

1. **Introducing SoapUI Pro** (*must read*)

Introducing SoapUI Pro:

SoapUI comes in two version- Open source free (SoapUI) and SoapUI Pro (now SoapUI NG pro).

*So far we were discussing the common automation concepts and functional testing basics such as creating projects, test suite/case/steps, properties, assertions, groovy script programmingetc. **You can check all these tutorials from this SoapUI series onthis page.***

Now, it is time for us to examine SoapUI Pro and its salient features which are more refined and are targeted at better and faster testing.

Note that we have not discussed features of SoapUI NG Pro version here. Apart from the few new features, all the features discussed below are also present in SoapUI NG pro.

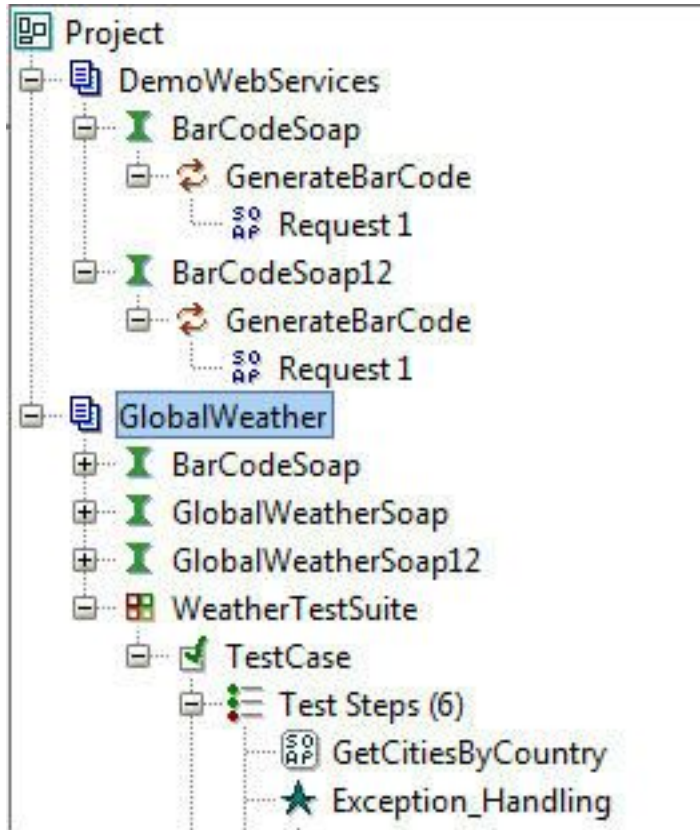
4 Important Features of SoapUI Pro

Feature #1: Point to Click (Drag & Drop):

This enables test steps in a certain test suite to be cloned easily. This will let you duplicate work without having to recreate it.

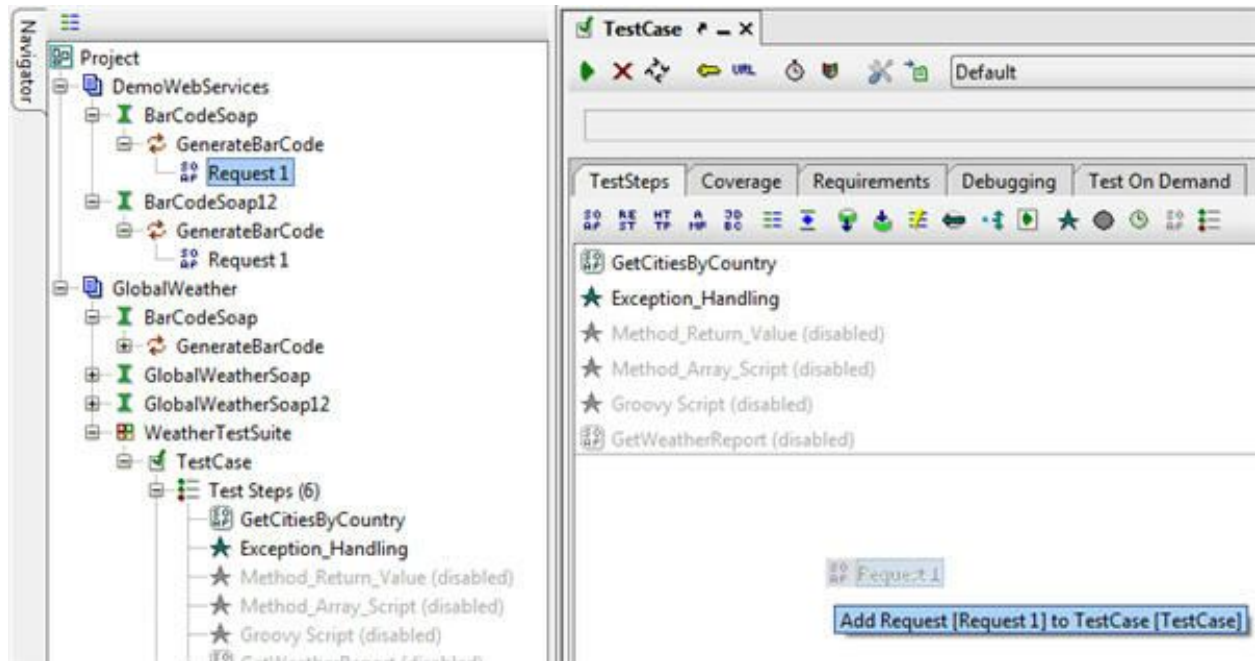
Here is how you can do it:

Make your project tree as below. We are going to add test request by dragging and dropping in to another project.

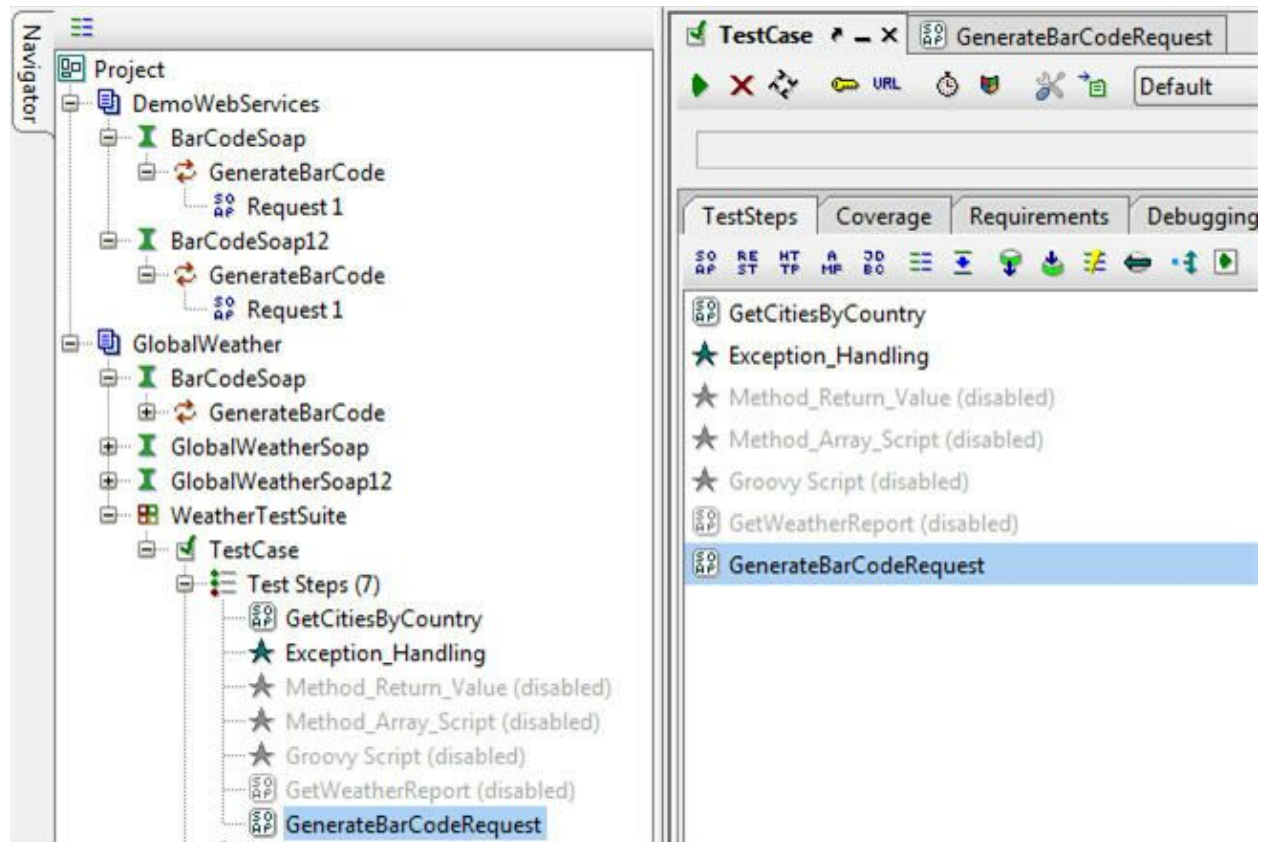


- Double click on **Testcase** node present under test suite from **GlobalWeather** project tree
- Then drag the **Request1** from **DemoWebServices -> BarCodeSoap** tree and drop into test case screen.
- Take a look at the following screenshot to get picture better idea.

(Click image for enlarged view)



- SoapUI Pro will ask us for confirmation. Choose Yes.
- We will see Add Request to **Testcase** dialog where we need to enter new request name
- Let me enter "**GenerateBarCodeRequest**" in the request text field.
- Once it is done, click OK to save. Here's the screenshot that shows newly added test case under the test suite.



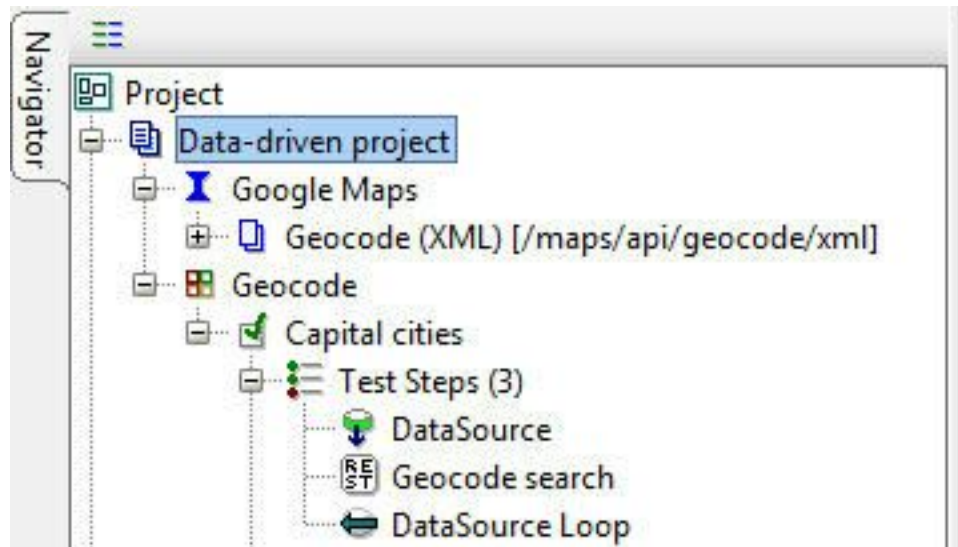
Similarly we can add test suite or other nodes from one project to another project just by dragging and dropping.

Feature #2: Data Driven Testing

SoapUI Pro uses the data source test step that will connect external data source and feed the data to the web service. Data source test step comes with data source loop that enables iterating the data and send it to the web service. This feature is useful for both functional-data driven and load testing.

The data sources supported are: Excel, JDBC, XML and any other compatible databases. More than one data sources can be connected at one time.

Here is an example – A data source project will look as below:



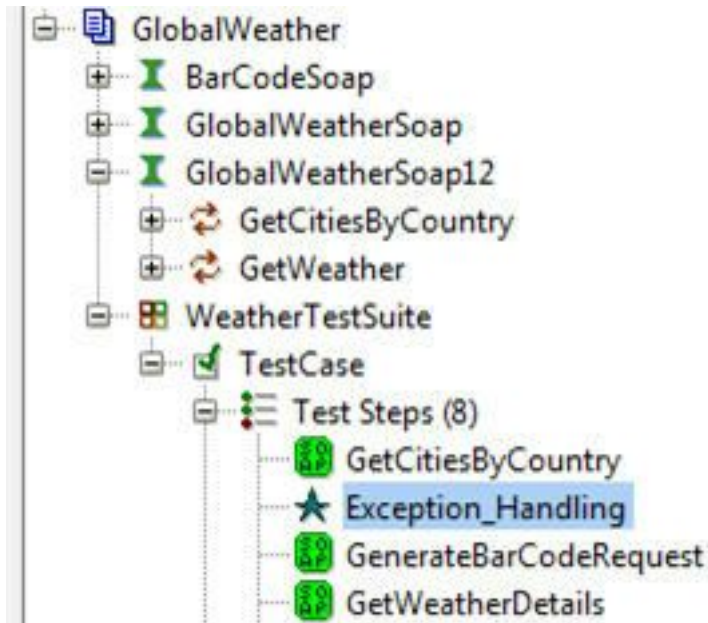
Feature #3: Reporting


SoapUI Pro offers different types of reports for better and easier analysis. They are:

- **Printable Report** – It allows us to export as PDF, HTML, RTF, Excel and so on.
- **Data Export** – can be used to extract specific data as XML and CSV formats.
- **HTML Reports** – generates the result in an HTML format that can be published on any web page.

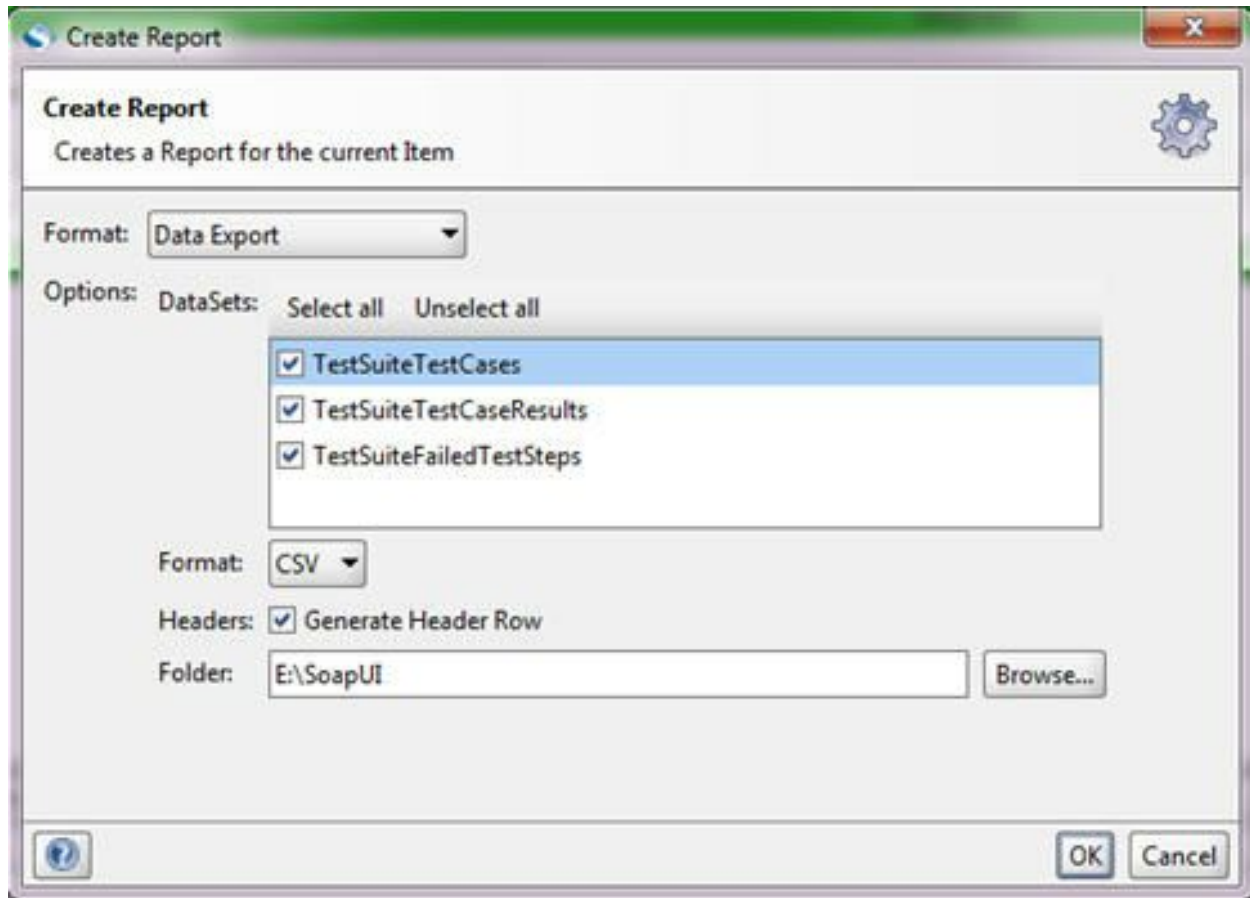
Now let us create sample report for the test suite execution. Follow these steps:

- Create a project with <http://www.webservices.net/globalweather.asmx?WSDL>
- And then add test suite and test steps as shown in the following screen shot




- Once the test requests are configured properly, double click on the test suite
- Click on the run button from the toolbar to start the execution
- SoapUI will start the execution and finally shows the test suite execution status
- Now click on the **Create Report** icon  from the test suite toolbar.
- It will ask you to select the report type that you wish to create
- Make the required changes and then click OK to generate report. Now you can see the generated report.

Similarly we can generate data export report. Let us click on the create report icon from the toolbar. SoapUI Pro will launch the Create Report dialog window. In the dialog, check all the data sets and change the format to **CSV** so we can verify the report data with Excel. Also check **Generate Header Row** check box. Finally specify the destination folder where the report has to be saved. Look at the following screenshot.



On OK, SoapUI Pro will generate three files (with the name as shown in the data sets section) in the mentioned location in your hard drive. If any errors occurred during execution, error log files also will be created in the same location.

JUnit Style HTML Reports:

JUnit Style HTML Report will generate the test results for each test suite and test cases. To create HTML report, click on the  icon. In the Format drop down, click JUnit-Style HTML report option. Next, click Single Page if it is not selected already. Then specify the destination folder path and click OK. The following HTML format report is obtained.

(Click image for enlarged view)

SoapUI Test Results

Summary

TestCases	Failures	Errors	Success rate	Time
1	0	0	100.00%	41.051

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

Projects

Note: Project statistics are not computed recursively, they only sum up all of its testsuites numbers.

Name	TestCases	Errors	Failures	Time (s)	Time Stamp	Host
GlobalWeather	1	0	0	41.051		

Project GlobalWeather

Name	TestCases	Errors	Failures	Time (s)	Time Stamp	Host
GlobalWeather.WeatherTestSuite	1	0	0	41.051		

[Back to top](#)

TestSuite GlobalWeather.WeatherTestSuite

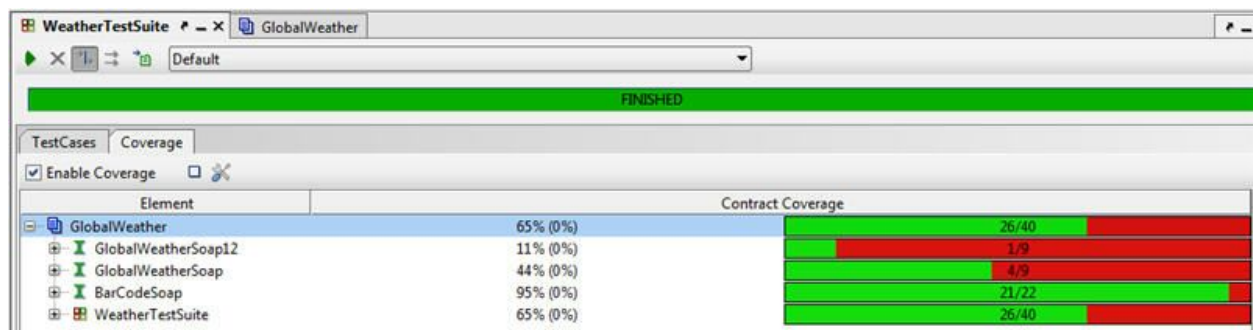
TestCase	Status	Type	Time (s)
TestCase	Success		41.051

Feature #4: Coverage Feature in SoapUI Pro

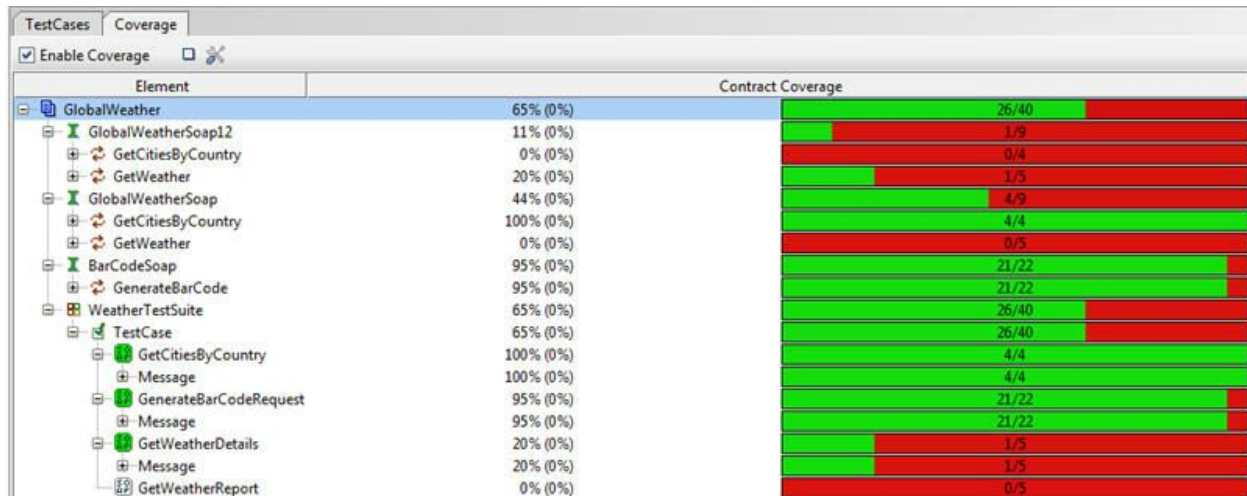
Coverage feature is used to analyse the covered REST or SOAP services. This will be captured during functional testing, mock services testing and so on. Coverage status can be checked for a test suite as follows:

- Double click on the test suite name
- Execute the test suite by clicking on the run icon
- Once execution is completed, we can see the **Coverage** tab present next to the **test cases** tab. Please refer the following screenshot

(Click image for enlarged view)



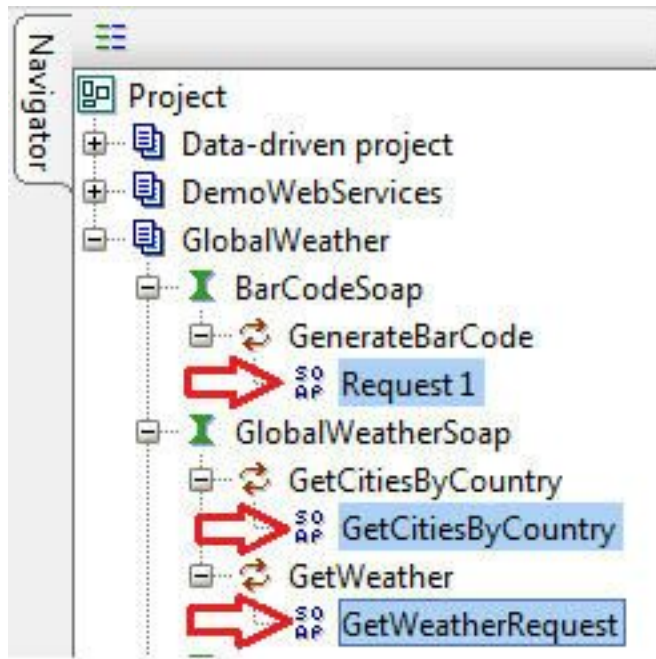
Coverage feature focuses on functional tests, mock services and HTTP monitor scenarios. It also covers project level, test suite and test case levels as can be seen below:



SOAP and REST Services:

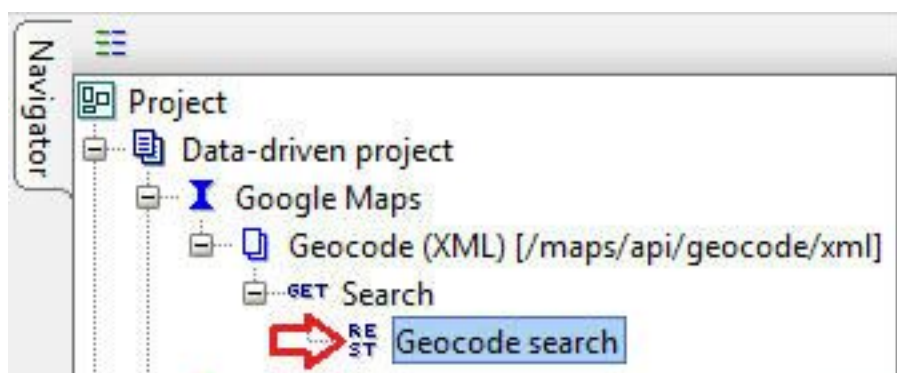
SOAP, created by Microsoft, follows certain standards. It is a protocol that uses XML to transfer the messages across the world through Internet. Its important feature is the built-in error handling- in both request and response. Interestingly, we don't need to use Web Services Description Language (WSDL-a file that is associated with SOAP). WSDL contains the definition of how the web services are working and how we can refer it.

As we all know SOAP based services will be identified by its image indicator in the SoapUI project. See the below screenshot.



Representational State Transfer services (REST) are an alternative to SOAP because of their lighter control. For instance, if we use any script such as JavaScript with SOAP, we will have to prepare XML structure accordingly- which might be harder.

REST does not have complexities as it supports CSV, JSON and RSS format. So we can get the output data for REST services in the above mentioned formats. Please see the screenshot below for REST services in SoapUI Pro.



There are some variances between SOAP and REST web services. Let us use what they are.

SOAP:

- *Heavy weight standard that requires some procedure to access the web services.*
- *Platform, language and transport independent as it does not require HTTP*
- *Widespread acceptance*
- *Error handling integration.*
- *Seamless integration with many languages*

REST:

- *Faster than SOAP*
- *Efficient as it supports various formats like JSON, CSV, RSS*
- *Many open source plugins and tools are available to test REST services easily.*
-

In Conclusion:

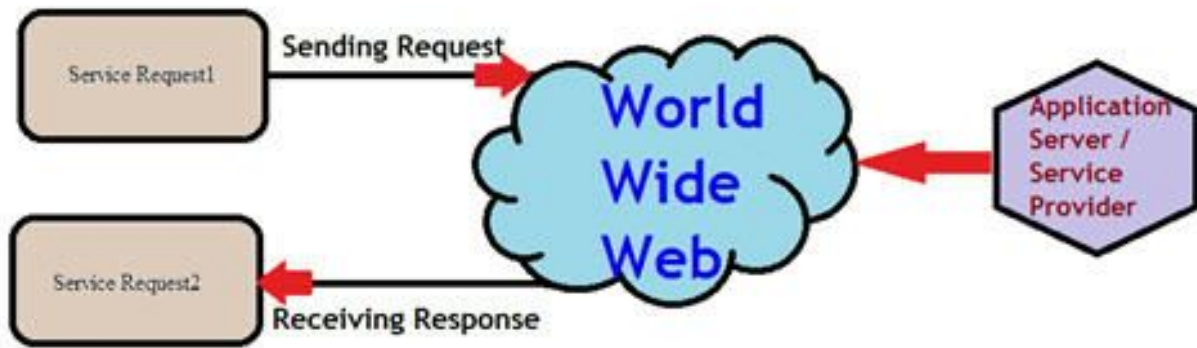
SoapUI Pro as the name indicates is really the Pro version where there are many additional features, which make it easier to use, faster, secure, more capable and versatile. Even though there are many other features that are more specific to pro audience, the ones listed above are the ones applicable to most generic users.

2. *Understanding REST and SOAP Services* (must read)

Understanding REST and SOAP Services:

A web service is a program that helps us to connect two computers over the World Wide Web. Web services are the software component that supports machine to machine interaction over network. This is called interoperability which can be achieved by machine understandable format document called WSDL. WSDL is processed by SOAP and it transfers via HTTP in the form of XML.

Look at this pictorial representation of the web service flow.



What is SOAP Service?

It is basically a protocol which has a set of defined rules to transfer the structured information implemented through web services. SOAP uses XML format data which is platform independent so it can support all the major protocols such as HTTP, FTP, TCP, and UDP and so on.

SOAP services follow the standards for sending and receiving message with the unique format. Usually SOAP message contains the following information:

- *Request / Response Data*
- *action to be performed*
- *Header information*
- *Error details if any failure messages*

In SOAP, security related services given by WS-Security standards are in both the client and server side. WS-Security offers data integrity and privacy. WS-ReliableMessaging is another feature that provides end to end reliable services for success and failure cases.

WSDL is the major technique for handling SOAP service information.

What is REST (Representational State Transfer)?

It is architecture based specially designed for networking applications and is used in client-server systems to send request and response. REST services are also called as RESTful APIs as it is implemented by using Hypertext Transfer Protocol (HTTP). It is GUI independent, and we can test REST APIs using SoapUI without the actual application. It follows a stateless method which means, whenever the client sends the request to the server, server does not store any data in the session.

SOAP vs. REST

- *SOAP is a protocol and REST is architecture. It allows us to send SOAP envelopes to REST based applications.*
- *REST supports different message formats but SOAP permits XML only.*
- *REST services are faster and easy to handle.*
- *SOAP is tied with SMTP and HTTP protocols whereas REST relies on HTTP only.*
- *SOAP is more secure and structured format.*
- *REST does not depend on any specific standards as it supports various messaging formats like JSON, CSV and XML.*
- *SOAP web services allow us to build the client with RESTful services.*
- *SOAP was introduced for distributed computing.*
- *After REST's entry, it accommodated the web by its performance and scalability as it is a light weight component.*
- *REST is stateless whereas SOAP is a state-ful specification.*
- *REST uses Uniform Resource Identifier (URI) and it has the methods like GET, PUT, POST and DELETE to expose their resources.*
- *SOAP uses named operations and interfaces to achieve its business logics.*

Now let us discuss REST services by creating REST project in SoapUI Pro.

Creating REST Project in SoapUI Pro:

Follow the below steps:

1) Open SoapUI Pro application and right click on the Projects node present in the Navigator panel

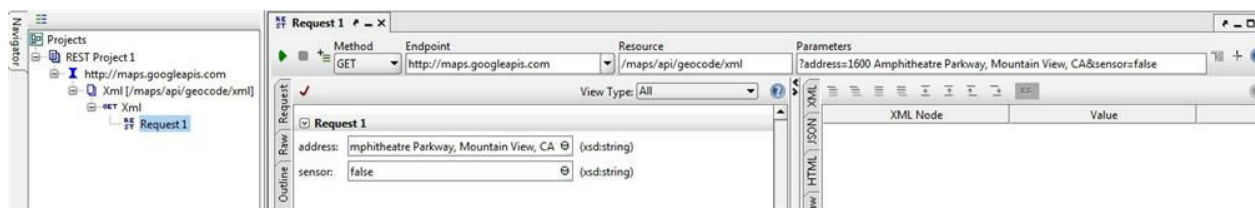
2) In the context menu, click New REST Project option

3) Enter the following Google Map API location in the given text field:


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

4) On OK, SoapUI Pro will create project tree along with resources, service, methods, and endpoint with input request in the editor. See below:

(Click image for enlarged view)



5) As you can see in the above screenshot, there is a parameters section. If you click on it, it will show you the parameters that are used in the service in a separate popup window.

6) Now let us execute this service by clicking on the Run  icon. SoapUI Pro generates the following output for the given endpoint in the form of XML.

(Click image for enlarged view)

Request 1

Method	Endpoint	Resource	Parameters
GET	http://maps.googleapis.com	/maps/api/geocode/xml	?address=1600 Amphitheatre Parkway, Mountain View, CA&sensor=false


Name	Value	Style	Level
address	1600 Amphith...	QUERY	RESOURCE
sensor	false	QUERY	RESOURCE


```

<GeocodeResponse>
  <status>OK</status>
  <result>
    <type>street_address</type>
    <formatted_address>1600 Amphitheatre Parkway, Mountain View, CA 9404
    <address_component>
      <long_name>1600</long_name>
      <short_name>1600</short_name>
      <type>street_number</type>
    </address_component>
    <address_component>
      <long_name>Amphitheatre Parkway</long_name>
      <short_name>Amphitheatre Pkwy</short_name>
      <type>route</type>
    </address_component>
    <address_component>
      <long_name>Mountain View</long_name>
      <short_name>Mountain View</short_name>
      <type>locality</type>
      <type>political</type>
    </address_component>
    <address_component>
      <long_name>Santa Clara County</long_name>
      <short_name>Santa Clara County</short_name>
      <type>administrative_area_level_2</type>
      <type>political</type>
    </address_component>
    <address_component>
      <long_name>California</long_name>
      <short_name>CA</short_name>
    </address_component>
  </result>
</GeocodeResponse>
  
```

We are done with functional testing for Google Map API. Let's add test suites and test cases to get to know more about REST services.

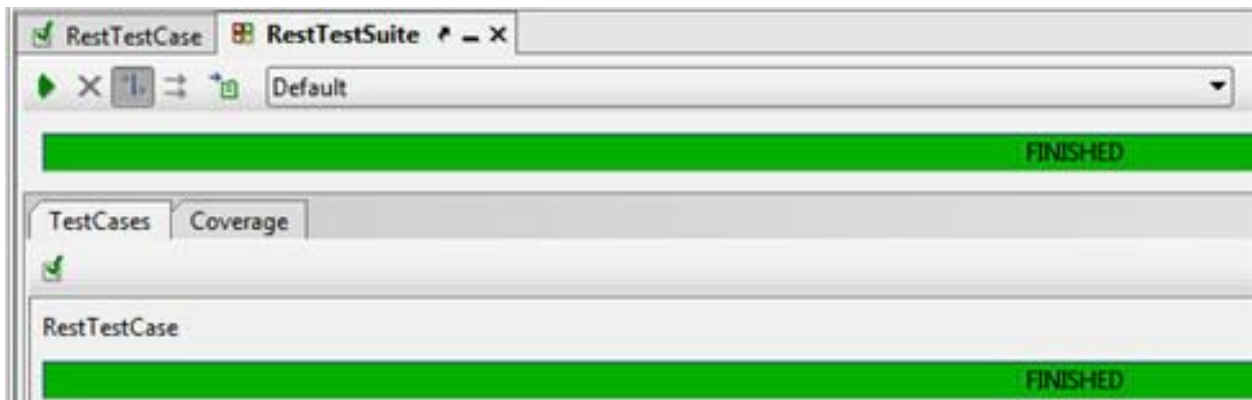
To add test case, do the following:

- 1) Click on the  icon to add test case request
- 2) Enter the test suite name and then click OK
- 3) Then provide the test case name and click OK button
- 4) In the **Add Request to Testcase** dialog, enter the request name and then click OK button
- 5) Now the test suite tree will look like this.



6) Run the test suite by double click on the test suite name

7) Here is the test suite results



8) To get the test results report, click on the  icon from the tool bar.

10) In the Create Report window, make sure the format is selected **TestSuite Report**

11) Or else you can use JUnit-Style HTML Report format

12) Click OK button and verify the results

TestSuite Results Report for RestTestSuite

TestSuite Metrics

Overview

 Project	REST Project 1
 TestSuite name	RestTestSuite
Description	

TestCase Summary

 RestTestCase
--

Base Metrics

 Number of TestCases	1
 Number of TestSteps	1
 Number of Assertions	0
 Number of LoadTests	0


TestSuite Properties

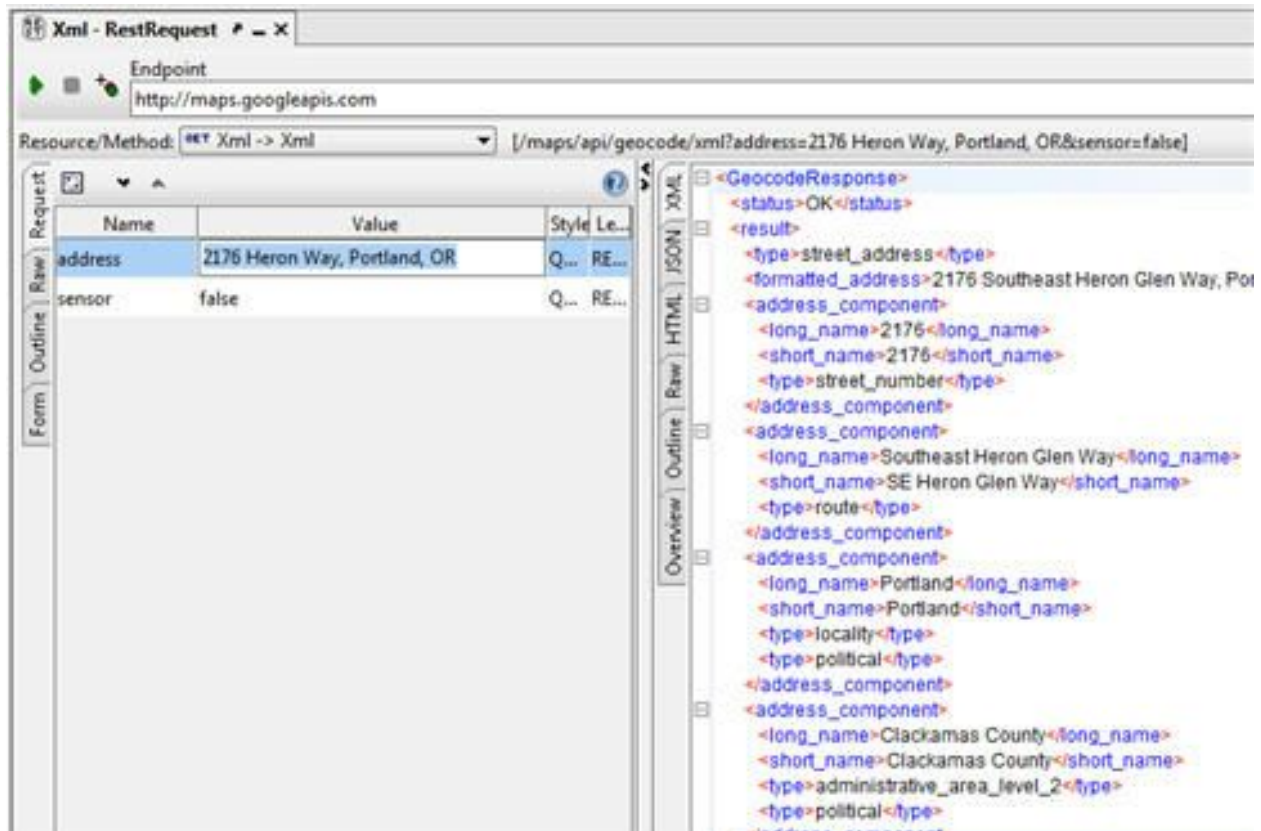
Name	Value
------	-------

Similarly, we can add REST services, resources and methods. As we discussed we can have any number of resources in the resource path.

Let us start with adding REST service:

- Right click interface name which shows as <http://maps. Googleapis.com>
- Then click New Resource option from the context menu
- It opens the **New REST Resource** In that enter the resource path as <http://maps.googleapis.com/maps/api/geocode/xml?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&sensor=false>
- Click OK

- Now the request is added under the project tree. If we wish, we can rename it to be meaningful
- In the **Form** tab, change the address as **2176 Heron Way, Portland, OR**
- Click on the  icon to view the results. Refer the following screenshot.



Difference Between SOAP and REST services:

Now you know how to add REST resources. Here I have summarized the **differences** between **SOAP** and **REST** services.

SOAP

REST

<i>SOAP is abbreviated as Simple Object Access Protocol</i>	<i>REST stands for Representational State Transfer</i>
<i>It is basically XML based message transfer protocol</i>	<i>REST is standard architecture to build web services.</i>
<i>Request and Response data are used in the form of XML</i>	<i>REST service request and response data can be JSON, CSV and XML</i>
<i>It is complicated whenever the WSDL file is changed because we need to re-generate WSDL to build the client accordingly.</i>	<i>We can use REST APIs without disturbing the existing client.</i>
<i>SOAP is tied with HTTP and SMTP protocols</i>	<i>REST relies on only HTTP</i>
<i>Do not have built-in error handler</i>	<i>Supports error handler for identifying the faults during run-time</i>
<i>SOAP messages cannot be cached when it reads</i>	<i>REST data can be cached</i>

Conclusion:

So far in this tutorial, we learned SOAP and REST services and their advantages and differences.

We can also add assertions for the REST services to assert our services. We can add any number of REST test steps and transfer the data between each with the property transfer.

3. Understanding Data Driven Testing

Understanding Data Driven Testing in SoapUI Pro:

In this SoapUI Pro tutorial, we are going to see Data Driven Testing using SoapUI Pro. Performing load testing and performance testing with huge data is often time consuming. This can be overcome through Data driven testing in SoapUI pro.

What is Data Driven Testing?

Reading test data through the test scripts for and iterating execution multiple times is known as data driven testing. Test data is pre-prepared based on the requirements in external sources that could be any of the following:

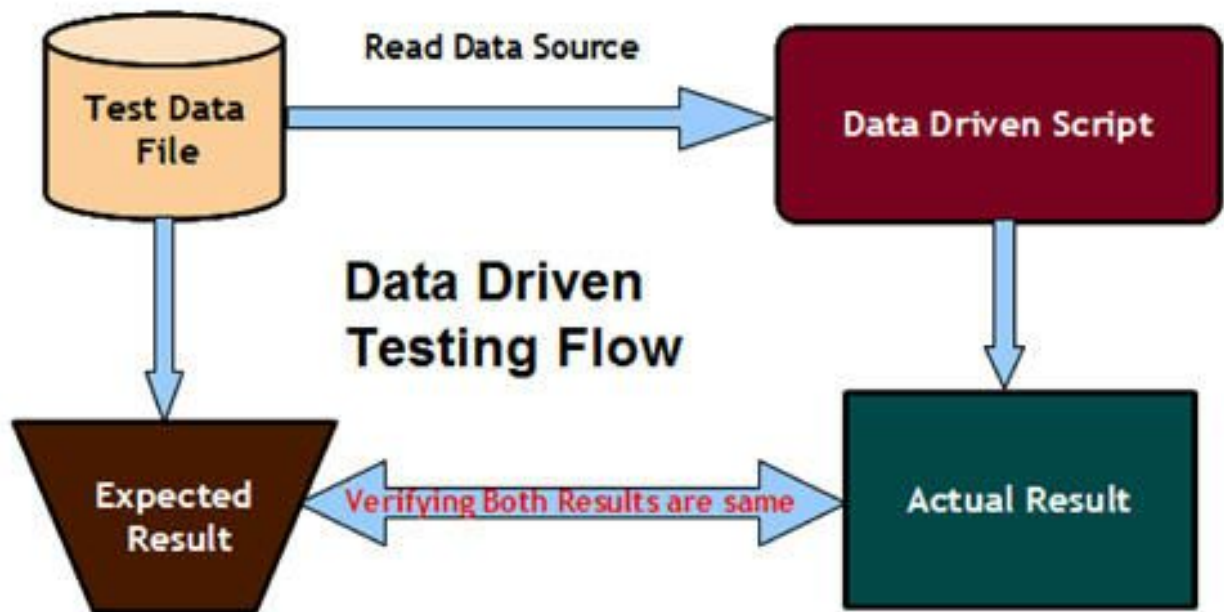
- *Excel Files*
- *CSV Files*
- *ODBC Sources*
- *SQL / ADO Objects*

While running data driven scripts, it will fetch the data from external data source file and then put in to the corresponding variables present in the scripts. For example, let us take login form. This form usually contains user name and password fields. During functionality testing of the login screen, we need to prepare the test data with different combinations of the user name and password and there should be variables to obtain the data in the script correspondingly.

When we call the web service from SoapUI data source test step, it will read first set user name and password. And then it will assign the values to the corresponding variables in the

script. After that, the web service will start the process internally with the username and password.

Take a look at the flow diagram of common data driven testing. This can be implemented via SoapUI pro.



Keyword Driven Testing

Keyword driven testing is a software testing type applicable for both manual and automated testing (most commonly used). It is also called table driven testing. Even though this is quite simple, it does need more time to collect keywords and appropriate functionalities.

In keyword driven framework, we prepare test data like data tables along with the keywords. There are several components available in keyword driven testing framework. They are

- *Control File*
- *Test Cases File*
- *Startup Script*
- *Driver Script*

- *Utility Script*

The "Control File" contains test scenarios to be executed / automated. When testing from the initial stage, user has to select the particular test scenario from the data file. This will be determined based on the flag (Yes / No) present in the data file or excel file.

*"**Test Cases File**" component contains the detailed steps of flow to be executed and this will be prepared in the form excel containing keywords, objects, parameter and check point columns.*

*Next component is "**Startup Script**". This is the first executable script which instantiates the objects and reads the data from the content file. After that it will start executing the test scenarios that are marked as **Yes** in the control file.*

Driver Script

The driver script is responsible for reading test case file and validates the keywords. Then it will call the respective utility script functions based on the keywords available in the test case file. Apart from this, we need to handle the run time errors in the driver script itself.

Utility Script

It consists of relevant logical methods / functions based on the keywords. These scripts will be generic and can be utilized across the applications.

Detailed steps on how to perform data driven testing in SoapUI Pro:

This can be done using Excel, CSV or SQL through JDBC drivers

We will use CurrencyConvertor web service to practice. Before creating new project in SoapUI Pro, prepare test data based on the input request for the web service as shown in the below screenshot.

	A	B
1	FromCurrency	ToCurrency
2	USD	INR
3	INR	USD
4	DZD	USD
5	AUD	USD
6	BSD	USD
7	HNL	USD

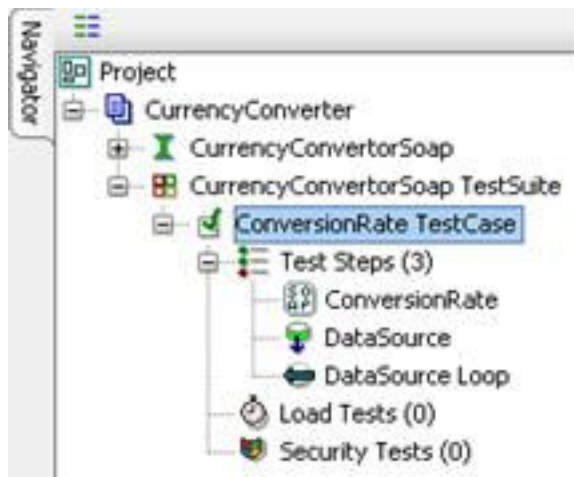
Once test data is ready, open SoapUI Pro and follow these steps.

Step 1: Create a SOAP project using

(<http://www.webservices.com/CurrencyConvertor.asmx?wsdl>)

Step 2: Add test suite and test case steps with the name of "**CurrencyConvertorSoap**TestSuite" and "ConversionRateTestCase" respectively

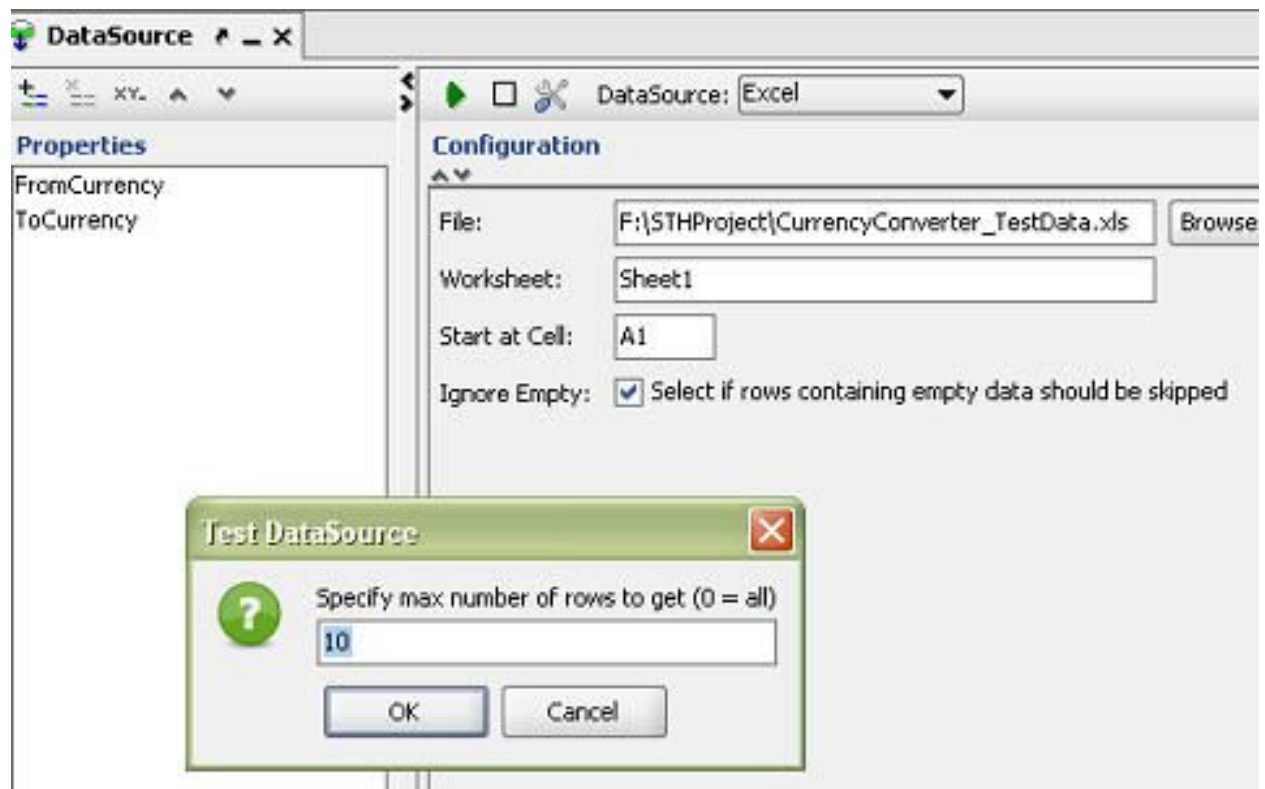
Step 3: Add service request under the test case as below:



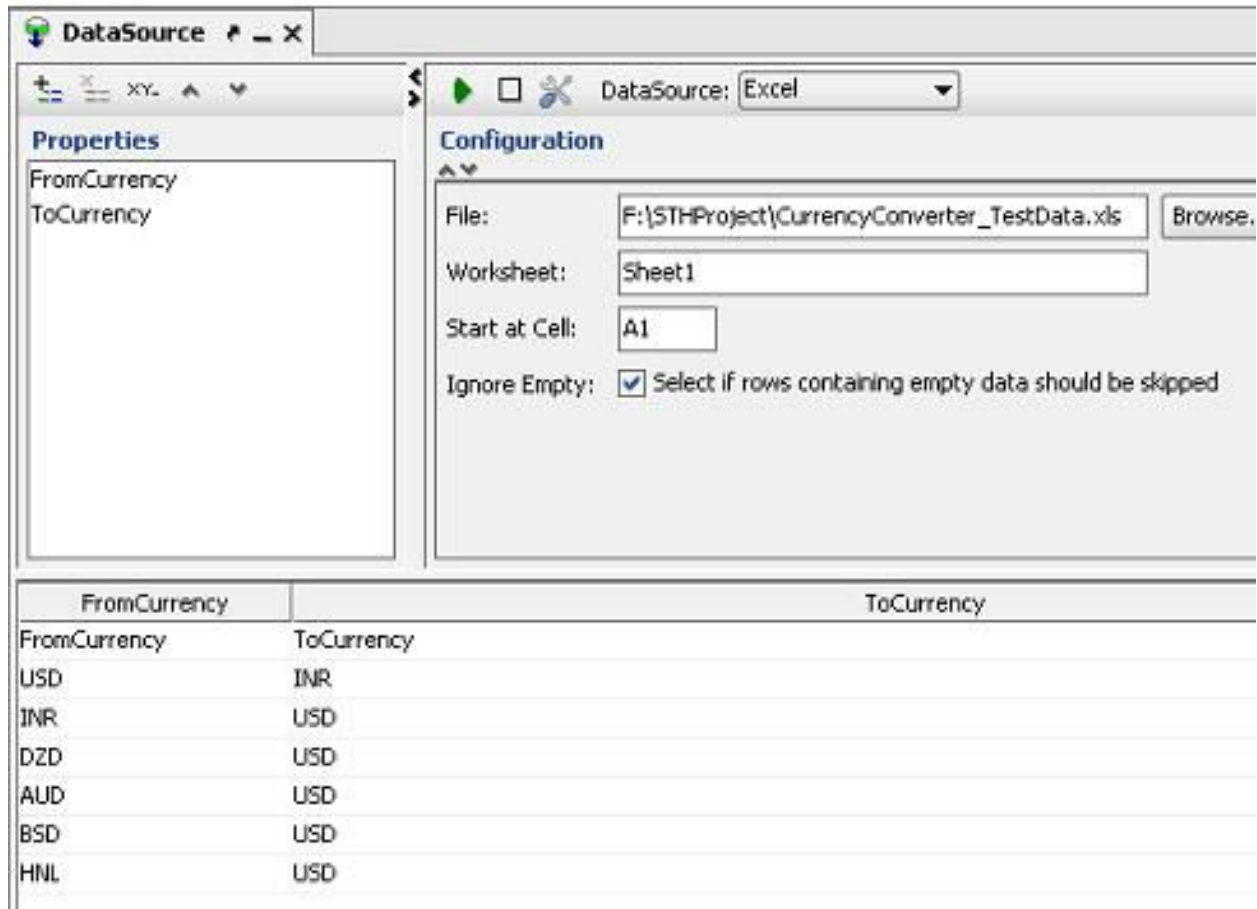
Next we configure the data source test step as described below:

1) Double click on the data source test step present under the project tree

- 2) In the right side screen, select Excel option from the data source drop down*
- 3) SoapUI displays Configuration screen with the **File, Worksheet, Start at Cell** and **Ignore Empty** options.*
- 4) **File** option should be filled with the valid file name selected from local computer. Next enter the exact worksheet name in the **Worksheet** text field.*
- 5) Following that, "Start At Cell" text field has default cell value as "A1". Change if needed.*
- 6) The "Ignore Empty" option helps us to avoid processing blank cells from the selected cell range. If it is checked, SoapUI will not consider the blank cells from the worksheet.*
- 7) Click on the Browse button to select excel file stored in the hard drive*
- 8) Enter the worksheet name as present in the original excel file. Let us specify "Sheet1" as we have entered the test data in the Sheet1.*
- 9) Leave the default Cell name and check the check box. Next we need to add Property names according to the excel headers name. Add property name icon will be present in the toolbar with the (+) symbol.*
- 10) When all the required property names are added, we can execute the data source.*
- 11) Click Run icon to start the execution which loads the test data to the SoapUI grid*
- 12) Now SoapUI Pro will prompt us to specify the number of rows to be fetched from the excel file. If we need all the rows means, we can specify as zero. See the below screenshot for your reference.*



13) Finally click OK to populate the data present in the bottom of the section.



14) To iterate row by row during execution of the test suite, add a data source loop under the test suite where the data source test step is added.

15) For that right click on the test steps node and then click **Add Step -> Data source Loop**

16) Enter data source name in the Add Step popup and then click OK

17) After adding data source loop, we need to configure data source step and target step. For that, right click on the data source loop step and click "Configure" option. Make sure data source step should be "data source" and target step as "conversion rate"

18) Click OK to close the popup

Now it is time to execute the test suite by passing various input data. Double click on the test suite name and then click Run icon. After the execution of the test suite, SoapUI Pro will show us the test results.

Conclusion:

This is a really useful feature to maximize the amount of testing with faster and in an enhanced fashion. However, it is sensitive to changes made to the UI during future releases. It is easy to recover by making appropriate changes to the data tables.

4. **Storing Request and Response in a File** (must read)

Storing Request and Response in a File:

We will start from creating SOAP project:

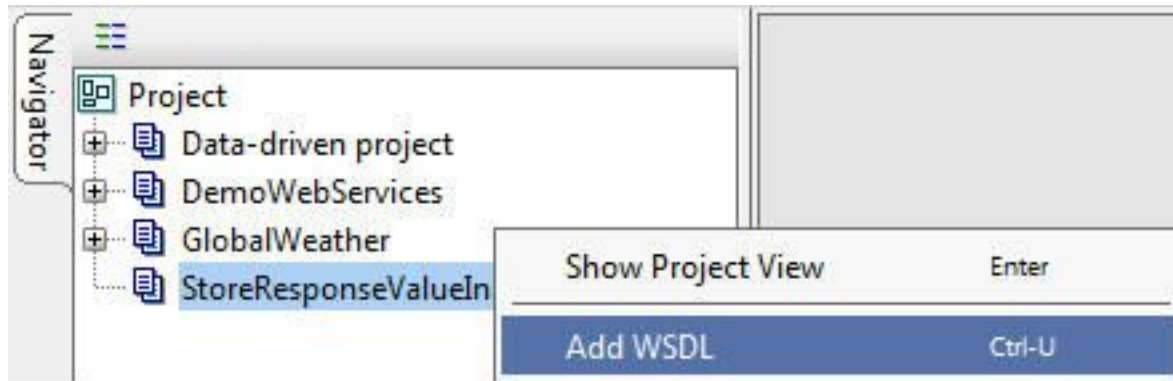
1. Open SoapUI Pro
2. In the Navigator panel, right click on the Project
3. From the context menu, click on New SOAP Project option.(You can also use Ctrl+N)
4. **New SOAP Project** dialog appears on the screen.
5. In the New SOAP Project window, enter the project name,**StoreResponseValueInFile**.
6. click OK
7. Now the project is created successfully without any test steps as we did not include WSDL document

Adding WSDL

Let us add currency convertor WSDL in this section.

1) In the project tree, right click on the project name that was created in the previous section. i.e. *StoreResponseValueInFile*

2) In the popup menu, click **Add WSDL** option. Refer the following screenshot.



3) In the Add WSDL window, enter the currency convertor URL.
<http://www.websvcx.net/CurrencyConvertor.asmx?WSDL>

4) Click OK

5) We can now see the **StoreResponseValueInFile project** under the project tree along with the interface steps. If you expand a web service (i.e. *ConversionRate*), there will be a node known as **Request1** by default.

6) Double click on the **Request1** node to view the request XML content. You could rename it if needed.

Add Test Suites and Test Cases

1) In the project tree, right click on the **CurrencyConvertorSoap** interface

2) From the popup menu, click **Generate Test Suite** option

3) Leave the default settings in and click OK.

- 4) Enter the test suite name in the given text field, **SampleTestSuite** and click OK
- 5) Now test suite is created along with the test case name, **ConversionRate TestCase**.
- 6) Rename the test case name as **SampleTestCase** by right clicking on the test case name and click **Rename** option

Add Groovy steps:

- 1) Under **SampleTestSuite**, right click on the **SampleTestCasenode**
- 2) From the popup menu, click **Add Step -> Groovy Script** test step
- 3) Enter name of the groovy script as **GetResponseValue** and click OK
- 4) Groovy script test step is added under the test suite and redirected to the script editor.
- 5) In the editor, add the following script.

```
def response = context.expand( '${ConversionRate - Request 1#Response}' )
```

```
new File( "D:/Groovy/" + "_response.txt" ).write( response )
```

Note: You should have executed **ConversionRate** request with required input parameters.

- 6) Double click on the **SampleTestSuite -> SampleTestCase -> TestSteps(2) -> ConversionRate - Request1**

- 7) Click on the **XML** tap from the **Vertical** tab bar

- 8) Here, replace the following input data instead of question mark (?) in the soap code.

- FromCurrency = USD
- ToCurrency = INR

9) Double click on **SampleTestSuite** -> **SampleTestCase** -> **TestSteps(2)** -> **GetResponseValue**

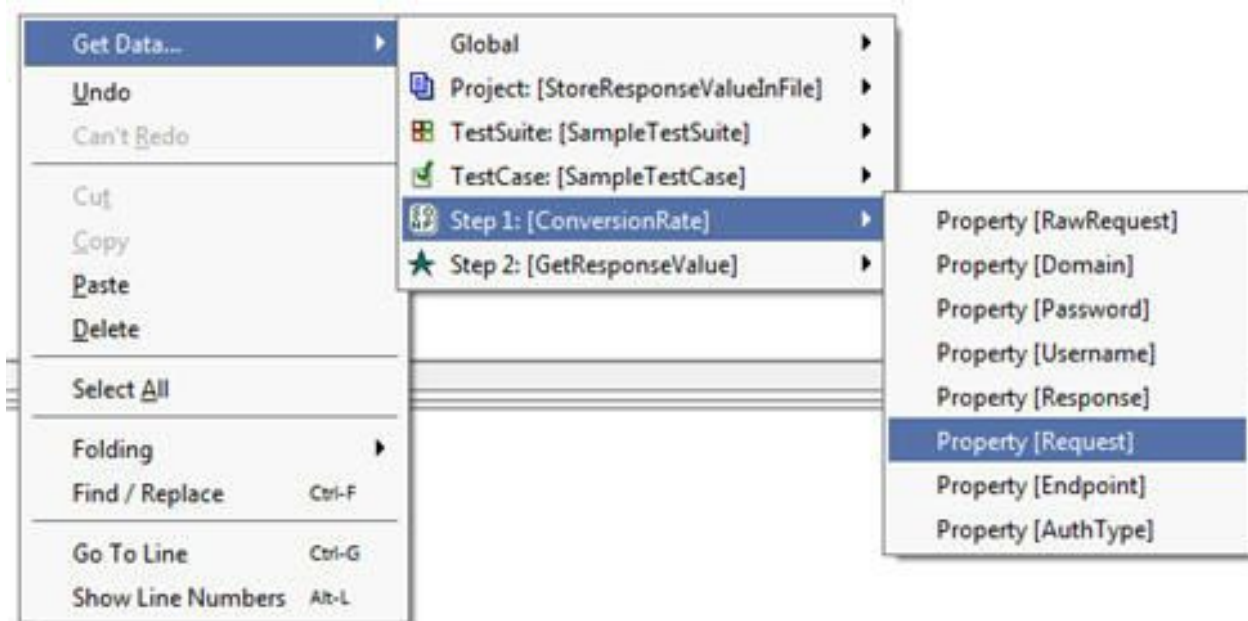
10) Now **Run** the TestSuite by clicking on this icon.

Now we are done storing response data in our local drive.

Verify the response in the response tab and also in the hard drive to ensure both responses are identical.

We can also save the request information in a file. For that, do the following.

1) Right click on the script editor and then click **Get Data** -> **Step 1 [ConversionRate]** -> **Property [Request]** option. Refer the following screenshot.



2) SoapUI Pro will give you auto generated code like this.

```
def request = context.expand( '${ConversionRate#Request}' )
```

3) Then add the following script in the next line

```
new File( "D:/groovy/" + "_request.txt" ).write( request )
```

*If we execute this script, we will get a file with the name of **_request.txt** in our local drive as mentioned in the script.*

Here are some tips about storing raw request through groovy scripting. Take a look at the following sample scripts.

```
testCase.getTestStepByName(<teststep name>).getProperty("Request").getValue()
```

Or

```
context.testCase.getTestStepAt(<index>).getProperty("Request").getValue()
```

Or

```
testRunner.testCase.getTestStepAt(<index>).getProperty("Request").getValue()
```

The above scripts are used to get the raw web service request while executing the test step. Following the script, we can write another line to receive request information in the variable as we have seen in the above samples. See the below sample code to understand this better.

```
def request =  
testCase.getTestStepByName("ConversionRate").getProperty("Request").getValue()
```

```
new File( "D:/GroovyRequest/" + "RawRequestData.txt" ).write( request )
```

Conclusion:

That brings us to an end, not only of this article, but the entire series. We hope this has been useful to you and brought you closer to your SoapUI learning skills. As always,

practice, patience and persistence are the most as important as the tutorials themselves for best results.