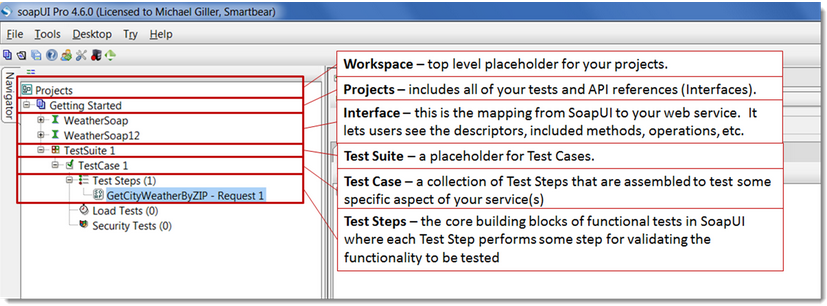
# Soap UI Pro

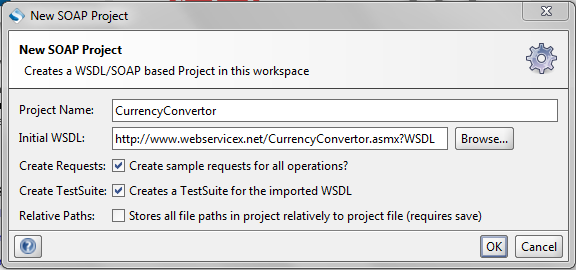
## Structure:



## Create Sample Project

1. In soap UI > File > New Soap UI project.
2. Load WSDL,

WSDL Sample: <http://www.webservicex.net/CurrencyConvertor.asmx?WSDL>

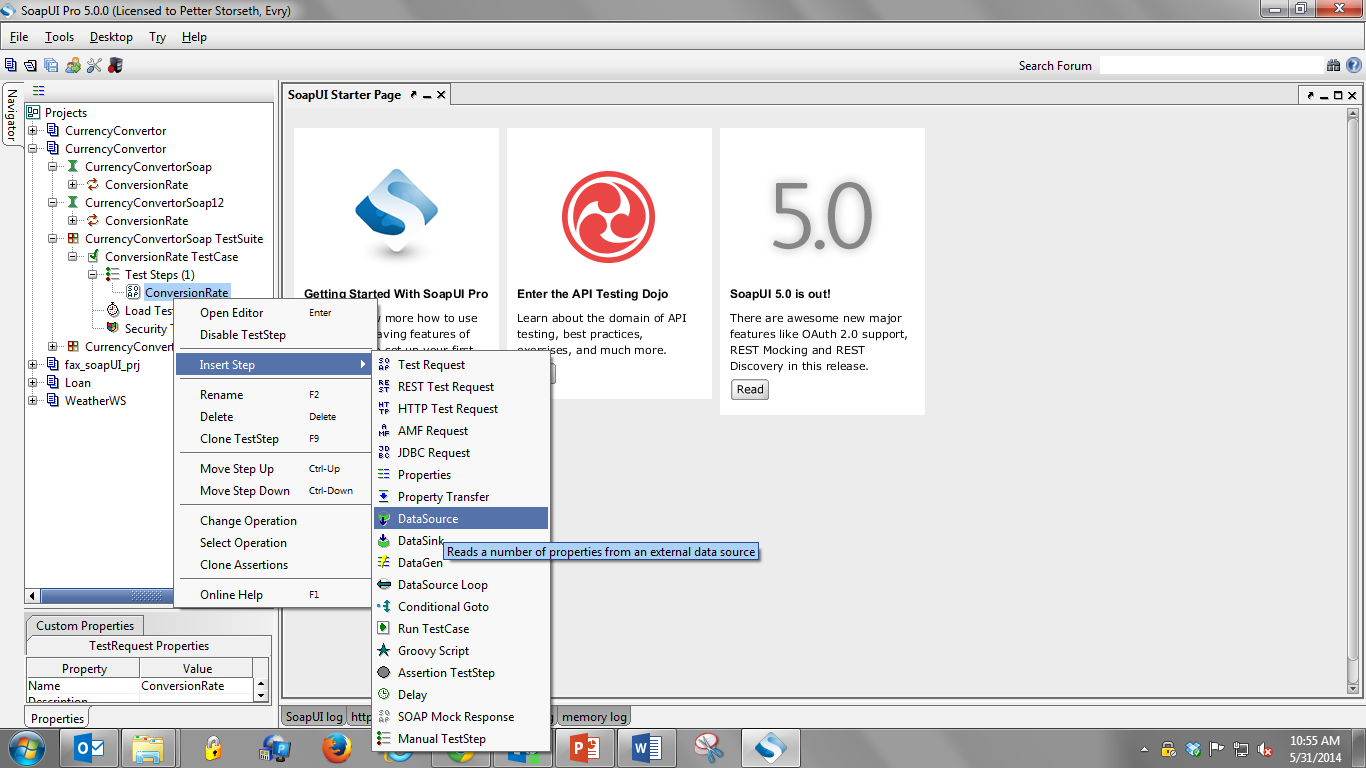


1. Click OK…
2. WSDL for many Web Services are available at <http://www.webservicex.net/ws/default.aspx>,
3. Open WSDL in browser for confirmation (Availability).

# Data Driven Testing in SoapUI Pro.

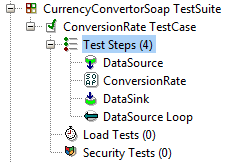
## Add Test Steps

1. Data source:



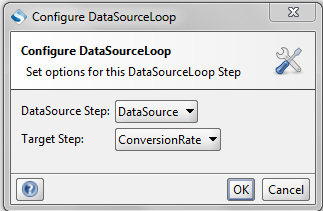
* Right click on test step
* Click “Add Step” OR “Insert Step”
* Select the test Step To be added as shown below.

1. Similarly add **Data Sink** and **Data Source Loop**
2. Rearrange test steps (Drag and Drop) as shown



* **Data Source loop** test step loops back the test step for next iteration.

1. Data source loop setting:
2. Data source Step is the Data Source Test Step
3. Target Step Is the next Step to Data Source.

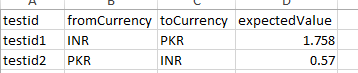


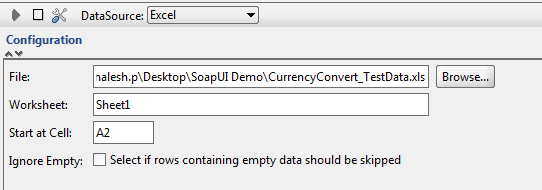
1. Configure Data Source.

NOTE 01: EXCEL SHEET MUST BE SAVED AS “.xls” (Excel 97-2003 Workbook) FORMAT

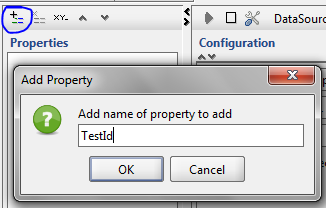
NOTE 02: Expected results are added to dataSource excel sheet for simplification in example, ideally this value should be fetched from database.

* Open Excel sheet > File > Save As > Save As Type: Excel 97-2003 Workbook
  + Select Data Source as Excel, as shown below
  + Browse the Excel File location
  + Give Work Sheet name
  + Start At Cell: A2 – second row, (first row for Header will be ignored)





* Add Data Source Properties (Header Names):
* Click add property and Give name:



* Add all header names to properties.
* Run data source by clicking on the Green arrow on the top

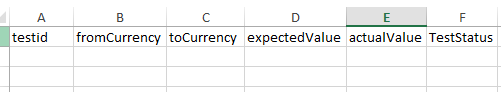


Configure Test Request (Currency Convert) Test step:

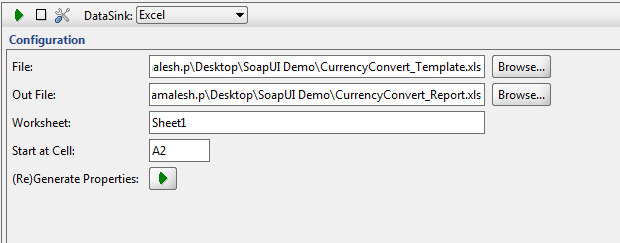
* Right click input field, Get Data > Data Source > Select Property Name.

Configuring Data Sink

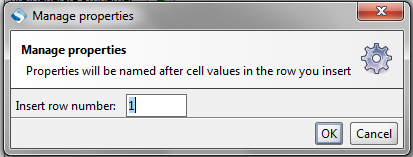
* Similar to Data Source
* File: can be a template (Empty File).



* Out File can be same as File or different file.

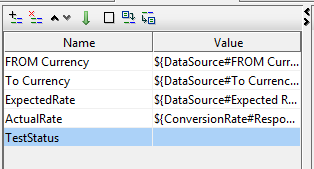


* (Re)Generate Property will add all header names to property list from file.



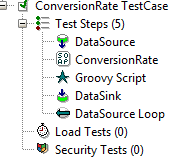
**NOTE:** some version of SoapUI tool may have issue while fetching column names from excel in dataSink, use SoapUI version 5.0.0.

* Configure property by right clicking > Get data > test Step > Property name.



Validation using Groovy Script:

* Add Test Step Groovy Script:



Add Script:

//declare Groovy Util context

def groovyUtils = new com.eviware.soapui.support.GroovyUtils( context )

//declare variable for Expected Rate

def expectedRate = context.expand( '${DataSource#Expected Rate}' )

//declare response Xml holder

def holder = groovyUtils.getXmlHolder("ConversionRate#Response")

//Get value From Response XML using XPATH

def actRate = holder.getNodeValues("//\*:ConversionRateResult")

//To write Data to data sink

def runnerComplete = testRunner.testCase.testSteps.DataSink

//compare expected and actual values

if (expectedRate.equals(actRate)){

TestStatus = "Pass"

} else {

TestStatus = "Fail"

}

//write result to dataSink

runnerComplete.setPropertyValue("TestStatus",TestStatus)

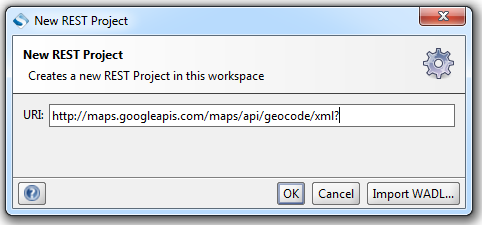
**For Further details on Groovy Script refer,**

<http://www.soapui.org/Scripting-Properties/tips-a-tricks.html>

# Rest Services

## Creating a sample rest project:

1. In soap UI > File > New REST Project.
2. Provide the URL: (can include parameters in URL or without parameter)



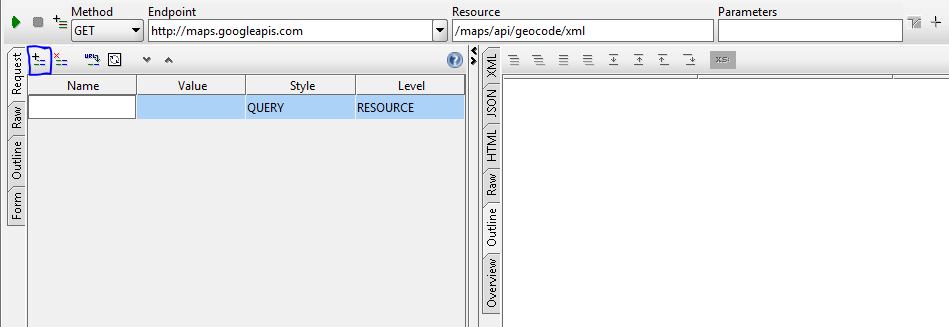
URL with parameters:

<http://maps.googleapis.com/maps/api/geocode/xml?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&sensor=false>

URL without parameters:

<http://maps.googleapis.com/maps/api/geocode/xml>?

1. Adding parameters to request:



1. Create test suite
2. Add groovy script to build a JSON object:

//define a JSON builder

def builder = new groovy.json.JsonBuilder()

//declare test request (used to set JSON content to request)

def request = testRunner.testCase.testSteps.Request1.getTestRequest()

//declare variables from data source to be used in JSON request

def FromCurrency = context.expand( '${DataSource#FromCurrency}' )

def ToCurrency = context.expand( '${DataSource#ToCurrency}' )

// log.info prints the variable value

log.info FromCurrency

//Build the Request

builder{

// condition if parameter is optional input field (declared in JSON object only if it has value)

If (FromCurrency != “”) {

fromCurrency FromCurrency

}

toCurrency ToCurrency

}

//print the JSON object into request body.

request.setRequestContent(builder.toPrettyString())

1. Response validation is same as soap response validation, only difference is while declaring XML holder for response.

//declare response Xml holder

def holder = groovyUtils.getXmlHolder("ConversionRate#ResponseAsXML")