What is web service?

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

what is XML and what will be format?

* 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
* Maybe it is a little hard to understand, but XML does not DO anything.
* This note is a note to Tove, from Jani, stored as XML:
* <note>  
    <to>sami</to>  
    <from>Jan</from>  
    <heading>Reminder</heading>  
    <body>don’t be late</body>

**ResponseText and ResponseXML**

The response can come in many different forms, such as XML, plain text, (X)HTML, or JavaScript Object Notation (JSON). Depending on the data format that you receive, there are two different ways to handle it: with responseText or with responseXML. The responseText method is used for all formats that are not based on XML. It returns an exact representation of the response as a string. Plain text, (X)HTML, and JSON are all formats that use responseText. Using this method on plain text or HTML is trivial:

if(ajax.checkReadyState(’body’, ’loading...’, ’loading...’, ’loading...’) == "OK")

{

document.getElementById(’body’).innerHTML = ajax.request.responseText;

}

Different http methods?

The set of common methods for HTTP/1.1 is defined below and this set can be expanded based on requirements. These method names are case sensitive and they must be used in uppercase.

|  |  |
| --- | --- |
| **S.N.** | **Method and Description** |
| 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. |

what is json and how is the format look like?

[JSON](http://en.wikipedia.org/wiki/JSON) is short for **JavaScript Object Notation**, and is a way to store information in an organized, easy-to-access manner. In a nutshell, it gives us a human-readable collection of data that we can access in a really logical manner.

As a simple example, information about me might be written in JSON as follows:

|  |  |
| --- | --- |
|  | var jason = { |
|  | "age" : "18", |
|  | "hometown" : "harrisburg, MT", |
|  | "gender" : "male" |
|  | }; |

[**view raw**](https://gist.github.com/jlengstorf/2760279/raw/af448c95b9b3616d962388df6da11f261bfc9f86/gistfile1.js)[**gistfile1.js**](https://gist.github.com/jlengstorf/2760279#file-gistfile1-js) hosted with  by [**GitHub**](https://github.com/)

This creates an object that we access using the variable jason. By enclosing the variable's value in curly braces, we're indicating that the value is an object. Inside the object, we can declare any number of properties using a "name": "value" pairing, separated by commas. To access the information stored in jason, we can **simply refer to the name of the property we need.**For instance, to access information about me, we could use the following

The simplest way to check if JSON is valid is to load the JSON into a JObject or JArray and then use the [IsValid(JToken, JsonSchema)](http://www.newtonsoft.com/json/help/html/M_Newtonsoft_Json_Schema_Extensions_IsValid.htm) method with the JSON Schema.

**Validate JSON with IsValid**

1string schemaJson = @"{

2 'description': 'A person',

3 'type': 'object',

4 'properties':

5 {

6 'name': {'type':'string'},

7 'hobbies': {

8 'type': 'array',

9 'items': {'type':'string'}

10 }

11 }

12}";

13

14JsonSchema schema = JsonSchema.Parse(schemaJson);

15

16JObject person = JObject.Parse(@"{

17 'name': 'James',

18 'hobbies': ['.NET', 'Blogging', 'Reading', 'Xbox', 'LOLCATS']

19}");

20

21bool valid = person.IsValid(schema);

22// true

|  |  |  |
| --- | --- | --- |
| **No.** | **SOAP** | **REST** |
| 1) | SOAP is a **protocol**. | REST is an **architectural style**. |
| 2) | SOAP stands for **Simple Object Access Protocol**. | REST stands for **REpresentational State Transfer**. |
| 3) | SOAP **can't use REST** because it is a protocol. | REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP. |
| 4) | SOAP **uses services interfaces to expose the business logic**. | REST **uses URI to expose business logic**. |
| 5) | **JAX-WS** is the java API for SOAP web services. | **JAX-RS** is the java API for RESTful web services |

assertions in soapUI

* HTTP Download all resource - downloads all resources referred to be an HTML document (images, scripts, etc.) and validates that they are all available. Applicable to any property containing HTML.
* Invalid HTTP Status Codes - checks that the target TestStep received an HTTP result with a status code not in the list of defined codes. Applicable to any TestStep that receives HTTP messages
* Not SOAP Fault - validates that the last received message is not a SOAP Fault. Applicable to SOAP TestSteps.
* Schema Compliance - validates that the last received message is compliant with the associated WSDL or WADL schema definition. Applicable to SOAP and REST TestSteps. The schema definition URL supports [Property Expansions](https://www.soapui.org/scripting---properties/property-expansion.html) (e.g.${#System#my.wsdl.endpoint}/services/PortType?wsdl ).
* SOAP Fault - validates that the last received message is a SOAP Fault. Applicable to SOAP TestSteps SOAP Request - validates that the last received request is a valid SOAP Request. Applicable to MockResponse TestSteps only.
* SOAP Response - validates that the last received response is a valid SOAP Response. Applicable to SOAP TestRequest Steps only.
* Valid HTTP Status Codes - checks that the target TestStep received an HTTP result with a status code in the list of defined codes. Applicable to any TestStep that receives HTTP messages.
* WS-Addressing Request - validates that the last received request contains valid WS-Addressing Headers. Applicable to MockResponse TestSteps only.
* WS-Addressing Response - validates that the last received response contains valid WS-Addressing Headers. Applicable to SOAP TestRequest Steps only.
* WS-Security Status - validates that the last received message contained valid WS-Security headers. Applicable to SOAP TestSteps.

xpath from w3schoo?

* 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 the XSLT standard
* XPath is a W3C recommendation

how to read attributes in xpath?

|  |  |
| --- | --- |
|  | Given an XML structure like so:  <?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> |

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

I'm trying to learn xpath. I looked at the other contains() examples around here, but nothing that uses an AND operator. I can't get this to work:

//ul[@class='featureList' and contains(li, 'Model')]

On:

...

<ul class="featureList">

<li><b>Type:</b> Clip Fan</li><li><b>Feature:</b> Air Moved: 65 ft.

Amps: 1.1

Clip: Grips any surface up to 1.63"

Plug: 3 prong grounded plug on heavy duty model

Usage: Garage, Workshop, Dorm, Work-out room, Deck, Office & more.</li><li><b>Speed Setting:</b> 2 speeds</li><li><b>Color:</b> Black</li><li><b>Power Consumption:</b> 62 W</li><li><b>Height:</b> 14.5"</li><li><b>Width:</b> Grill Diameter: 9.5"</li><li><b>Length:</b> 11.5"</li>

<li><b>Model #: </b>CR1-0081-06</li>

<li><b>Item #: </b>N82E16896817007</li>

<li><b>Return Policy: </b></li>

</ul>

...

#### 