message above includes four accessor elements: name , price , description , and SKU. Each element can have its own data type; for example, name is specified as a string , whereas price is specified as a double.

SOAP Transport

|  |
| --- |
| * SOAP is not tied to any one 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.  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.  SOAP responses delivered via HTTP are required to follow the same HTTP status codes. For example, a status code of 200 OK indicates a successful response. A status code of 500 Internal Server Error indicates that there is a server error and that the SOAP response includes a Fault element. |

Example:

|  |  |  |
| --- | --- | --- |
| In the example below, 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.  Here is the SOAP 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>MiscroSoft</m:QuotationsName>      </m:GetQuotation>  </SOAP-ENV:Body>  </SOAP-ENV:Envelope> |   A corresponding SOAP response will look like :   |  | | --- | | 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> | |