

An Introduction to....

# Web Services

Module-2  
Prof. Nirav Suthar  
FCAIT-UG

## Module-2

- Background
- What is SOAP
- Building Blocks
- Message Structure
  - The Envelope element
  - The header element and header fault
  - The body element
  - The Fault element
- SOAP Encoding
- SOAP Binding
- Demonstration on elements of SOAP

# BACKGROUND

Basic XML provided a firm foundation for the data interchange and interoperability. The XML specification takes care of only the data representation part. However, it does not make any assumptions or rely on any protocol for its transport.

In fact, the transport part has consciously been excluded in the design of XML. However, with the Internet revolution, HTTP has become critical to the success of XML. The idea of transporting XML data as a payload of HTTP has triggered the emergence of web services.

**Dave Winer's** work has resulted in a simple and "portable" way of malting RPC (Remote procedure call) calls over HTTP. These RPC calls use text-based as against the conventional RPC calls that were used in other distributed environments. This resulted in the opening of a new perspective in the history of middleware technologies, laying the foundation for a new protocol called Simple Object Access Protocol (SOAP).

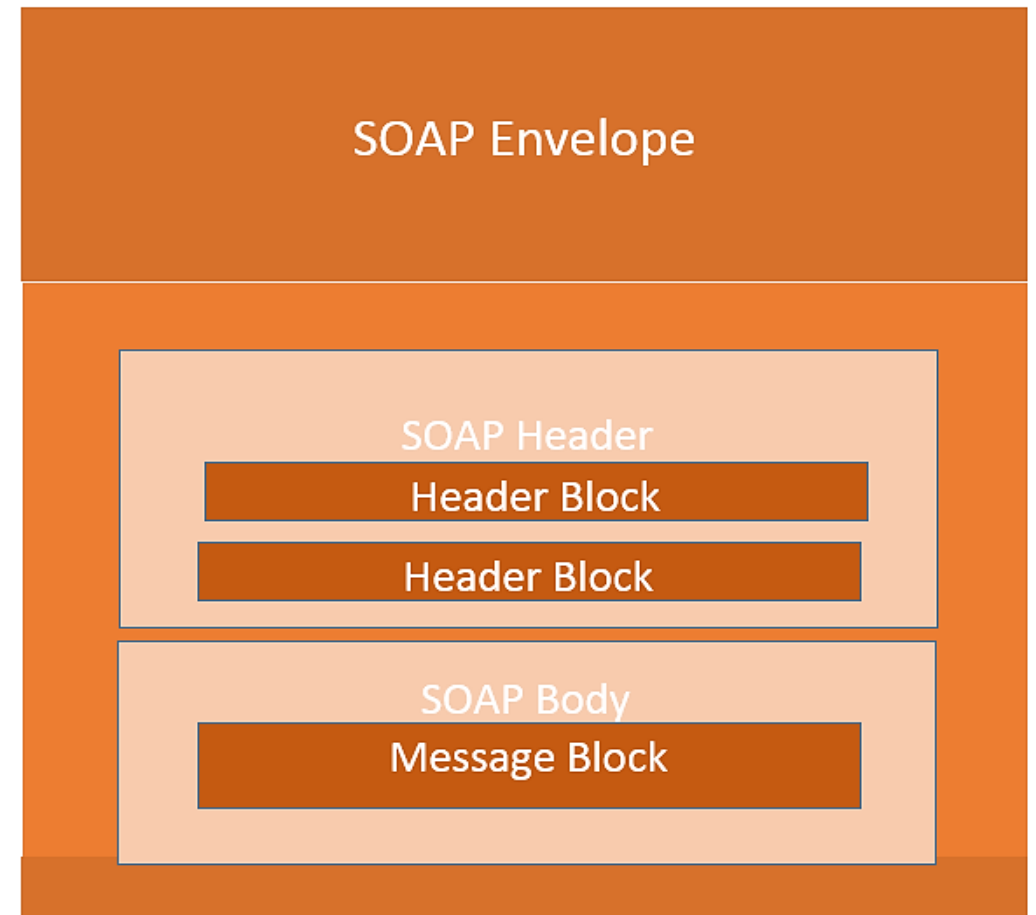
# SOAP

- SOAP is an acronym for **Simple Object Access Protocol(SOAP)** is a **network protocol for exchanging structured data between nodes**.
- SOAP is a **XML-based protocol** for accessing web services.
- SOAP is a W3C (governing body for all web standards) recommendation for communication between applications.
- SOAP is XML based, so it is platform independent and language independent. In other words, it can be used with Java, .Net or PHP language on any platform.

- SOAP was developed as an intermediate language so that applications built on various programming languages could talk easily to each other and avoid the extreme development effort.
- Every programming language can understand the **XML( Extensible Markup Language)**. Hence, XML was used as the underlying medium for data exchange.
- But there are no standard specifications on use of XML across all programming languages for data exchange. That is where SOAP comes in.
- SOAP was designed to work with XML over HTTP and have some sort of specification which could be used across all applications.

# SOAP Building blocks

The SOAP specification defines something known as a "**SOAP message**" which is sent to the web service and the client application.



# SOAP Message Structure

- The Envelope element
- The header element and
- The body element
- The Fault element (Optional)

SOAP message transmits some basic information as given below

- Information about message structure and instructions on processing it.
- Encoding instructions for application-defined data types.
- Information about Remote Procedure Calls and their responses.

**Note:** That whenever a client application calls a method in the web service, the web service will automatically generate a SOAP message which will have the necessary details of the data which will be sent from the web service to the client application.

## An Envelope element that identifies the XML document as a SOAP message

This is the containing part of the SOAP message and is used to encapsulate all the details in the SOAP message. This is the root element in the SOAP message.

The SOAP envelope element is used to indicate the beginning and end of a SOAP message. This enables the client application which calls the web service to know when the SOAP message ends.

### The following points can be noted on the SOAP envelope element.

- Every SOAP message needs to have a root Envelope element.
- Every Envelope element needs to have **at least one soap body** element.
- If an Envelope element **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 element.
- **The envelope changes when SOAP versions change.**



SOAP  
Envelope  
Element

<?xml version="1.0" encoding="utf-8"?>  
<soap:Envelope xmlns:xsi=  
<http://www.w3.org/2001/XMLSchema-instance>>

SOAP Body  
Element

<soap:Body>  
    <Unit2Webservice xmlns=<http://tempura.org/>>  
        <StudentID> int </StudentID>  
    </Unit2Webservice>  
</soap:body>  
</soap:Envelope>

Parameters  
required by web  
service

Name of the  
Web Service

1. As seen from the above SOAP message, the first part of the SOAP message is the envelope element which is used to encapsulate the entire SOAP message.
2. The next element is the SOAP body which contains the details of the actual message.
3. Our message contains a web service which has the name of "Unit2Webservice".
4. The "Unit2Webservice" accepts a parameter of the type 'int' and has the name of "StudentID".

## **A Header element that contains header information**

- The header element can contain information such as authentication credentials which can be used by the calling application.
- It can also contain the definition of complex types which could be used in the SOAP message.
- By default, the SOAP message can contain parameters which could be of simple types such as strings and numbers, but can also be a complex object type.

- Suppose we wanted to send a structured data type which had a combination of a **"TopicName"** and a **"TopicDescription,"**.
- The complex type is defined by the element tag **<xsd:complexType>**.
- All of the required elements of the structure along with their respective data types are then defined in the complex type collection.

```
<xsd:complexType>  
  <xsd:sequence>  
    <xsd:element name="TopicName"  
      type="string"/>  
    <xsd:element  
      name="TopicDescription"  
      type="string"/>  
  </xsd:sequence>  
</xsd:complexType>
```

# A Body element that contains call and response information

- This element is what contains the actual data which needs to be sent between the web service and the calling application.
- Below is an example of the SOAP body which actually works on the complex type defined in the header section.
- Here is the response of the **TopicName** and **TopicDescription** that is sent to the calling application which calls this web service.

```
<soap:Body>  
  <Unit2Webservice>  
    <TopicName>Web Services</TopicName>  
    <TopicDescription>All abt web Services</TopicDescription>  
  </Unit2Webservice>  
</soap:Body>
```

# The Fault message

When a request is made to a SOAP web service, the response returned can be of either 2 forms which are a **successful response or an error response**. When a success is generated, the response from the server will always be a SOAP message. But if SOAP faults are generated, they are returned as "**HTTP 500**" errors. The SOAP Fault message consists of the following elements.

1. **<faultCode>**- This is the code that designates the code of the error. The fault code can be either of any below values
  1. SOAP-ENV:VersionMismatch – This is when an invalid namespace for the SOAP Envelope element is encountered.
  2. SOAP-ENV:MustUnderstand - An immediate child element of the Header element, with the mustUnderstand attribute set to "1", was not understood.
  3. SOAP-ENV:Client - The message was incorrectly formed or contained incorrect information.
  4. SOAP-ENV:Server - There was a problem with the server, so the message could not proceed.
2. **<faultString>** - This is the text message which gives a detailed description of the error.
3. **<faultActor> (Optional)**- This is a text string which indicates who caused the fault.
4. **<detail>(Optional)** - This is the element for application-specific error messages. So the application could have a specific error message for different business logic scenarios.

# Example for Fault Message

An example of a fault message is given below. The error is generated if the scenario wherein the client tries to use a method called GetStudentID in the class GetTopic. The below fault message gets generated in the event that the method does not exist in the defined class.

```
<?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 (GetStudentID) in class (GetTopic)
      </faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## Output:

When you execute the above code, it will show the error like "Failed to locate method (GetStudentID) in class (GetTopic)"

# Advantages of SOAP Web Services

**WS Security:** SOAP defines its own security known as WS Security.

**Language and Platform independent:** SOAP web services can be written in any programming language and executed in any platform.

# Disadvantages of SOAP Web Services

**Slow:** SOAP uses XML format that must be parsed to be read. It defines many standards that must be followed while developing the SOAP applications. So it is slow and consumes more bandwidth and resource.

**WSDL dependent:** SOAP uses WSDL and doesn't have any other mechanism to discover the service.



# 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 data types are divided into two broad categories – scalar types/simple type and compound types.
- Scalar types/simple type 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.

# SOAP Encoding

A **Simple Type or Scalar Type** can either be a **built-in** or a **derived datatype**.

A **built- in data type** is a data type that is defined in the XML Schema. (*Example1*)

**Example2** provides example of **derived data types** : Where the top box provides the **definition of three data types**, namely **<deposit>**,**<withdraw>**, and **<accountType>** and their respective datatypes are **<float>** , **<float>** and **<string>**

## Example1:

```
.....  
.....  
<int>123543</int>  
<float>3.14</float>  
<string>Server Down</string>  
.....  
.....
```

# SOAP Encoding

## Example2:

.....

<xs: element name : “deposit” type: “xs: float”>

<xs: element name : “withdraw” type: “xs: float”>


<xs: element name : “accountType” type: “xs: string”>

.....



<accountType> Savings</account-type>


<withdraw> 5500.00</withdraw>



Ex-1

<accountType> Savings</account-type>

<deposit> 1500.00</withdraw>

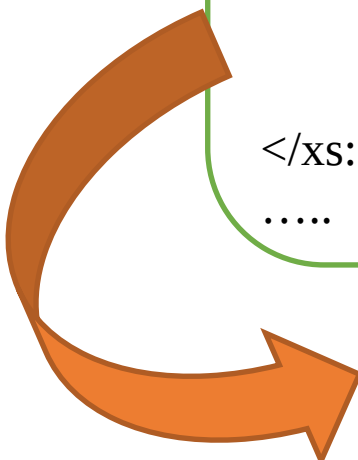


Ex-2

# SOAP Encoding

Compound Type can be of Struct or Array data types. The struct or “structure” data type is explained below with example:

```
.....  
<xs: element name="accountInfo" xmlns: xs ="http://www.w3.org/2001/XMLSchema">  
<xs: complexType>  
  <xs: sequence>  
    <xs: element name = "holderName" type="xs: string" />  
    <xs: element name = "accountId" type="xs: int" />  
    <xs: element name = "holderAddress" type="xs: string" />  
  </xs: sequence>  
</xs: complexType>  
</xs:element>  
.....
```



```
...  
<e: accountInfo  
  xmlns : e= http://www.Infybank.com/data/accounts>  
  <holderName>ABCD</holderName>  
  <accountId>123456</accountId>  
  <holderAddress><1,apq ,Ahmedabad </holderAddress>  
  </e:accountInfo>  
.....
```

# SOAP Encoding

...

<e: accountInfo

xmlns : e= http://www.Infybank.com/data/accounts>

<holderName>LMNO</holderName>

<accountId>898445588</accountId>

<holderAddress>5,h1 ,Ahmedabad </holderAddress>

</e:accountInfo>

....

# SOAP Binding

- The design of SOAP ensures that there is no explicit dependency on any transport protocol. However in enterprise scenarios, HTTP is used as the primary transport protocol and is a natural choice for SOAP transport for enterprises connected on the Internet.
- However, SOAP can be bound to other transport protocols such as FTP It is also important to note here that any other
- transport protocols can as easily carry SOAP.
- HTTP is the choice of transport protocol in the B2B scenarios as well. Thanks to the Internet revolution, most of the enterprises are already using web as the medium.
- One other reason that HTTP has become the protocol of choice for sending SOAP is that HTTP is a request-response oriented protocol.
- Service invocation using request-response is one of the widely accepted choices in the enterprise B2B scenario.

# SOAP Binding

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>
```

# SOAP Binding

A corresponding SOAP response looks 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>
```