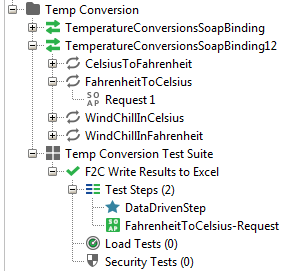
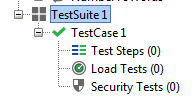
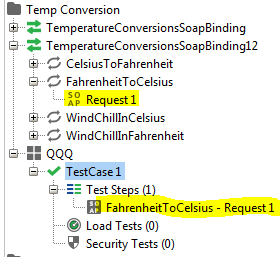
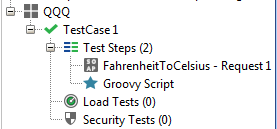
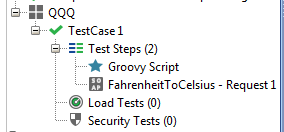
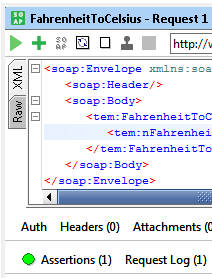
SoapUI Data-Driven Test Automation Framework Assembly Instructions

1. Download and install the Open Source (free) version of SoapUI
2. Install the JXL.jar file (used to read and write to Excel .xls files)
   1. Goto <https://www.softpedia.com/get/Programming/Components-Libraries/JExcelApi.shtml>
   2. Download **jexcelapi**, unzip it and copy only the **jxl.jar** file into SoapUI’s **\bin\ext** folder. By default, it is located in **C:\Program Files (x86)\SmartBear\SoapUI-5.4.0\bin\ext**
   3. If SoapUI was up while you copied jxl.jar into SoapUI’s **\bin\ext** folder, then close SoapUI and restart it.
3. In SoapUI, create a New SOAP Project – File > New SOAP Project or click the SOAP icon
   1. Enter a project name.
   2. Don't fill in the "Initial WSDL" text box yet – you could, but we will do it another way.
4. In SoapUI's left-hand pane, right-click on your project and pick "Add WSDL"
   1. For the temperature conversion webservice, enter <http://webservices.daehosting.com/services/TemperatureConversions.wso?WSDL>
      1. If problems, check <http://webservices.daehosting.com/services/TemperatureConversions.wso>
      2. Click on "please review the [Service Description](http://webservices.daehosting.com/services/TemperatureConversions.wso?WSDL)."
   2. For the number-to-words webservice, enter <http://www.dataaccess.com/webservicesserver/NumberConversion.wso?WSDL>
      1. If problems, check <http://www.dataaccess.com/webservicesserver/>
      2. Click on [Number Conversion Service](http://www.dataaccess.com/webservicesserver/NumberConversion.wso)
      3. Click on "...please review the [Service Description](http://www.dataaccess.com/webservicesserver/NumberConversion.wso?WSDL)"
   3. Keep the defaults and press OK
   4. You should see the Bindings appear. There will be 2 – one for SOAP 1.1 and "12" is for Soap 1.2
   5. 
5. Right-click on the project and pick New TestSuite
   1. Give the TestSuite a name – this won't use Test Suites but they need to be present
6. A new window appears in SoapUI with a "Test Cases" tab – click the green checkmark.
   1. This creates a Test Case entry. Give it a name or use the default name.
   2. If the window doesn't appear or you have closed it, right-click on the TestSuite name and click New TestCase.
7. Expand the treeview (left pane) until you see Test Steps. It should look like this: 
8. Pick the Soap envelope request you want to use (under Bindings) and drag it down under Test Steps. It should look like this:  It will ask you if you want to move it (copy it really) – say YES. Another window will pop up - take the defaults.
9. Rename the Request Test Step to "**FahrenheitToCelsius-Request**" This must be done or you must modify the 1 line in the Groovy Script so that it **matches the Test Step Request name**.
10. Add the Groovy Test Step by right-clicking on the Test Step node and picking Add Step > Groovy Script. Give it a name if you wish – it can always be renamed later. It now looks like: 
11. Pull the Request below the Groovy Script. Not sure if this is really needed. It now looks like this: 
12. Copy and paste the Groovy Script code (DataDrivenGroovyCode in GitHub) in here.
13. Double-click on the Request envelope.
    1. You can test the webservice to see if it's working by replacing the ? with a Fahrenheit value and pressing the green "Play" button. It should return with a Celsius value.
    2. Replace the "?" in line <tem:nFahrenheit>?</tem:nFahrenheit>with ${#TestCase#Fahrenheit}. So it looks like <tem:nFahrenheit>${#TestCase#Fahrenheit}</tem:nFahrenheit> See SoapRequestEnvelope in GitHub.
    3. Add an assertion – see ContainsAssertion and/or XPathMatchAssertion in GitHub. Both can be added.
       1. Click on green Assertions button. 
       2. Click on the + sign to add an assertion. Pick Property Content>Contains or Property Content>XPath Match and copy the code from GitHub in there.
14. Create or copy the Excel sheet used for testing (Fahrenheit-Temps.xls in GitHub). Save this to an area that can be read and written to (not always allowed in the root of C:).
15. Update the Groovy Code with the location of the Excel file.
16. Save All in SoapUI. It will automatically save it when you exit SoapUI, unless you use the File menu exit option and choose "Exit without Saving"
17. Run it – double click on the Test Case or Test Step node (both bring up the Test Case run window.
    1. Press the green play button and you should see the test cases pass by at the bottom of the window. Green=PASS, Red=Fail
    2. It will read the Excel sheet until it encounters an empty row, at which point it ends.
    3. If it stops after the first failure, click on the gear icon at the top of the test run window and uncheck "Abort on error" then re-run your tests by clicking the green play button again.

That should be it.

See the document "SoapUI Gotchas" which details some of the issues I have run into using SoapUI 4.5.0 Open Source version.