**SOAPUI**

Java –jar Prathap.jar –server.port=8086

**API:** Application Programming Interface

**Web Service:** It is a method of communication b/w two applications or electronic devices over the www. It is collection of operations.

**The difference between API and WebServices is:**

**API:** Pure Methods, which available in the system they given to us like Jars, We don’t have access to their code.

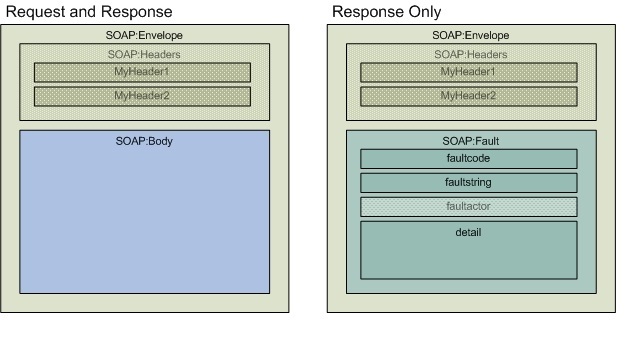
**Web Services:** Over the Network by HTTP call

**Data Collector:** JSON, XML

**Authentication:** Session Key/Token

**WSDL(Webservices Defination Language):**

* It is an contract for web services
* It has location of web services
* It has different operations what that particular web service support.
* It has data types of using in web services, data elements for each elements of that.
* It is in XML format in below structure.



**Types of web service:**

**i) Soap(simple object access protocol)**

If we communicate web services by **soap** protocol is call “soap web services”.

It accepts soap protocol messages.

It uses XML as underline language

**ii) Representational State Transfer (REST)**

It expects data in rest Style and response back in rest style.

It uses JSON language

It is lightweight

|  |  |  |
| --- | --- | --- |
| S.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. |
| 6) | SOAP defines standards to be strictly followed. | REST does not define too much standards like SOAP. |
| 7) | SOAP requires more bandwidth and resource than REST. | REST requires less bandwidth and resource than SOAP. |
| 8) | SOAP defines its own security. | RESTful web services inherits security measures from the underlying transport. |
| 9) | SOAP permits XML data format only. | REST permits different data format such as Plain text, HTML, XML, JSON etc. |
| 10) | SOAP is less preferred than REST. | REST more preferred than SOAP. |

**SOAP UI TOOL:**

* Down load soap ui tool and install it
* Click on **Soap** in home page
* Create New project
* And Browse for WSDL file (or get URL)
* Click Ok

1. **Creating Test Suite:**

* Right Click On Project and select new test suite
* Give test suite name
* Right click on test suite
* Select test Case and give name
* In test case we have test steps
* Right click on test steps
* Add step
* Right click on step select add step then select soap request
* Give name and select operation what u will test

**TestExampleProjectWith**

[**http://216.10.242.178:8082/axis2/services/EmployeeManagementService?wsdl**](http://216.10.242.178:8082/axis2/services/EmployeeManagementService?wsdl)

**and DB**

1. **SOAP**

It is a protocol for sending and receiving messages b/w applications without confronting interoperability issue.

**Structure Of Soap:**

Soap Envelope

Soap Header

Soap Body

Soap Fault

**SOAP Manual Testing By SOAPUI**

**Basic Assertions for Test validations**

* Add Project
* Create Testsuite
* Create Testcase
* Right click on teststeps
* Add steps 🡪 soap reqst
* Ok 🡪 ok 🡪 ok
* Add employee Details
* Click On Assertion(green+ symbol on top)

**// Contains assertion**

* Click on “Contains”
* Type “True”
* Click on “Assertions in the bottom”
* Click on contains valid
* Then type other word thn true, it fail

**// Not Contains assertion**

* Click On Assertion(green+ symbol on top)
* Click on property content
* Click on not contain, ok
* Enter false, it pass

**// Soap response assertion**

* Click On Assertion(green+ symbol on top)
* Click on compliance, status
* Select soap response 🡪 ok

**// Http Code**

* Click On Assertion(green+ symbol on top)
* Click on compliance, status
* Select “valid http status code”
* Ok, enter 200( to check http value, in right side execution window, select raw and see)
* It pass

**// SLA**

**To check response time of result**

* Click on sla
* Click on response sla
* Give 200

**// Security**

* Click on security
* Click on sensitive info
* Click on green +
* Enter random
* Enter “info”

1. **Different Types of Assertions:**

**XpathMatch:**

To check Assertion by using Xpath (To check value EX: age = 26)

* Click On Assertion(green+ symbol on top)
* Click on property content
* Click onXpath Match, ok
* Give name
* Click on Declare
* Enter Xpath for your operation (either absolute or relative xpath)
* Enter your expected result in below
* Click on Save

**To Check particular node is Exists or not (Ex: age)**

* In above xpath window enter
* Exists (xpath of node) [Example: **exists(//ns: age)]**
* Enter expected result
* Click save

**To Check only one instance of node present:**

for example only name node is one

* In above xpath window
* Count(xpath of node) **[example: Count(//ns:name)]**
* Enter expected result
* Click save

**To Check Dynamic values:**

**By using Wildcards give \* inthat particular keep changing value place**

**Properties in SOAPUI:**

**1. Test Suite/ Test Case Level Properties Access:**

It is used to avoid hardcore value in request, we have to create data in test case level.

* Go to particular test case
* In the bottom click on custom properties
* Enter Key and value pair (name = prathap)
* In request where you want to pass name give right click
* Select get data
* Select test case and select property

Syntax for Test suite: ${#Testsuite#property} (ex: ${#testsuite#name})

Syntax for Test Case: ${#Testcase#property} (ex: ${#testcase#id})

**2. Project Level Properties Access:**

Same like above method, but we have to give properties in project level

Syntax for Project: ${#Project#property} (ex: ${#Project#id})

**3. Properties Level:**

If you would like to give all data(properties) in single sheet then we have to use This properties level

* Go to test case
* Give right click
* Click on add steps
* Click on properties
* Give name
* Then enter properties

To retrieve this go to particulat request and right click

* Get data
* Step2[properties]
* Property[name]

Syntax: ${Properties#Department}

This is not generic keyword from soapui, so we dnt have # like above.

**4. Properties upload from External file:**

* The file in .properties extension only
* Go to where you want to use this file like project/ test suite or test case level
* Click on custom properties on bottom
* Click on upload external file symbol
* Browse file
* Add file
* And use

**Property Transfer Functionality:**

When we would like to clone all test cases into single step, we use this functionality.

* Go to Particular Test Case Request (Get Employee(soap request)
* Right Click
* Clone Test Step
* Give name and select test step where you would like to clone
* Ok
* Do this step for all test steps which we would like to add

**When we have delete employee by depending on get employee details we have to do property transfer.**

* First go to testcase
* Right Click, Insert step
* Select Property Transfer
* Give name
* Ok (at the same time go to test suite and give custom property name(ex: property transfer).
* And Click on green + symbol to add new property transfer
* Select source and Target
* And select property for both target and source
* And go to your source (get employee response window)
* Copy ns url(ex: ns=<http://sample.com/reservation/guest/types>)
* Go to property transfer window
* Click on ns on top two green signals(>>)
* Then enter ns url in source
* And write Xpath for name property(which property u are going to use)
* EX: //ns:getEmployeeDetailsResponse/ns:return/ns:name/text()
* {text() is for grab the name(reddy) from xpath)
* And give above url in Target window also
* Select property as request(because we have to send above name in delete request)
* And give xpath where above text() has to go

**We can achieve property transfer by manually:**

If I would like to name from add employee to get employee.

We can write like

${fromtestcasename #whereyouare getting text #xpath of text}

**EX: Syntax**

**${addemployee#Request#//typ:addEmployee/typ:name/text()}**

**SOAP Automation Testing By SOAPUI**

* Create Test suite and Test cases etc.. like above
* After adding test steps and one more test step as groovy Script
* Goto test step
* Right click—Add step
* Select Groovy Script
* Give name
* Add ok

**Groovy Basics:**

Log.info: to print something in console

Context: Used to access and modify the properties of test cases, it will work inside the particular test case only.

TestRunner: It will helps to access properties across the project level.

**To retrieve the data by using testrunner from different test case level, we have to write like:**

TestRunner.testCase.testSuite.testCases["whichtestcaseyouliketogetdata(Ex:getEmp)"].getPropertyValue("id")

**If you have property value in same test case then:**

testRunner.testCase.getPropertyValue(“id”)

**If we have to get property from test step of other test case:**

testRunner.testCase.testSuite.testCases["GetEmp"].testSteps["Get"].getPropertyValue("Request")

**To set value in groovy by using Set:**

testRunner.testCase.testSuite.testCases["GetEmp"].setPropertyValue("id", 4567).

**To Retrieve Custom Property from project level:**

def Project = testRunner.testCase.testSuite.project

Project.getPropertyValue("Gender")

**End to End Testing**

xmlHolder is an library which supports to pass data into xml.

To add data into requestxml, goto that location:

DefaddReqtestRunner.testCase.testSuite.testCases["AddingEmp"].testSteps["Adding"].getPropertyValue("Request")

def xmlAdd = new xmlHolder(addreq).

**We can use setNodevalue method to insertdata into xml**

xmlAdd.setNodevalue(“xpath of node”, “value of node”)

**REST Manual In SOAPUI**

**HTTP Methods:**

C—Create--POST  
R—Retrieve--GET  
U—Update--PUT  
D—Delete—Delete

HTTP Request:

* URI
* Headers
* JSON/XML Payload

HTTP Response:

* Status-200 ok
* Response Payload
* String Message-successful

**POST: To Create new Entity in server or DB**

* JSON/XML form

**GET: To Retrieve (Fetch) Data from server/DB**

* URI
* Path Parameter
* Query Parameter
* Headers

**PUT: To Update the info**

**DELETE: To delete data.**

**URI (Uniform Resource Identifier) = URL+HOST+PORT+(Path parameter/query parameter)**

**URI= URL(Endpoint URL)+ (API) Service URL**

**EX: https://reqres.in/api/users?page=2**

**? = Query parameter**

**Page : Parameter**

**=2: parameter Value**

**Status Response Code:**

* **200 Ok**
* **201 Created success**
* **400 Bad Request**
* **401 Unauthorized(authentication error)**
* **404 Not Found**
* **500 internal Error**

**IN SOAPUI:**

* Create REST Project
* Enter End Point
* Create Test Suite/ test case/ test steps for each request
* We can pass entire url in above end point or we can enter in request body
* Do validations and property Transfer as like above (soap services)

**REST Automation In SOAPUI**

* Create REST Project
* Enter End Point
* Create Test Suite/ test case/ test steps for each request
* We can pass entire url in above end point or we can enter in request body

**We can perform assertion two ways by assertion like above and by Script assertion**

* By using script assertion window we can avoid groovy script window.
* We can write scripting in script assertion window only.
* But it won’t support **Test Runner** variable, we can use log, context
* We have to

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**GroovyScript TestStep:** We can update xml, json. We can push back into request, trigger testcases.

**Script Assertion:** We can only write script for validate your responses.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**JSON Assertions:**

**SoapUI Pro Features:**

**Pro Features:**

1. **Data source**

Uses for store data manually or store from excel sheet

1. **Data source loop**

Uses to run multiple sets of data which we stored/imported from excel in data source step

1. **Data Sink**

Which we use to export response to an excel sheet

1. **Data Gen**

To randomize input to parameter use, it uses any value from list

* 1. List
  2. Number (Random numbers, with interval 2 etc..)
  3. Template 🡪 It does concatenate the lines data

1. **Test Case Coverage**

Depends on WSDL or Rest Resources, how many services we utilized from them in %.

1. **Test Case Debugging**

By keeping a break point at test step in test case we can debug the issue. We get star if parameter changes in step.

1. **Test on Demand**

We can run our test cases in different locations (UK, Chicago etc..)

1. **Test Case Requirement**

Giving US#, AC etc.. We can store related to test case in future we don’t get confuse

1. **Create Report:**

Have 3 types of reports

* 1. Test case report
  2. JUnitStyle-HTML report
  3. Data export

1. **Record and play**

**Database Testing:**

Crete a data base.

CREATE DATABASE DATABASENAME

EX**: CREATE DATABASE** PRATHPDB

USE DATABASENAME

CRAETE TABLE Training (name VarChar(20), course VARCHAR(20), location VARCHAR(20), skill VARCHAR(20));

INSERT INTO Training VALUES(‘prathap’, ‘SopaUI’,’US’, ‘Perf’);

**JDBC In SoapUI:**

Driver: com.microsoft.sqlserver.jdbc.SQLServerDriver

**ConnectionString :** jdbc:sqlserver://Csvt0000ca35.tus.ams1907.com:23100;user=tmeconnect;password=TmEdatabase2018

**Assertion:**

JDBC status

JDBC Timeout

**SoapUIPro feature for DB:**

Build Query: If we select rows, soapui builds query for us

Every JDBC Step as a option “Response as xml”, which gives response in XML.

By using this we can pass a value from other test steps to one test step

In query, by giving ‘’ and right click in b/w will select response as xml and select value

**Performance Testing:**

Have to select strategy what kind of test we need.

Simple: Specific number of threads, without fluctuations.

Burst: Burst of time. Lot of users hit at time, ramp up

Threads: Amount of threads, with increment, aggregate depends on time.

Variance: Randomize amount of load

**Threads**: Amount of concurrent users hit services at same time

**Test Delay:** wait time b/w concurrent users.

**Ex:** If we give 20 users, after execute 20, it wait amount of breathing time, then it execute again.

**Random:** The random factor of the Test Load. With a Test Delay of 600 ms and a Random factor of 0.5, the actual delay will be uniformly distributed between 300 ms and 600 ms. The actual delay can thus be calculated as *TestDelay - RandomNumberBetween( 0, TestDelay\*Random )*.

**Limit:** time for load on services

**SoapUI suggested use with 100 concurrent users. Better to use Load UI(License)**

**Have different validation for load test:**

Step TPS

Max Errors

Step Avg:

Step Max

Step Status

**JMETER:**

**TestPlan** : select Thread Group give details like no.of threads, **ramup period**, loop count.

Then go to **sampler** and HTTP Request.

And give details like server name, port, path(with XML/JSON) and parameters.

Then add **Listener** with Result Tree/aggregate report/Graph results

**WorkBench**

**SOA VS Web Services:**

1. **Monolithic (Single Unit):** Hosting number of components, hosted together and deliver together. It’s not scalable. It’s time taking and not fit for complex architecture, we cannot use diff technologies at a time.
2. **SOA (Coarse-Grained Architecture):** The features brokendown into smaller components, each feature may contain number of services.
3. **Micro Services (Fine-Grained):** The services broken down into task level pieces**.**

**EX:**

**SOA** is like an orchestra where each artist is performing with his/her instrument while the music director guides them all.

**Micro Services**: Dance group, each dancer independent and they know what they need to do. If they miss some steps they know how to get back.

**Difference b/w with respect Certain Parameters:**

1. **Architecture and Co-ordination Difference:**

**SOA Have:**

1. **Business Service:** Its core business operations, represented by XML/WSDL.
2. **Enterprise Service:** Implements the functionality defined by Business service**,**
3. **Application Service:** This is core functionality of the feature, can be invoked directly or through UI.
4. **Infrastructure Service :** It goes to non-technical/Non-functional features, like security, performance and scaling

These 4 services together form SOA/deliver a software product

**Micro Services Have:**

1. **Functional Service:** It’s a combination of Business, enterprise and application services.
2. **Infrastructure Service:** It goes to non-technical/Non-functional features, like security, performance and scaling.
3. **Heterogeneous Interoperability:**

**SOA:**

If Services are different technologies, it’s difficult to interact to one and other, so we need Messaging middle ware. It act as bridge b/w different apps which are in different languages.

**Micro Services**:

No need any broker, it can communicate any kind of apps with API Layer.

1. **Service Granularity:**

**SOA:**

The service consumer doing one operation, multiple services call at one time

**Micro Services:** One operation may involves multiple services or called by one and other. **1>2>3**

1. **Component Sharing:**

**SOA:** We have a service which pull data from different DBs and give to user for different features.

**Micro:** We have different services which pull data from different DBs and give to user for different features.