**1. what is web service?**

Ans: Web services (can be hosted in the internet) are can be used instead of web applications to unite the multiple platforms by using standard communication language called XML which is independent language.

Integration of heterogeneous applications can be possible by using web services.

SOAP and REST is a protocol used to make a communication between web services and XML.

Example: soap sends request with add operation to calculate request and it understands and revert back with soap response to the client side.

SOAP start and end with envelope tag.

And body is mandatory for soap.

**2. what is XML and what will be format?**

XML is used to encode all communications to a web service.

* XML stands for extensible Markup Language
* XML is a markup language much like HTML
* XML was designed to store and transport data
* XML was designed to be self-descriptive
* XML is a W3C Recommendation.

XML is a software- and hardware-independent tool for storing and transporting data.

XML is a file extension for an [Extensible Markup Language](http://searchsoa.techtarget.com/definition/XML) (XML) file format used to create common [information](http://searchsqlserver.techtarget.com/definition/information) formats and share both the format and the [data](http://searchdatamanagement.techtarget.com/definition/data) on the World Wide Web, intranets, and elsewhere using standard ASCII text.

**3. steps to test soap and rest web services using soapUI pro ( www.soapui.org )**

**Rest web services testing using SoapUI:**

In SoapUI

1. Go to File > New Soap Project
2. Enter the project Name and the WSDL URI location
3. Click OK
4. Expand the first request and double click on the 'Request1'. It will display the SOAP request in the XML format.
5. Enter the From Currency and To Currency
6. Click on the submit button
7. Response XML will be displayed right side pane.

**4. What is requestxml and responseXML**

You can construct a SOAP/XML (Simple Object Access Protocol/Extensible Markup Language) request to create, update, or get component interface rows. The request and response contain component interface data in a SOAP/XML format.

**5. Different http methods?**

|  |  |
| --- | --- |
| 1 | **GET:** The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data. |
| 2 | **HEAD:** Same as GET, but transfers the status line and header section only. |
| 3 | **POST:** A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms. |
| 4 | **PUT:** Replaces all current representations of the target resource with the uploaded content. |
| 5 | **DELETE:** Removes all current representations of the target resource given by a URI. |
| 6 | **CONNECT:** Establishes a tunnel to the server identified by a given URI. |
| 7 | **OPTIONS:** Describes the communication options for the target resource. |
| 8 | **TRACE:** Performs a message loop-back test along the path to the target resource. |

**6. What is json and how is the format look like?**

JSON (JavaScript Object Notation) is a lightweight data-interchange format.

JSON, or JavaScript Object Notation, is a minimal, readable format for structuring data. It is used primarily to transmit data between a server and web application, as an alternative to XML. Square space uses JSON to store and organize site content created with the CMS.

1. **Difference between soap web service and rest web service**

REST is almost always going to be faster. The main advantage of SOAP is that it provides a mechanism for services to describe themselves to clients, and to advertise their existence.

REST is much more lightweight and can be implemented using almost any tool, leading to lower bandwidth and shorter learning curve. However, the clients have to know what to send and what to expect.

In general, When you're publishing an API to the outside world that is either complex or likely to change, SOAP will be more useful. Other than that, REST is usually the better option.

Soap Vs Rest:

SOAP is surely a heavyweight choice for Web service access. It provides the following advantages when compared to REST:

* Language, platform, and transport independent (REST requires use of HTTP)
* Works well in distributed enterprise environments (REST assumes direct point-to-point communication)
* Standardized
* Provides significant pre-build extensibility in the form of the WS\* standards
* Built-in error handling
* Automation when used with certain language products

REST is easier to use for the most part and is more flexible. It has the following advantages when compared to SOAP:

* No expensive tools require to interact with the Web service
* Smaller learning curve
* Efficient (SOAP uses XML for all messages, REST can use smaller message formats)
* Fast (no extensive processing required)
* Closer to other Web technologies in design philosophy

1. **assertions in soapUI?**

Assertion means act of affirming or stating something. It can also be interpreted as check point or a validation point.

* Once a request is sent to a web server a response is received. We need to validate if the response contains the data that we expect. In order to validate the response, we need to use assertions.

**12) xpath from w3school**

XPath is a major element in the XSLT standard. XPath can be used to navigate through elements and attributes in an XML document.

* XPath is a syntax for defining parts of an XML document
* XPath uses path expressions to navigate in XML documents
* XPath contains a library of standard functions
* XPath is a major element in XSLT and in XQuery
* XPath is a W3C recommendation

XPath uses path expressions to select nodes or node-sets in an XML document. These path expressions look very much like the expressions you see when you work with a traditional computer file system.

XPath expressions can be used in JavaScript, Java, XML Schema, PHP, Python, C and C++, and lots of other languages.

XPath is a major element in the XSLT standard.

With XPath knowledge you will be able to take great advantage of XSL.

Xpath example:

<?xml version="1.0" encoding="UTF-8"?>  
  
<bookstore>  
  
<book category="cooking">  
  <title lang="en">Everyday Italian</title>  
  <author>Giada De Laurentiis</author>  
  <year>2005</year>  
  <price>30.00</price>  
</book>  
  
<book category="children">  
  <title lang="en">Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>  
  
<book category="web">  
  <title lang="en">XQuery Kick Start</title>  
  <author>James McGovern</author>  
  <author>Per Bothner</author>  
  <author>Kurt Cagle</author>  
  <author>James Linn</author>  
  <author>Vaidyanathan Nagarajan</author>  
  <year>2003</year>  
  <price>49.99</price>  
</book>  
  
<book category="web">  
  <title lang="en">Learning XML</title>  
  <author>Erik T. Ray</author>  
  <year>2003</year>  
  <price>39.95</price>  
</book>  
  
</bookstore>

**13) what is / and // in xpath**

/ Search directives :

1. It's starts search selection from root element in document.
2. XPath expressions is like absolute path from the root element.
3. /empinfo : When /empinfo is a absolute path from the root element. pretend as a root node.

You can use XPath search directives in middle of an XPath address to represent ancestor descendant relationships.

XPath addresses /empinfo/employee identifies employee elements. employee that is a next child (descendant) of empinfo element.

/ Search directives :

1. It's starts search selection anywhere in XML document.
2. XPath expressions is like relative path from the context node.
3. //empinfo : When //empinfo define node path in anywhere in XML document. "//" find empinfo element locate any depth of XML document.

XPath addresses /empinfo//designation identifies any descendant designation element of element empinfo.

14) how to read attributes in xpath

<?xml version="1.0" encoding="ISO-8859-1"?>

<bookstore>

<book>

<title lang="eng">Harry Potter</title>

<price>29.99</price>

</book>

<book>

<title lang="eng">Learning XML</title>

<price>39.95</price>

</book>

</bookstore>

15) different functions in xpath (contains(), text() etc)

The contains function determines whether the first argument string contains the second argument string and returns boolean true or false.

contains( haystack , needle )

The count function counts the number of nodes in a node-set and returns an integer

count( node-set )

**16) how to read child nodes/parent nodes in xpath**

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output omit-xml-declaration="yes"/>

<xsl:template match="/">

<xsl:copy-of select="/\*/\*/X/node()"/>

</xsl:template>

</xsl:stylesheet>

First Text Node #1

<y> Y can Have Child Nodes #

<child> deep to it </child>

</y> Second Text Node #2

<z />

WADL is a XML description of a deployed RESTful web application. It contains model of the deployed resources, their structure, supported media types, HTTP methods and so on. In a sense, WADL is a similar to the WSDL (Web Service Description Language) which describes SOAP web services. WADL is however specifically designed to describe RESTful Web resources.

XSD can also be used for generating XML documents that can be treated as programming objects. In addition, a variety of XML processing tools can also generate human readable documentation, which makes it easier to understand complex XML documents.

In general, a [schema](http://searchsqlserver.techtarget.com/definition/schema) is an abstract representation of an object's characteristics and relationship to other objects. An XML schema represents the interrelationship between the [attributes](http://searchsoa.techtarget.com/definition/attribute) and elements of an XML object (for example, a document or a portion of a document). The process of creating a schema for a document involves analyzing its structure and defining each structural element encountered. For example, a schema for a document describing a website would define a website element, a webpage element, and other elements that describe possible content divisions within any page on that site. Just as in XML and [HTML](http://searchsoa.techtarget.com/definition/HTML), elements are defined within a set of tags.

**18) security testing of soap and rest api**

WSDL has four transmission primitives that an endpoint can support:

* **One-way**. The endpoint receives a message.
* **Request-response**. The endpoint receives a message, and sends a correlated message.
* **Solicit-response**. The endpoint sends a message, and receives a correlated message.
* **Notification**. The endpoint sends a message.

WSDL refers to these primitives as **operations**. Although request/response or solicit/response can be modeled abstractly using two one-way messages, it is useful to model these as primitive operation types because:

* They are very common.
* The sequence can be correlated without having to introduce more complex flow information.
* Some endpoints can only receive messages if they are the result of a synchronous request response.
* A simple flow can algorithmically be derived from these primitives at the point when flow definition is desired.

Although request/response or solicit/response are logically correlated in the WSDL document, a given binding describes the concrete correlation information. For example, the request and response messages may be exchanged as part of one or two actual network communications.

Although the base WSDL structure supports bindings for these four transmission primitives, WSDL only defines bindings for the One-way and Request-response primitives.  It is expected that specifications that define the protocols for Solicit-response or Notification would also include WSDL binding extensions that allow use of these primitives.

Operations refer to the messages involved using the **message** attribute of type QName. This attribute follows the rules defined by WSDL for linking (see [section 2.1.2](https://www.w3.org/TR/wsdl#_document-n)).

#### 2.4.1 One-way Operation

The grammar for a one-way operation is:

<wsdl:definitions .... > <wsdl:portType .... > \*

**<wsdl:operation name="nmtoken">**

**<wsdl:input name="nmtoken"? message="qname"/>**

**</wsdl:operation>**

</wsdl:portType >

</wsdl:definitions>

The **input** element specifies the abstract message format for the one-way operation.

#### 2.4.2 Request-response Operation

The grammar for a request-response operation is:

<wsdl:definitions .... >

<wsdl:portType .... > \*

<wsdl:operation name="nmtoken" parameterOrder="nmtokens">

**<wsdl:input name="nmtoken"? message="qname"/>**

**<wsdl:output name="nmtoken"? message="qname"/>**

**<wsdl:fault name="nmtoken" message="qname"/>\***

</wsdl:operation>

</wsdl:portType >

</wsdl:definitions>

The input and output elements specify the abstract message format for the request and response, respectively. The optional fault elements specify the abstract message format for any error messages that may be output as the result of the operation (beyond those specific to the protocol).

Note that a request-response operation is an abstract notion; a particular binding must be consulted to determine how the messages are actually sent: within a single communication (such as a HTTP request/response), or as two independent communications (such as two HTTP requests).

#### 2.4.3 Solicit-response Operation

The grammar for a solicit-response operation is:

<wsdl:definitions .... >

<wsdl:portType .... > \*

<wsdl:operation name="nmtoken" parameterOrder="nmtokens">

**<wsdl:output name="nmtoken"? message="qname"/>**

**<wsdl:input name="nmtoken"? message="qname"/>**

**<wsdl:fault name="nmtoken" message="qname"/>\***

</wsdl:operation>

</wsdl:portType >

</wsdl:definitions>

The output and input elements specify the abstract message format for the solicited request and response, respectively. The optional fault elements specify the abstract message format for any error messages that may be output as the result of the operation (beyond those specific to the protocol).

Note that a solicit-response operation is an abstract notion; a particular binding must be consulted to determine how the messages are actually sent: within a single communication (such as a HTTP request/response), or as two independent communications (such as two HTTP requests).

#### 2.4.4 Notification Operation

The grammar for a notification operation is:

<wsdl:definitions .... >

<wsdl:portType .... > \*

**<wsdl:operation name="nmtoken">**

**<wsdl:output name="nmtoken"? message="qname"/>**

**</wsdl:operation>**

</wsdl:portType >

</wsdl:definitions>

The **output** element specifies the abstract message format for the notification operation.

#### 2.4.5 Names of Elements within an Operation

The **name** attribute of the input and output elements provides a unique name among all input and output elements within the enclosing port type.

In order to avoid having to name each input and output element within an operation, WSDL provides some default values based on the operation name. If the name attribute is not specified on a one-way or notification message, it defaults to the name of the operation. If the name attribute is not specified on the input or output messages of a request-response or solicit-response operation, the name defaults to the name of the operation with "Request"/"Solicit" or "Response" appended, respectively.

Each fault element must be named to allow a binding to specify the concrete format of the fault message. The name of the fault element is unique within the set of faults defined for the operation.

#### 2.4.6 Parameter Order within an Operation

Operations do not specify whether they are to be used with RPC-like bindings or not. However, when using an operation with an RPC-binding, it is useful to be able to capture the original RPC function signature. For this reason, a request-response or solicit-response operation MAY specify a list of parameter names via the **parameterOrder** attribute (of type nmtokens). The value of the attribute is a list of message part names separated by a single space. The value of the parameterOrder attribute MUST follow the following rules:

* The part name order reflects the order of the parameters in the RPC signature
* The **return**value part is not present in the list
* If a part name appears in both the input and output message, it is an **in/out** parameter
* If a part name appears in only the input message, it is an **in** parameter
* If a part name appears in only the output message, it is an **out** parameter

Note that this information serves as a "hint" and may safely be ignored by those not concerned with RPC signatures. Also, it is not required to be present, even if the operation is to be used with an RPC-like binding.

**20) Groovy script**

Top of Form

Bottom of Form

Groovy is an object oriented language which is based on Java platform. Groovy 1.0 was released in January 2, 2007 with Groovy 2.4 as the current major release. Groovy is distributed via the Apache License v 2.0. In this tutorial, we would explain all the fundamentals of Groovy and how to put it into practice.

A variable declaration tells the compiler where and how much to create the storage for the variable.

Following is an example of variable declaration −

class Example {

static void main(String[] args) {

// x is defined as a variable

String x = "Hello";

// The value of the variable is printed to the console

println(x);

}

}

The **for** statement is used to iterate through a set of values. The **for** statement is generally used in the following way.

for(variable declaration;expression;Increment) {

statement #1

statement #2

…

}

**21) soap UI environment tab , what is the use of this**

<https://www.soapui.org/soapui-projects/environment-handling.html>

Web services validating XML over XSD?

Example:

🡪XML schema

<?xml version="1.0" encoding="UTF-8"?>

<!--

Simple XML that represent a computer.

The tag keyboard, mouse and monitor

could be anything

-->

<computer>

<keyboard>105 keys</keyboard>

<mouse>3 button</mouse>

<monitor>vga</monitor>

</computer>

🡪XSD Schema

<?xml version="1.0" encoding="UTF-8"?>

<!--

This schema ensures that the element "computer"

has exactly tree children in that exact order.

The content/attribute of this children is not defined.

-->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="computer">

<xs:annotation>

<xs:documentation xml:lang="it-IT">Definizione di un computer</xs:documentation>

<xs:documentation xml:lang="en-US">Definition of a computer</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:sequence>

<xs:element name="keyboard" />

<xs:element name="mouse" />

<xs:element name="monitor" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

LINK TO VALIDATE ANY XML FILE XSD:

<http://www.utilities-online.info/xsdvalidation/#.WNADC_nys2w>

security in web services

http://www.guru99.com/security-web-services.html