**SOAP UI**

**Web Services:** Web services are web components that transfers data between client and server. Client sends a web request to the server and the server then responds to client. This response will differ based on the web service request type.

Soap UI is designed for validating web services easily.

**WSDL- Web Service Description Language**

A WSDL is a document that should be written using XML. This document describes the following details about the web service:

* Origin of the web service
* Header information
* Port type
* Input and output messages

Tags in WSDL:

* **<types>** – XML Schema data types
* **<message>** – the actual request and response data being communicated
* **<portType>** – the target / end points where the actual web service is hosted to perform the operation
* **<binding>**– the protocol information is given for the data format
* **<definitions>**– the parent tag for the above mentioned tags

Validating web services using SoapUI is easy and is only possible with WSDL document because to configure web services in SoapUI, WSDL document is mandatory. If the WSDL document is not valid, SoapUI will throw an exception immediately

**SOAP – Simple Object Access Protocol**

It uses XML based data to interact with web applications.

* SOAP is language and platform independent as it is written by using XML.
* It creates the platform to communicate with the applications that are running in different operating systems using different technologies.
* Most of the Internet applications interact with each other over Remote Procedure Calls that use [DCOM](http://en.wikipedia.org/wiki/Distributed_Component_Object_Model) (Distributed Component) and [CORBA](http://www.corba.org/) (Common Broker Architecture)
* These technologies are different than the HTTP.

RPC (Remote procedure calls) are sometimes blocked by firewalls and proxy servers. To overcome these issues, SOAP was designed.

**SOAP document must contain the following elements:**

1. **Envelope element** is the top most tag which identifies the XML document as a SOAP message.
2. Followed by Envelope element, you see the**header element** that has header information.
3. The **Body element** specifies the call and response information.
4. Finally, you have a **Fault element** which contains errors and status information.

A protocol is a set of standard rules that transfers the data between two regions in the Internet over the web services. There are many protocols that are used in the Internet applications. They are, Transmission Control Protocol (TCP) which serves as a packet between two connections. Internet Protocol (IP) that sends and receives the messages between two destinations.

* Hyper Text Transfer Protocol (HTTP)
* File Transfer Protocol (FTP)
* Border Gateway Protocol (BGP) and
* Dynamic Host Configuration Protocol (DHCP)

**RDF (Resource Description FrameWork)**

RDF contains the description of the web resources such as title, author, content, and copyright information. This framework was designed so that computers can be read and understood easily by the web. RDF is completely written by using XML language.

RDF data can be transferred between different types of computers using different operating systems and programming languages. RDF uses Uniform Resource Identifiers (URIs) on the web and it describes the resources along with the property and property values.

**XML**

XML (e**X**tensible **M**arkup **L**anguage) is a mark-up language that is used for storing, sharing and formatting data.

**How XML works with SoapUI?**

As XML is a common language on Internet, it can be integrated with SoapUI because web services are mostly written in the form of XML. Also, if we pass XML input parameter to the web service, the response itself will be in the form of XML. SOAPUI can configure these web services.

**Features Of SoapUI and SoapUI Pro**

* User Friendly GUI
* Easy for Functional Testing
* Vulnerability Testing
* Load Testing
* Automation with Groovy
* Data Driven Testing
* Assertions

**SoapUI Pro**

* Complete functional testing capability for SOAP API, REST and other protocols
* Ready API platform which determines the actual functionality of the API service and its expected behavior.
* It allows ad-hoc testing or command line interface to test our APIs effectively.
* All the REST, SOAP API and other service components can be used by simply drag and drop method
* In SoapUI NG Pro, data driven feature is little enhanced in retrieving information from external data sources for example, excel, XML, JDBC data sources and file / directories etc. Then these retrieved data will be converted into SoapUI NG Properties test step.
* We can transfer the property test step values to xpath-queries, scripts and so forth.
* SoapUI NG Pro offers the feature called **point-and-click** to generate test scenarios quickly
* SoapUI NG Pro allows the end user to customize their services easily even they are new to SoapUI Pro or development experience.
* Test Coverage: To analyze the API tests along with the functionality as expected
* Multi-environment Support: Allows to change the testing environment based on our requirements
* Test Debugging: This feature helps to analyze the test step-by-step debugging. It also includes variables, properties, input requests etc.
* Complex Scenarios: SoapUI NG Pro makes it easier the APIs which are involved in client server architecture
* Drag and Drop Test Creation: As it exists, it is easy to create and run the test scenarios by drag and drop feature
* SoapUI team also introduced LoadUI NG tool for LoadUI Pro users. It is used for performing load testing on Ready! API platform. It basically simulates the SoapUI NG Pro test cases and determines the load of the application server

**SoapUI Assertions:**

Compares the parts/all of the response message to the expected outcome.

Each type of assertion targets specific validations on the response such as matching text, comparing XPATH or we could also write queries based on our need.

When the test steps get executed, then the associated assertions receive the response for the respective test steps. If any response is failed then the respective assertion will be processed and the corresponding test step will be marked as failed.

* Contains and Not Contains assertions
* XPath match and
* XQuery match assertions.

**Properties:**

Properties are used to store and retrieve the datawhile executing the test cases. Internally property will store the value in key pair format.

**Accessing property:**

Get Property Value

def getTestCasePropertyValue = testRunner.testCase.getPropertyValue(“LocalPropertyName”)

def getTestSuitePropertyValue = testRunner.testCase.testSuite.getPropertyValue(“LocalPropertyName”)

def getGlobelPropertyValue = com.eviware.soapui.SoapUI.globalProperties.getPropertyValue(“GlobalPropertyName”)

Set Property Value

testRunner.testCase.setPropertyValue(“LocalPropertyName”, value)

testRunner.testCase.testSuite.setPropertyValue(“LocalPropertyName”, value)

com.eviware.soapui.SoapUI.globalProperties.setPropertyValue(“GlobalPropertyName”, value)

**Receiving Response Data:**

* Execute the service request once and verify the result
* Go to Groovy script editor and then right click on the editor as shown in the below screenshot

SoapUI Pro generates the script as below after specifying the property name.

def response = context.expand( ‘${ServiceRequest#Response}’ )

“**def**” is a groovy script keyword that represents defining properties / objects. By default, SoapUI Pro has the property name as “**response**” in the **Get Property** popup.

def response = context.expand( ‘${ServiceRequest#Response}’ )

testRunner.testCase.setPropertyValue(“Pro\_Response”, response)

def getLocalPropertyValue = testRunner.testCase.getPropertyValue(“Pro\_Response”)

log.info(getLocalPropertyValue)

it will log the entire response data in the **log** section.

**Types of properties in SoapUI:**

1. **Default Properties**: included in the SoapUI installation. We can edit some of the default properties but not all.
2. **Custom/user-defined properties**: These are defined by us at any level needed, such as global, project, test suite, test case or test step.

**Various property levels in SoapUI Pro:**

**Level #1. Project and Custom Properties**

**Level #2. Test Suite and Custom Properties**

**Level #3. Test Case and Custom Properties**

**Load Properties from External Source:**

* Add Properties test step under the test case
* Enter the property step name and then click OK
* In the property panel under the navigation panel, click Custom Properties tab
* Click [Working with Properties](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2015/05/Working-with-Properties.jpg) icon to load the properties from the external property file

Transferring properties among test steps is easier than writing repeated scripts to create new. SoapUI also gives the wonderful feature to import and export properties. This feature will be useful when we are using common properties such as login details, session details etc., for multiple projects. That way, we don’t have to create the same properties again and again for multiple projects. We can simply change in the property value against the properties based on the projects.

**Groovy Script:**

**Conditional statement:**

* If…else statement
* If…else if…else statement
* Ternary operator
* Switch statement

### Looping or Iterative Statements:

* For loop
* While loop

### Arrays Collection:

### Array collection helps store multiple values in a single variable or object. Array index starts at zero by default and we need to use that index with array name to access the corresponding value stored in the array. Syntax to declare arrays in groovy script:

### arrayName = new Object[5] or, arrayName = new Object[10][] or,  arrayName = new Object[10][10][]