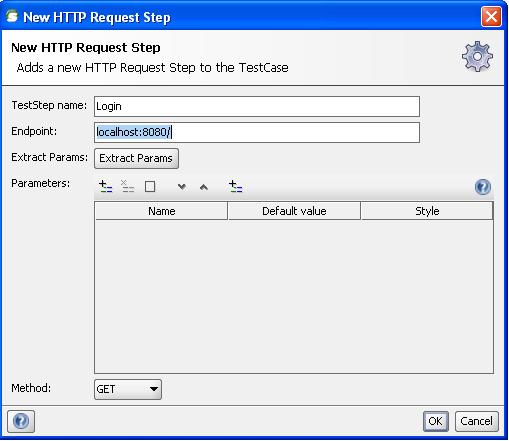
Soap UI API (HTTP Test request) testing for beginner

**About Soap UI:**  
  
Soap UI is a free and open source cross-platform Functional Testing solution. With an easy-to-use graphical interface, and enterprise-class features, Soap UI allows you to easily and rapidly create and execute automated functional, regression, compliance, and load tests. In a single test environment, Soap UI provides complete test coverage and supports all the standard protocols and technologies. There are simply no limits to what you can do with your tests.  
  
**Soap UI testing area:**  
  
SOAP UI is basically used for API testing, below are some areas of  SOAP UI testing.

1. Functional Testing.
2. Load testing.
3. Service mocking
4. Security Testing.
5. Http protocol request testing.
6. SOAP and WSDL
7. Rest API testing.
8. JMS API testing.
9. AMF type API testing.
10. WEB and HTTP.

**Procedure- test HTTP type API method.**  
  
Download and install Soap UI from url “[sourceforge.net/projects/soapui/files/](http://sourceforge.net/projects/soapui/files/)”.  
Go to start >All Programs > SmartBear >soap UI \*.\*.\*   and click on Soap UI-\*.\*.\*  
You should see Soap UI will open on machine.  
  
**Procedure Create First Test:**

1. Click on File > New Soap UI project, enter project name in "Project Name" filed and click on OK button.
2. Right click on created project and click on “New TestSuite” option. You should see a popup window. Enter suite name in "TestSuite" filed. Click on OK button.
3. Expand created project, right click on created suite and click on “New TestCase” option. You should see a popup window. Enter test case name in “TestCase” input filed and click on "OK" button.
4. Expand created suite and  test case. Right click on “Test Step” select add step and click on “HTTP Test Request” enter "HTTP Test Request" name in input field. Click on "OK" button as I entered login and after clicking on "OK" button screen should be as below.

[](http://4.bp.blogspot.com/-AwAmWrqZo3I/UU2O1VL3_gI/AAAAAAAAAGY/t3FmMFtt1jc/s1600/soapUI.teststep.jpg)

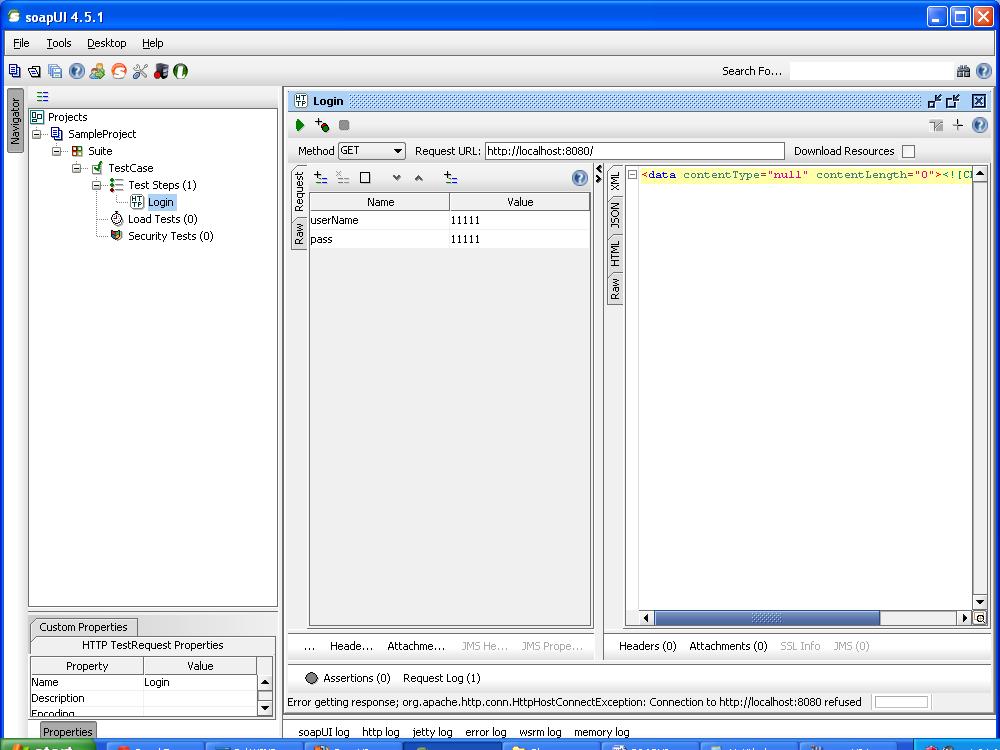
   5.  Enter end point url, end point” url is HTTP url, it may be locally or globally.

   6.  Select method type from "Method" drop down option as I selected “GET” method.

   7.  Click on + icon in parameters filed and add parameters if any as I added username and password field as parameter.

   8.  Click on "OK" button.

   9.  After successfully created test step, click on play icon your response of request is displayed as right side screen.

[](http://3.bp.blogspot.com/-YoNfBMtkTsQ/UU2PfxQwxzI/AAAAAAAAAGg/vaidlBt21bg/s1600/soapuitest1.jpg)

You can verify your responses manually. You can use assertion and verification to check automatics response at execution time.

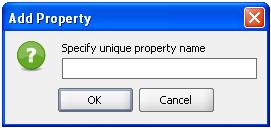
Parameterization in soap ui using soap ui property feature

Sometimes we need to use some values in several times in our api test, such as api end point, api methods name, user name, password etc. If these values we put in place and use these values when needed in our test. If these values changed then we need to update only a single place.  
  
In SoapUI we can use parametrization from external files of property feature of SoapUI.

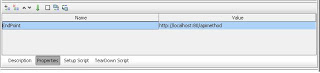
**Parametrization using SoapUI properties:**  
  
 In soapui properties can be created at three level

1. Test Case
2. Test Suite
3. Project.

Click on created test case. Click on properties tab at bottom of soapUI.  
Click on + icon and enter property name.

[](http://3.bp.blogspot.com/-e7ono56R1UM/UVCtMuAUaTI/AAAAAAAAAIg/01I5AKk5LQI/s1600/soaUIPro.jpg)

As I created a property for endpoint url .

[](http://2.bp.blogspot.com/-TNhgY7RNiNo/UVCsO9EyVVI/AAAAAAAAAIU/DRvRgkY_dMc/s1600/soapproperty.jpg)

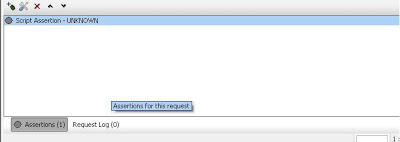
Similarly we can create properties for Test Suite and Project level..  
  
**How to use:**

1. ${#TestCase#EndPoint} for test case property.
2. ${#TestSuite#EndPoint} for test suite property.
3. ${#Project#EndPoint} for project property.

[http://3.bp.blogspot.com/-FmKJx-TcZjQ/UVCsQG8GuPI/AAAAAAAAAIc/l-vnThF5pXk/s320/Endpoint.jpg](http://3.bp.blogspot.com/-FmKJx-TcZjQ/UVCsQG8GuPI/AAAAAAAAAIc/l-vnThF5pXk/s1600/Endpoint.jpg)

### Soap UI Assertions

In this post I am going to explain what assertion in soapui is and how we can apply script assertion.  
  
**Assertion**  
  
Assertion functionality in SoapUI is used to validate the response of request received by the Test Steps at the time of execution. Usually assertion is to compare a part of message (or the entire message) to some expected value.  
Assertion applied in SoapUI at response of request, if any assertion failed then test marked as failed. Failed test can be verify and find out the reason.  
  
  
**Assertion types in SoapUI**  
  
1.    Contains  
2.    Not Contains  
3.    XPath Match  
4.    Xquery Match  
5.    Scripts  
6.    SLA  
7.    JMS  
8.    JDBC  
9.    Security  
  
**Script Assertion**  
In SoapUI Groovy script is used for scripts assertion. Click on assertion option from your created test step.

[](http://4.bp.blogspot.com/-nL3jnDDvUX0/UVc8C2kooFI/AAAAAAAAAJs/e-Nqslbq15k/s1600/SoapUi-Assertion.jpg)

Click on + icon, select scripts option from popup window and click Add button. Enter assertion name and click on OK button.  
Suppose you are getting response in xml format as my test step response is in xml format.

 <responce>

    <status>OK</status>

    <data>2</data>

 </responce>

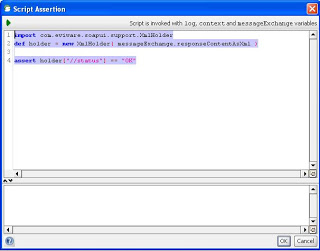
For above response I used below groovy scrip assertion for status node value

 import com.eviware.soapui.support.XmlHolder

 def holder = new XmlHolder( messageExchange.responseContentAsXml )

 assert holder["//status"] == "OK"

 assert holder["//data"] == "2"

[](http://4.bp.blogspot.com/-FAccOUXAR6s/UVc8CyIZ9XI/AAAAAAAAAJo/GxfRsHzDEWU/s1600/SoapAssertion2.jpg)

### Some Important SoapUI functions in groovy script.

In this post I will show you some important soapui functions which are basically used in soap ui groovy scripts.  
  
1. Following functions are used to read properties in soapui project.

projectPropertyValue = context.expand( '${#Project#Test}')

projectPropertyValue = context.expand( '${#TestCase#Test}')

projectPropertyValue = context.expand( '${#TestSuite#Test}'

I have create three properties with name "Test" at project, testsuite and test case level. Using above code we can read values in groovy scripts and can be use where needed.  
2. Setting properties value test case.

def myTestCase = context.testCase

myTestCase.setPropertyValue("Name", “Value”)

3. Reading end url value of any test steps. Replace your test step name with “TestStepName”

EndPointUrl  = context.getProperty("TestStepName","Endpoint")

4. Setting parameter value of test steps. Replace you test step name, parameter name and value with “TestStepName”, “dataSet” and “value”.

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

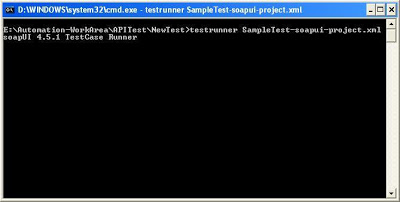
//set data parameter value in TestStep

groovyUtils.setPropertyValue("TestStepName", "dataSet", “value”)

Will continue adding more function

### Road to command line execution of soapUI project

We run SoapUI project using SoapUI GUI interface.  SoapUI also provide set off batch files to execute soapUI tests from command line without using SoapUI GUI interface.   
  
Following are the runner scripts inside installed SoapUI bin folder.  
1. **testrunner.bat :** this is use to run functional test using command line.  
2. **loadtestrunner.bat:** This is used to run soapui load testing using command line.  
3.**mockservicerunner.bat**:  This is used to run soapui mock services using command line.  
4. **toolrunner.bat:**This is used to launch soapui tools.  
5. **securitytestrunner.bat:** This command is used to run security tests  
  
**Note:**setup soaui bin folder path into your system variable path.  
  
**Command Line soapui tests execution:**  
1. Open command prompt and got to your created soapui project.  
2. Run command > testrunner {SoapProjectName}.xml

[](http://3.bp.blogspot.com/-2CcOSNFLxjM/Uf04ZNcsCaI/AAAAAAAAAYc/jVyuxIMlLjI/s1600/testrunner.jpg)

3. After running test you should see some text file is generated in root.

4. We can run test suite or test case using testrunner command. as below:

testrunner –c "APITestCase" -r SampleTest-soapui-project.xml

Where “APITestCase is test name and “SampleTest-soapui-project.xml” is soapui project. Here I use –r flag so that after execution console summary look like below.

SoapUI 4.5.1 TestCaseRunner Summary

-----------------------------

Time Taken: 24493ms

Total TestSuites: 0

Total TestCases: 1 (0 failed)

Total TestSteps: 2

Total Request Assertions: 0

Total Failed Assertions: 0

Total Exported Results: 0

5. Following are the testrunner flags and its description.

usage: testrunner [options]

 -v    Sets password for soapui-settings.xml file

 -t    Sets the soapui-settings.xml file to use

 -A    Turns on exporting of all results using folders instead of lon

       filenames

 -D    Sets system property with name=value

 -G    Sets global property with name=value

 -I    Do not stop if error occurs, ignore them

 -M    Creates a Test Run Log Report in XML format

 -P    Sets or overrides project property with name=value

 -S    Saves the project after running the tests

 -a    Turns on exporting of all results

 -c    Sets the testcase

 -d    Sets the domain

 -e    Sets the endpoint

 -f    Sets the output folder to export results to

 -h    Sets the host

 -i    Enables Swing UI for scripts

 -j    Sets the output to include JUnit XML reports

 -m    Sets the maximum number of TestStep errors to save for each  testcase

 -p    Sets the password

 -r    Prints a small summary report

 -s    Sets the testsuite

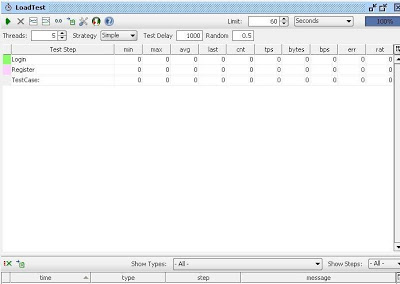
 -u    Sets the username

 -w    Sets the WSS password type, either 'Text' or 'Digest'

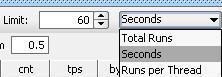
 -x    Sets project password for decryption if project is encrypted

### Road to performance testing using soapUI

In this post I would like to show you performance testing of webservices using soap ui.  
Performance testing of web services is not just running SOAP or XML messages in a loop to run the service. It should be a well-planned activity which must be aligned with the performance expectations of the overall service-oriented solution.  You can run multiple type of performance test such as SOAP , JSON, XML format messaging through a single interface  
SoapUI allows users to configure various load testing options such as delays in between threads to simulate real-world use cases and run tests in burst mode to stress test services.  
  
**How to create Load Test:** Just go to Navigation bar of soapUI click on your create soaui project and expand test case. Right click on “load test” options, enter load test name on opened window and click on OK button. Your created load test window looks like as below.

[](http://3.bp.blogspot.com/-QSvTrMy3DnY/UfPkydEgP4I/AAAAAAAAAW8/jifQHlOVH58/s1600/LoadTest.jpg)

As in above window I have two soap ui test method “login” and “Register” both methods are automatically reflect on load test window.  
  
**Load Test Limit:** load test limit define interval of test case execution. SoapUI has two variable for limit 1. Limit value 2. Limit type. Limit type has three options “Total Runs”, “seconds” “Runs per thread”  
1. Total Runs limit type is used to set the number of times the TestCase needs to be executed during each load test run.  
2. Seconds, lumit type is used to be execute of load test for a particular time.  
3 .Runs per thread limit type is used to set number of time to be execute per thread of TestCase.

[](http://4.bp.blogspot.com/-GUwYaB_i-u8/UfPlj5-7W1I/AAAAAAAAAXE/HL_zhPshZus/s1600/LimitSoauI.JPG)

**Threads:**Threads act as virtual users in a load test. If the thread count is set as N, soapUI creates N number of clones of the associated TestCase and executes them.

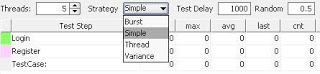
**Load test strategies:** following type load strategies are used in soau UI

1. Simple

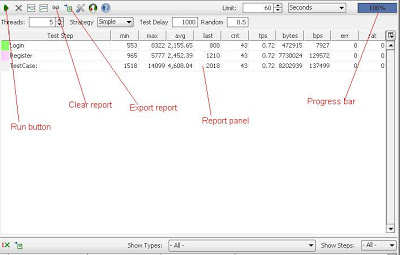
2. Burst

3. Thread

4. Variance

[](http://3.bp.blogspot.com/-rdzb7A5Gi_A/UfPlkwmIWnI/AAAAAAAAAXM/56Mtr3idUCI/s1600/Strategy.jpg)

Set any load strategy as your requirement and click on run icon. After execution your window looks like as below screen.

[](http://2.bp.blogspot.com/-dfntrbBc8Bk/UfPrTjKr8bI/AAAAAAAAAXc/bbQjjMM0AXA/s1600/Soapexecution.JPG)

### Execution of SoapUI project using Junit

In this post I am going to show you how to execute SoapUI project using Junit.  
SoapUI provide testrunner class name “SoapUITestCaseRunner” which can be used to run soapui test using java class, maven or ant build tool. We create object of this class in java file and call the run function to execute soapui project.  
  
**Steps to create Junit test for soap ui project:**  
1. Chose any IDE as eclipse or NetBean  
2. Add “soapui-4.5.1.jar” from soapui bin folder and soapui lid folder as library to class path of your project (eclipse).  
3. Use “SoapUITestCaseRunner” class object to run soapui project as mentioned in below code:

//import classes

import org.junit.Test;

import com.eviware.soapui.tools.SoapUITestCaseRunner;

//Junit test method

@Test

public void testSoapUI() {

    //initialize  SoapUITestCaseRunner class object

    SoapUITestCaseRunner soapUITestCaseRunner = new SoapUITestCaseRunner();

    //set your project absolute path of your project

    soapUITestCaseRunner.setProjectFile("D://SampleTest-soapui-project.xml");

    try {

            //call run method to execute all test of project.

              soapUITestCaseRunner.run();

        } catch (Exception e) {

              e.printStackTrace();

    }

}

4. Use below code if you want to execute test case not just executing all test case

//import classes

import java.io.IOException;

import java.util.List;

import org.apache.xmlbeans.XmlException;

import org.junit.Assert;

import org.junit.Test;

import com.eviware.soapui.impl.wsdl.WsdlProject;

import com.eviware.soapui.model.support.PropertiesMap;

import com.eviware.soapui.model.testsuite.TestCase;

import com.eviware.soapui.model.testsuite.TestRunner;

import com.eviware.soapui.model.testsuite.TestSuite;

import com.eviware.soapui.support.SoapUIException;

//Junit test method

@Test

public void testSoapUI() throws XmlException, IOException, SoapUIException {

    // Create a  WsdlProject instance by specifying the project absolute path

    WsdlProject project = new WsdlProject("D://SampleTest-soapui-project.xml");

   // retrieve all test suite from project

    List testSuiteList = project.getTestSuiteList();

    // Iterate all TestSuites of project

    for (TestSuite ts : testSuiteList) {

         System.out.println("\*\*\*\*Running Test suite " + ts.getName() + "\*\*\*\*\*\*\*\*");

         // Retrieve all TestCases from a particular TestSuite

         List testCaseList = ts.getTestCaseList();

          // Iterate all TestCases of the particular TestSuite

           for (TestCase testcase : testCaseList) {

                 System.out.println("\*\*\*\*Running Test Case " + testcase.getName()+ "\*\*\*\*\*");

                  // Run the specific TestCase

                  TestRunner testRunner = testcase.run(new PropertiesMap(), false);

                  //verify where test case pass or not

                 Assert.assertEquals(TestRunner.Status.FINISHED, testRunner.getStatus());

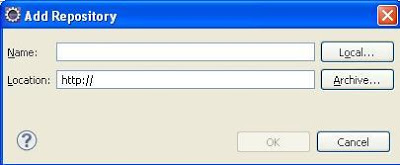
            }

         }

    }

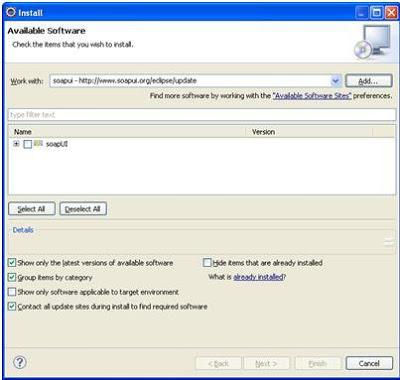
### Road to installation of SoapUI plugin into Eclipse.

Following are the steps to install soapui plug-in into eclipse.  
1. Open Eclipse IDE and goto “Help” menu and click on “Install New Software”  
2. Click on add button you should see like below window.

[](http://3.bp.blogspot.com/-OR-cSjL2J3w/Uf113lrTySI/AAAAAAAAAYs/lWuPhBFKvdM/s1600/SoaupEclipse1.jpg)

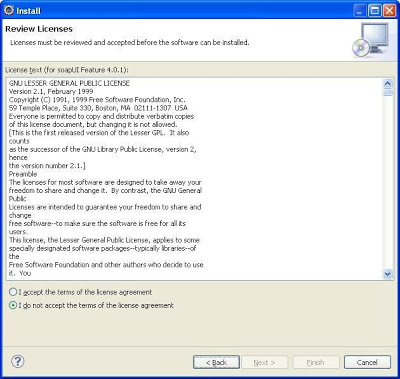
3. Enter some text into “name” field as “SoapUI“ and enter url http://www.soapui.org/eclipse/update into “Location” field.

4. Click on “OK” button, you should see below screen and soapUI will display.

[](http://1.bp.blogspot.com/-9KxzHf9osW0/Uf114HE-UVI/AAAAAAAAAZA/aHwlfTzAAkA/s1600/SoapUIEclipse.jpg)

5. Click on soapui check box and click next button.

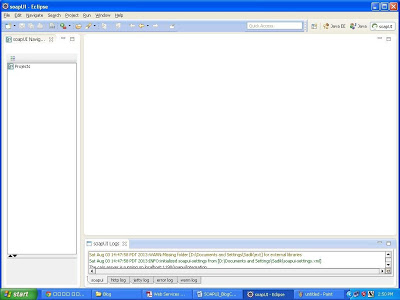
6. Click on Next button again you should see license agreement window.

[](http://3.bp.blogspot.com/-CMGbqbew0fE/Uf1133i1SvI/AAAAAAAAAZI/1IAcGVRQ7v8/s1600/SoapEclipse2.jpg)

7. Accept term and license agreement and click on next button. you should see installation start.

[](http://2.bp.blogspot.com/-AYCBDfgCNK4/Uf114WkKvlI/AAAAAAAAAZE/2qivdsMvnAs/s1600/SuapUIEclipse4.jpg)

8. Accept term and license agreement and click on next button.  
9. During installation security warning popup window displayed, click on “OK” button of security warning.  
10. Restart Eclipse to take effect on the new plugin installation  
11. In eclipse goto “window>>Open Perspective>>Other” on top menu bar and select soapUI. You should see a familiar soapUI project explorer

[](http://4.bp.blogspot.com/-g-eAMMq36E8/Uf17jdK_1FI/AAAAAAAAAZU/W18ub-h_RDg/s1600/soauEclipse5.jpg)

### Road to reading data from text file in SoapUI using groovy script

In this post, I am going to show you how to read data from text file in SoapUI. SoapUI Pro has some advance feature which is not in SaopUI as data fetching from external sources so in SoapUI we use Groovy script for that. Following are the peace of groovy script code for reading data from text file.  
  
1. Reading all data from text file.

//reading all txt file at once

File file = new File("D://user.txt")

fileContent = file.getText()

log.info fileContent

2. Reading data line by line from text file.

//reading text line by line

File file1 = new File("D://user.txt")

List textLine = file1.readLines()

log.info textLine

3. Reading data randomly of any line from text file.

//reading text randon line number

File file2 = new File("D://user.txt")

List textLine2 = file2.readLines()

rowIndex  =  Math.abs(new Random().nextInt() % 4 + 1)

log.info textLine2[rowIndex]

### Road to data driven testing in SoapUI from csv file

In this post I will show you how to execute SoapUI test for a set of values. For this I have put all data in csv file. I write groovy scripts for reading data from csv and executing the test steps.  
Below is the groovy script for reading data from “URL.csv” file where I put all user credentials.

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

def csvFilePath = "D:\\URL.csv"

context.fileReader = new BufferedReader(new FileReader(csvFilePath))

rowsData = context.fileReader.readLines()

int rowsize = rowsData.size()

for(int i =0;  I < rowsize;  i++)

{

    rowdata = rowsData[i]

    String[] data = rowdata.split(",")

    log.info data[1]

    groovyUtils.setPropertyValue("Login", "UserName", data[0])

    groovyUtils.setPropertyValue("Login", "Pass", data[1])

    testRunner.runTestStepByName( "Login")

}

In above code I read data from “URL.csv” file and pass user name and password into “UserName “ and “Pass” of Login steps parameters.

“Login”  test step executed by using test runner for every data.

Road to data driven testing in SoapUI using groovy script with excel file

SoapUI Pro has a feature to read data from external files like: excel, csv etc. But SoapUI does not provide such feature to read data from excel file. So for reading data from excel file in SoapUI, we need to write some code in groovy script.  
I this post I will show you, how to read data from excel file.I am using poi jar files to read data from excel file in groovy, download following jar files and put into SoapUI lib folder.

* poi-3.8-beta5-20111217.jar
* poi-examples-3.8-beta5-20111217.jar
* poi-excelant-3.8-beta5-20111217.jar
* poi-ooxml-3.8-beta5-20111217.jar
* poi-ooxml-schemas-3.8-beta5-20111217.jar
* poi-scratchpad-3.8-beta5-20111217.jar
* dom4j-1.6.1.jar

In this post, I have created a SoapUI project for “ConversionRate“ API, and created a test case and teststep with name “ConversionRate”. So here I need to run this test for the set of data, where data is in external excel file.

[](http://4.bp.blogspot.com/--0E9-zAq9bo/Uq2I5_9J36I/AAAAAAAAAgc/CgbpAg43Y30/s1600/SoapExcel.jpg)

I have created a “ReadXLSFile” groovy step and write below code to read data from “Book1.xlsx” file for the “ConversionRate” API method. Below is excel file data:

|  |  |
| --- | --- |
| To | From |
| USD | ALL |
| AFA | DZD |
| AWG | BSD |
| BSD | BDT |

**GroovyScript Code:**

[?](http://roadtoautomation.blogspot.in/2013/12/road-to-reading-data-from-excel-file-in.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | import org.apache.poi.ss.usermodel.\*;  import org.apache.poi.hssf.usermodel.\*;  import org.apache.poi.xssf.usermodel.\*;  import org.apache.poi.ss.util.\*;  import org.apache.poi.ss.usermodel.\*;  import java.io.\*;    class ExcelReader {      def readData() {          def path = "E:\\Automation-WorkArea\\APITest\\Book1.xlsx";          InputStream inputStream = new FileInputStream(path);          Workbook workbook = WorkbookFactory.create(inputStream);          Sheet sheet = workbook.getSheetAt(0);            Iterator rowIterator = sheet.rowIterator();          rowIterator.next()          Row row;          def rowsData = []          while(rowIterator.hasNext()) {               row = rowIterator.next()               def rowIndex = row.getRowNum()               def colIndex;               def rowData = []               for (Cell cell : row) {                   colIndex = cell.getColumnIndex()                    rowData[colIndex] = cell.getRichStringCellValue().getString();               }               rowsData << rowData           }           rowsData    }   }    def groovyUtils = new com.eviware.soapui.support.GroovyUtils(context)  def myTestCase = context.testCase    ExcelReader excelReader = new ExcelReader();  List rows = excelReader.readData();  def d = []  Iterator i = rows.iterator();  while( i.hasNext()){           d = i.next();           myTestCase.setPropertyValue("From", d[0])           myTestCase.setPropertyValue("To", d[1])           testRunner.runTestStepByName( "ConversionRate")  } |

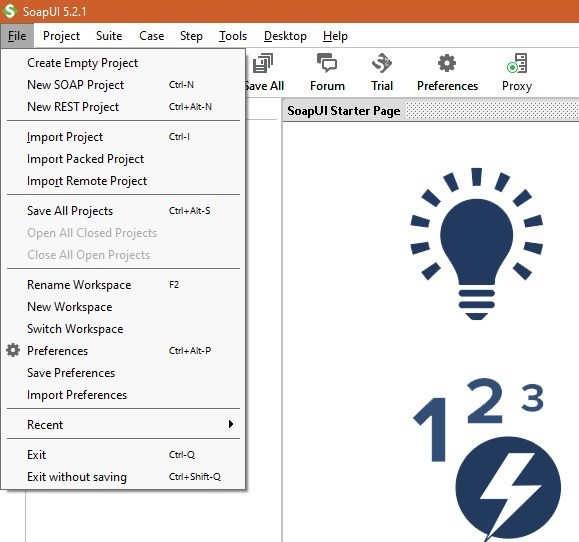
**Description:**

* ExcelReader class containing a function “readData” to read data from “Book1.xlsx" file.
* myTestCase.setPropertyValue("From", d[0]) and myTestCase.setPropertyValue("To", d[1])  are  used to set testcase “from” and “To” properties  value.
* testRunner.runTestStepByName( "ConversionRate") this steps is used to run test step “ConversionRate”

So in this way when I run test “ReadXLSFile” read data from xls file and will execute test “ConversionRate” for each setoff data

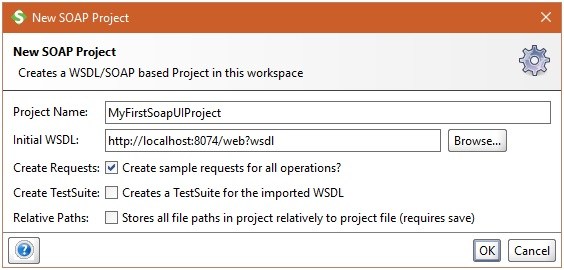
# Steps to Create a New SOAP Project:

**Step 1:** Click on the File link and from the drop down menu click on **New SOAP Project**. There is shortcut link to open a new project window using (Ctrl+N). Image below demonstrate about to open a new project using SoapUI tool.

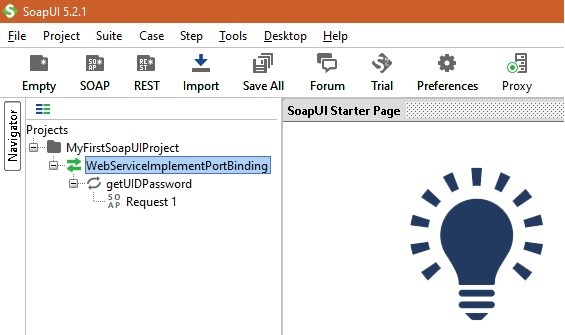
[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot1.jpg)

**Step 2:** Above step will open a new SoapUI project window as shown I below image which will accept below two details.

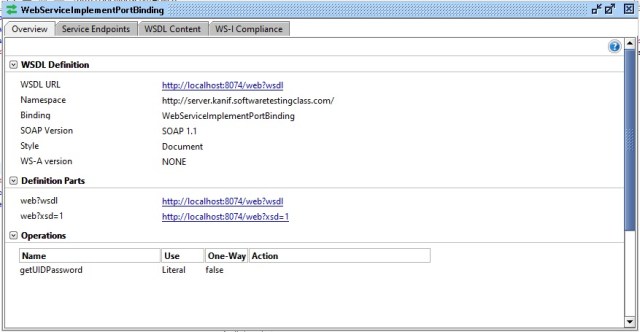
1. Project Name as indicated below in diagram.
2. WSDL file location for SOAP Web Service. This WSDL location was created after running the webservice project developed in the last tutorial “***Web Service Sample Project and its Testing***”. Use the URL for WSDL which was created in that web service project.

[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot1-1.jpg)

**Step 3:**Click OK, it will list down all the operations in the left window pane. Since the WSDL file that we are using has single operation as ‘’getUIDPassword’, it will display that operation there.

[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot2.jpg)

WSDL Description:

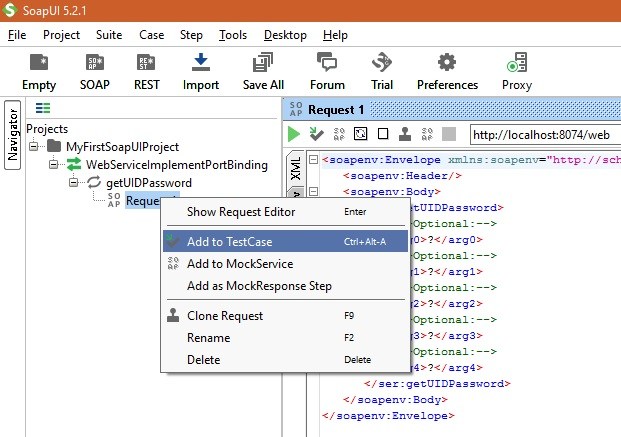
[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/WSDL-Description.jpg)

**Step 4:** Click on the “Request 1” link on the window pane on left hand side. Here a request and response instant test window will get open as demonstrated in the below Image. Here place the values in the arguments 0 to 4 in the left hand side window and trigger the start button, the response from web service will be visible on the right hand side window.

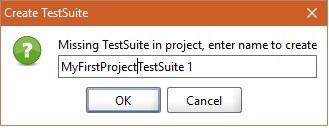
[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/Request-response-model.jpg)

## Steps to Prepare the Test Suite and Test cases with in the suite:

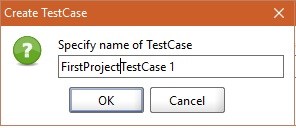
**Step 1:**Right click on the operation which was imported from the WSDL file and from the drop down window select “Add to TestCase”.  It is demonstrated in the below image.

[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot3.jpg)

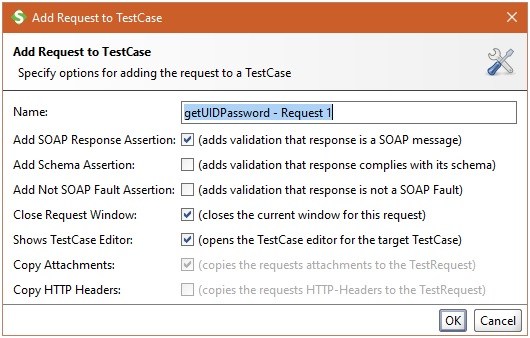
**Step 2:** After clicking “Add to TestCase”, a small window will be opened to create TestSuite where you can give this test suite a name. Below in the image given name is “MyFirstProjectTestSuite1”.

[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot4.jpg)

**Step 3:** After click Ok button of TestSuite window, another window will pop up asking to specify TestCase name. Specify the desired name there and click on the ok button.

[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot5.jpg)

**Step 4:**Next pop up window will ask you to add WSDL request on which current test case is testing on. Under the “Name:” include the “getUIDPassword – Request 1”. Keep the checkboxes indicated below in the image as checked. This will allow to add SOAP Response Assertion, Close Request window after test and will open the TestCase editor for the target TestCase. Click on the OK button.

[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot4-1.jpg)

Up till now, we are familiar about how to use SoapUI interface to create a SOAP project, add a web service operation in it. On the operation how to add a test suite and the test cases with in the test suite. Now let’s work on preparing the complete test suite for SOAP Web Service project.

## First Project Test Case 1:

In the last tutorial, we prepared a web service which accepts input details of the customer such as name, phone number, email Id, address and account number. If these input details matches the details present in the Web Service, it responds with UID and Password else exception message. Therefore to test such operation in web service, we have two test cases one with valid test string input set and other with Invalid test string input set as shown below:

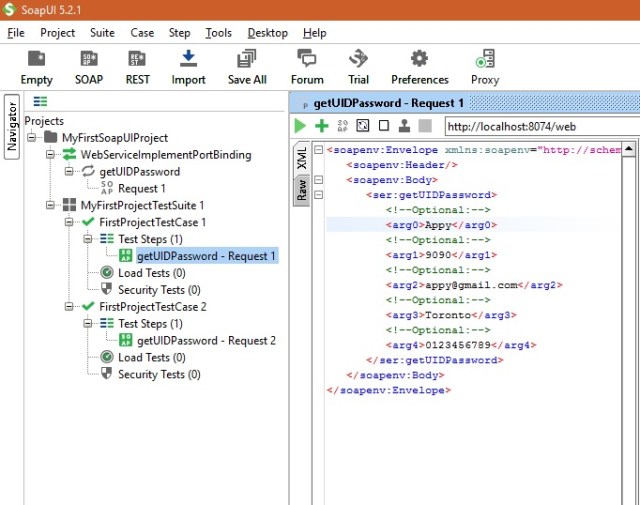
**Valid Test String:**“Appy”, “9090”,”appy@gmail.com”,”Toronto”,”0123456789″.

**Invalid Test String:**“wrong”, “wrong”,”wrong@gmail.com”,”Wrong”,”0123456789″.

Like any other testing let’s prepare the test case first as shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input Value** | **Response Output** |
| **WS1** | Valid test case, if this string is passed as “Appy”, “9090”, “appy@gmail.com”, “Toronto”, “0123456789”. | “Appy”, “9090”, “appy@gmail.com”, “Toronto”, “0123456789”. | XML format message with response as “UID and Password successfully created. UID: SoftwareTesting Classes; PWD: Let’sLearn”. |
| **WS2** | Valid test case, if this string is passed as “wrong”, “wrong”, “wrong@gmail.com”, “Wrong”, “0123456789”. | “wrong”, “wrong”, “wrong@gmail.com”, “Wrong”, “0123456789” | XML format message with response as “Input data does not match the details requested, UID and Password not created”. |

Let’s work on **test case WS1** in SoapUI “FirstProjectTestCase1” as shown in the below image. Here in the input XML SOAP message we have given the valid test string input arguments.

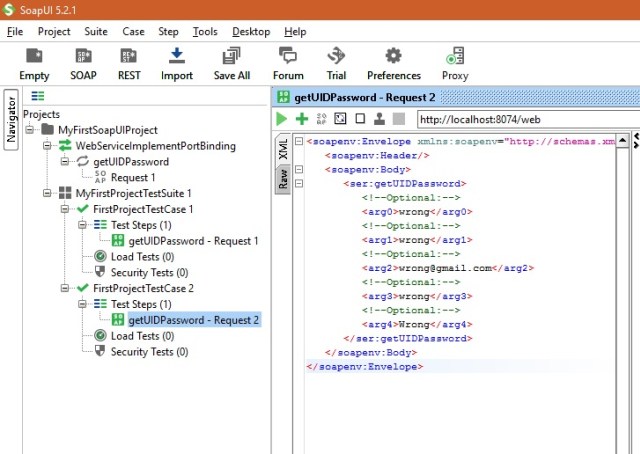
[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot6.jpg)

**Test Response:**If you individually run this Test Case, below SOAP XML message response will be visible on right hand pane. Since we entered all the valid arguments therefore we received the expected response as “UID and Password successfully created. UID: SoftwareTestingClasses; PWD: Let’sLearn”. This passed the test case.

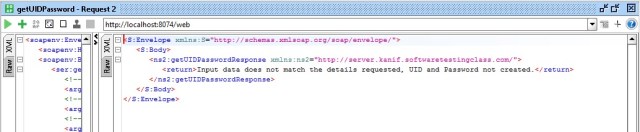
[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot8.jpg)

## **First Project Test Case 2**:

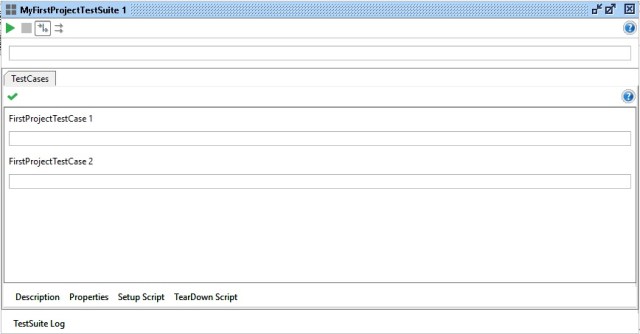
Next, let’s work on **test case WS2** in SoapUI “FirstProjectTestCase2” as shown in the below image. Here in the input XML SOAP message we have given the invalid test string input arguments.

[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot7.jpg)

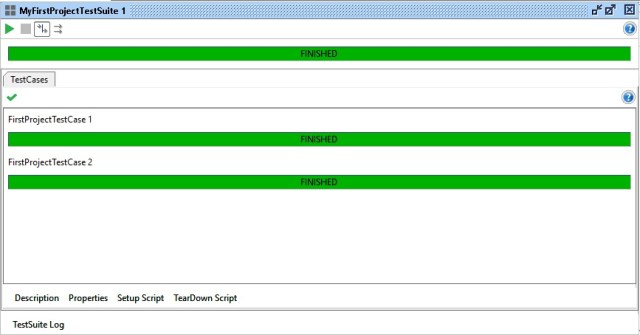
**Test Result:**If you individually run this Test Case 2, below SOAP XML message response will be visible on right hand pane. Since we entered all the invalid arguments therefore we received the expected response as “Input data does not match the details requested, UID and Password not created”. This passed the test case.

[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot9.jpg)

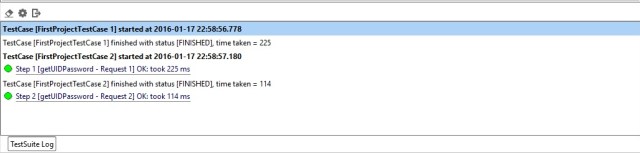
**Test Suite 1:**Let’s try to execute the entire test suite which has above two test cases. Click on the test suite “MyFirstProjectTestSuite1” and click the green arrow button, it will trigger complete test suite. Below image demonstrates when test suite prepares to get executed.

[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot10.jpg)

After completion of test suite execution, below screen will be visible. Since both test cases have passed successfully, entire test suite including the tests look green.

[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot11.jpg)

**Test Suite Log:**SoapUI tool provides the feature using which we can check the test suite log as shown in the below image.

[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot12.jpg)

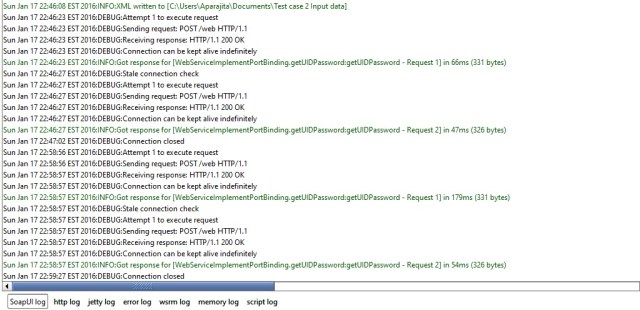
**Memory Log:**SoapUI tool provides the feature using which we can check the memory log as shown in the below image. With this feature we can monitor the memory usage while executing the test suite.

[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot13.jpg)

**Http Log:**SoapUI tool provides the feature using which we can check the Http log as shown in the below image.

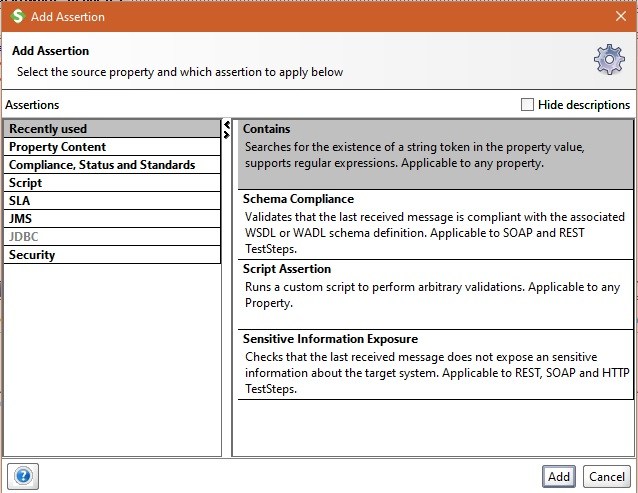
[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot14.jpg)

**SoapUI Log:**SoapUI tool provides the feature using which we can check the SoapUI log as shown in the below image. This will provide the step by step events that take place while making Web Service call.

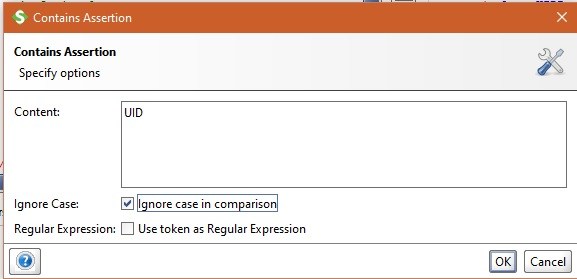
[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot15.jpg)

## Brief Introduction to Assertions:

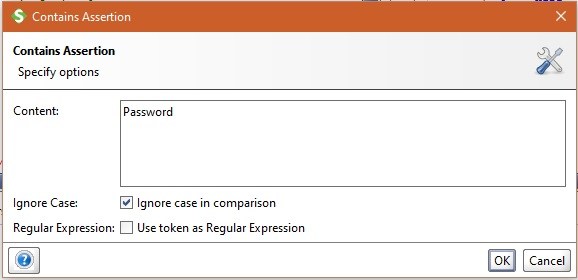
Click on the “Add Assertion” option which can be added to each test case, in this tutorial lets add assertion through which we can check “UID” and “Password” is present in the response message.

[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-assertion-1.jpg)

Here in the Content space add “UID” and check the checkbox for Ignore case. This will be an Assertion “Contains UID” added to the TestCase1.

[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-assertion-2.jpg)

Similarly add another assertion “Contains Password” to the TestCase1 as indicated in below image.

[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-assertion-3.jpg)

Now run the TestCase1 and as we know the response message will have both UID and Password present in the XML message therefore both assertions are going to pass successfully and the output will look like as below.

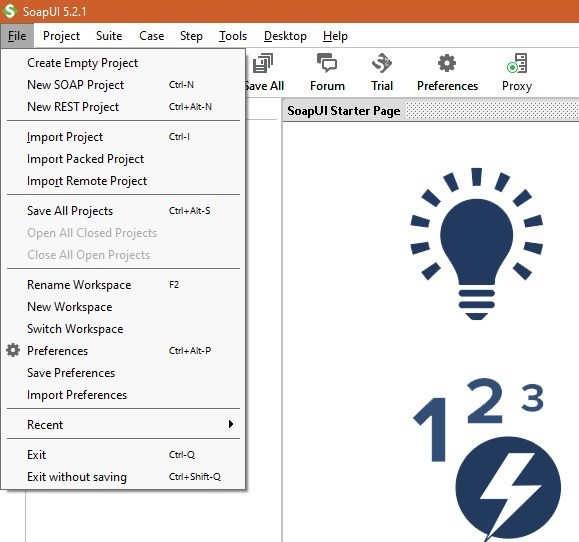
[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-assertion-4.jpg)

Likewise there are following assertions available in SoapUI.

* Schema compliance: It validates the last received message as response is WSDL schema compliant or not. It is applicable to both SOAP and RESTful Web services.
* Script assertions: Runs a custom script to run arbitrary validations.
* Sensitive information exposure: This to make sure that the last response message does not contain any sensitive information about the target system.

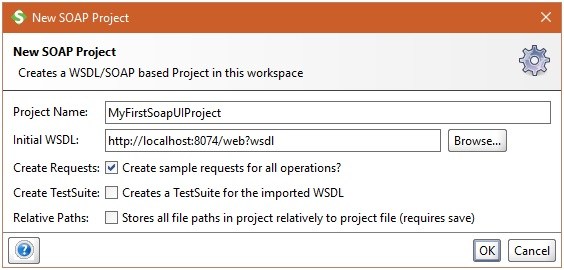
# Steps to Create a New SOAP Project:

**Step 1:** Click on the File link and from the drop down menu click on **New SOAP Project**. There is shortcut link to open a new project window using (Ctrl+N). Image below demonstrate about to open a new project using SoapUI tool.

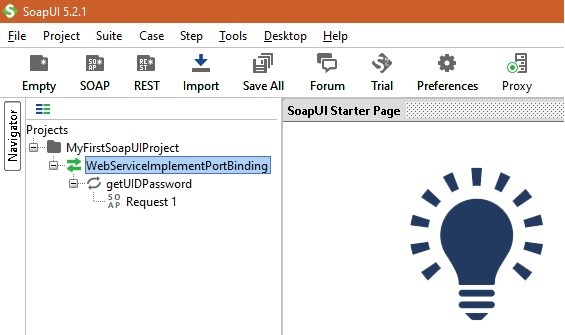
[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot1.jpg)

**Step 2:** Above step will open a new SoapUI project window as shown I below image which will accept below two details.

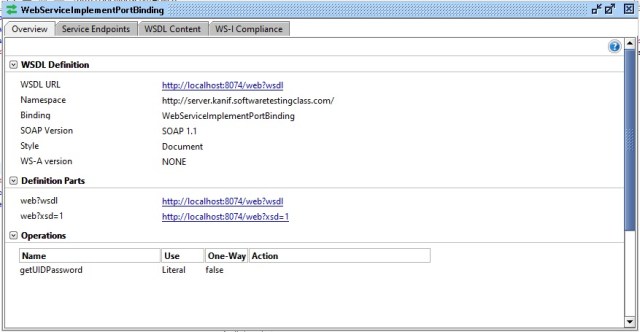
1. Project Name as indicated below in diagram.
2. WSDL file location for SOAP Web Service. This WSDL location was created after running the webservice project developed in the last tutorial “***Web Service Sample Project and its Testing***”. Use the URL for WSDL which was created in that web service project.

[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot1-1.jpg)

**Step 3:**Click OK, it will list down all the operations in the left window pane. Since the WSDL file that we are using has single operation as ‘’getUIDPassword’, it will display that operation there.

[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot2.jpg)

WSDL Description:

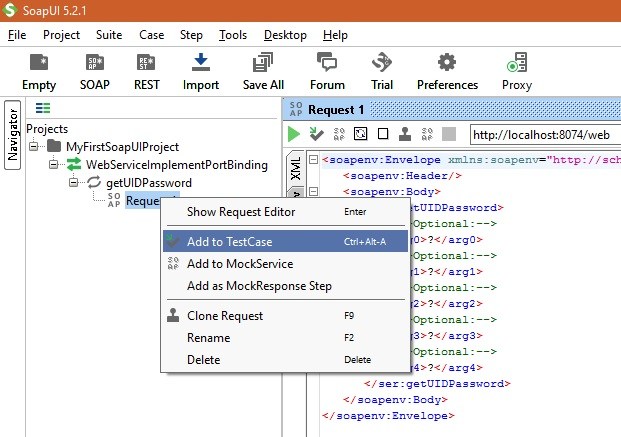
[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/WSDL-Description.jpg)

**Step 4:** Click on the “Request 1” link on the window pane on left hand side. Here a request and response instant test window will get open as demonstrated in the below Image. Here place the values in the arguments 0 to 4 in the left hand side window and trigger the start button, the response from web service will be visible on the right hand side window.

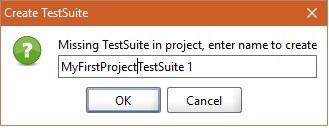
[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/Request-response-model.jpg)

## Steps to Prepare the Test Suite and Test cases with in the suite:

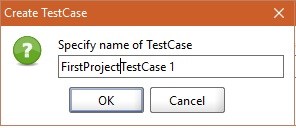
**Step 1:**Right click on the operation which was imported from the WSDL file and from the drop down window select “Add to TestCase”.  It is demonstrated in the below image.

[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot3.jpg)

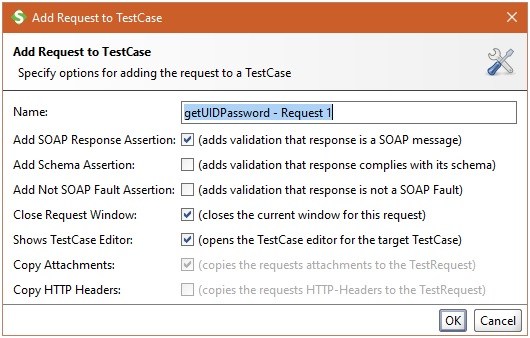
**Step 2:** After clicking “Add to TestCase”, a small window will be opened to create TestSuite where you can give this test suite a name. Below in the image given name is “MyFirstProjectTestSuite1”.

[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot4.jpg)

**Step 3:** After click Ok button of TestSuite window, another window will pop up asking to specify TestCase name. Specify the desired name there and click on the ok button.

[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot5.jpg)

**Step 4:**Next pop up window will ask you to add WSDL request on which current test case is testing on. Under the “Name:” include the “getUIDPassword – Request 1”. Keep the checkboxes indicated below in the image as checked. This will allow to add SOAP Response Assertion, Close Request window after test and will open the TestCase editor for the target TestCase. Click on the OK button.

[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot4-1.jpg)

Up till now, we are familiar about how to use SoapUI interface to create a SOAP project, add a web service operation in it. On the operation how to add a test suite and the test cases with in the test suite. Now let’s work on preparing the complete test suite for SOAP Web Service project.

## First Project Test Case 1:

In the last tutorial, we prepared a web service which accepts input details of the customer such as name, phone number, email Id, address and account number. If these input details matches the details present in the Web Service, it responds with UID and Password else exception message. Therefore to test such operation in web service, we have two test cases one with valid test string input set and other with Invalid test string input set as shown below:

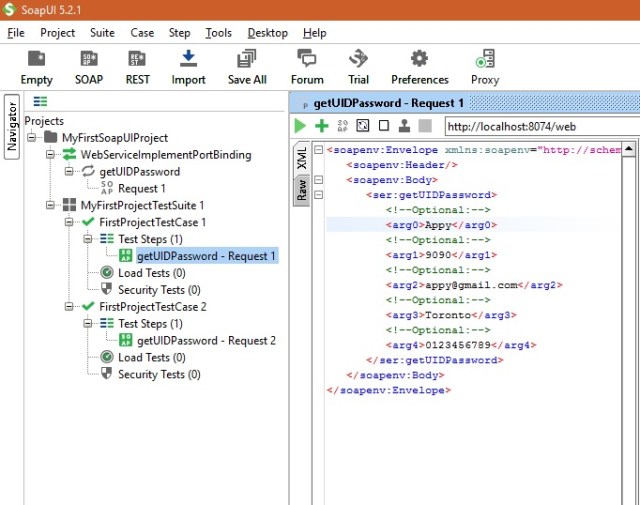
**Valid Test String:**“Appy”, “9090”,”appy@gmail.com”,”Toronto”,”0123456789″.

**Invalid Test String:**“wrong”, “wrong”,”wrong@gmail.com”,”Wrong”,”0123456789″.

Like any other testing let’s prepare the test case first as shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input Value** | **Response Output** |
| **WS1** | Valid test case, if this string is passed as “Appy”, “9090”, “appy@gmail.com”, “Toronto”, “0123456789”. | “Appy”, “9090”, “appy@gmail.com”, “Toronto”, “0123456789”. | XML format message with response as “UID and Password successfully created. UID: SoftwareTesting Classes; PWD: Let’sLearn”. |
| **WS2** | Valid test case, if this string is passed as “wrong”, “wrong”, “wrong@gmail.com”, “Wrong”, “0123456789”. | “wrong”, “wrong”, “wrong@gmail.com”, “Wrong”, “0123456789” | XML format message with response as “Input data does not match the details requested, UID and Password not created”. |

Let’s work on **test case WS1** in SoapUI “FirstProjectTestCase1” as shown in the below image. Here in the input XML SOAP message we have given the valid test string input arguments.

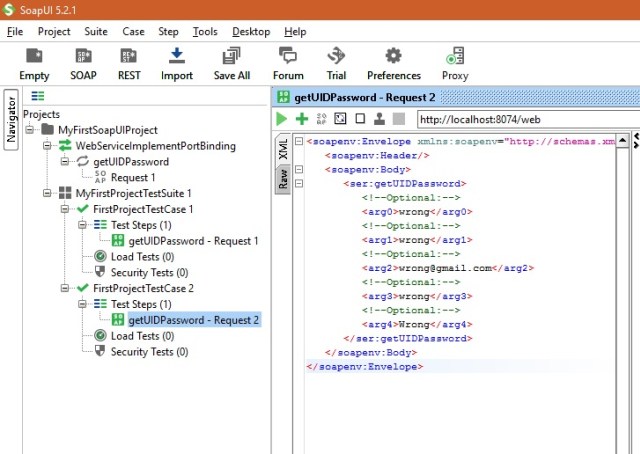
[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot6.jpg)

**Test Response:**If you individually run this Test Case, below SOAP XML message response will be visible on right hand pane. Since we entered all the valid arguments therefore we received the expected response as “UID and Password successfully created. UID: SoftwareTestingClasses; PWD: Let’sLearn”. This passed the test case.

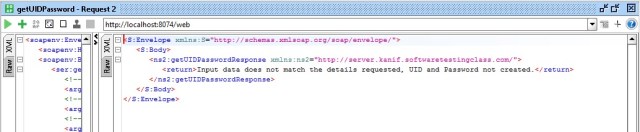
[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot8.jpg)

## **First Project Test Case 2**:

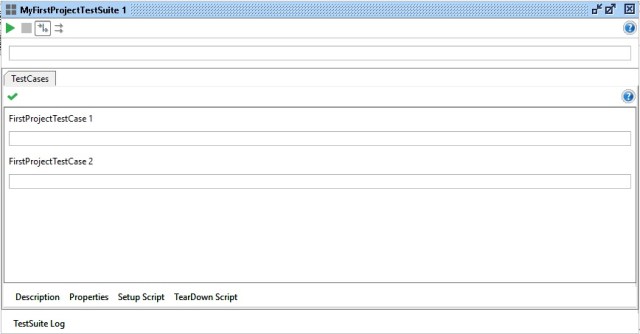
Next, let’s work on **test case WS2** in SoapUI “FirstProjectTestCase2” as shown in the below image. Here in the input XML SOAP message we have given the invalid test string input arguments.

[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot7.jpg)

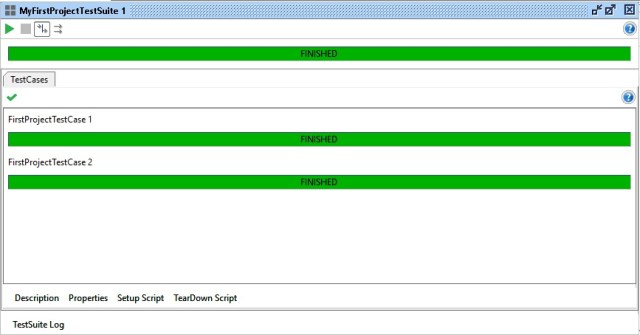
**Test Result:**If you individually run this Test Case 2, below SOAP XML message response will be visible on right hand pane. Since we entered all the invalid arguments therefore we received the expected response as “Input data does not match the details requested, UID and Password not created”. This passed the test case.

[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot9.jpg)

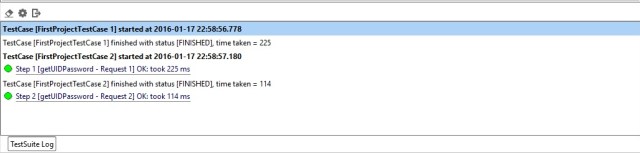
**Test Suite 1:**Let’s try to execute the entire test suite which has above two test cases. Click on the test suite “MyFirstProjectTestSuite1” and click the green arrow button, it will trigger complete test suite. Below image demonstrates when test suite prepares to get executed.

[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot10.jpg)

After completion of test suite execution, below screen will be visible. Since both test cases have passed successfully, entire test suite including the tests look green.

[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot11.jpg)

**Test Suite Log:**SoapUI tool provides the feature using which we can check the test suite log as shown in the below image.

[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot12.jpg)

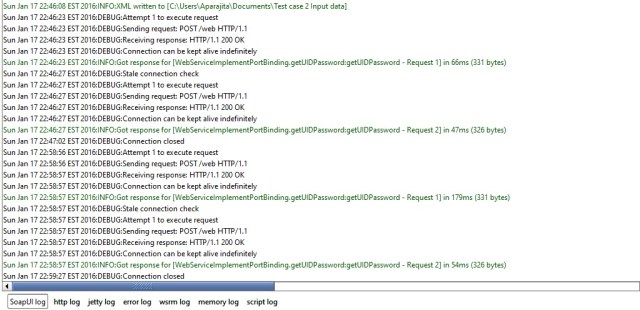
**Memory Log:**SoapUI tool provides the feature using which we can check the memory log as shown in the below image. With this feature we can monitor the memory usage while executing the test suite.

[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot13.jpg)

**Http Log:**SoapUI tool provides the feature using which we can check the Http log as shown in the below image.

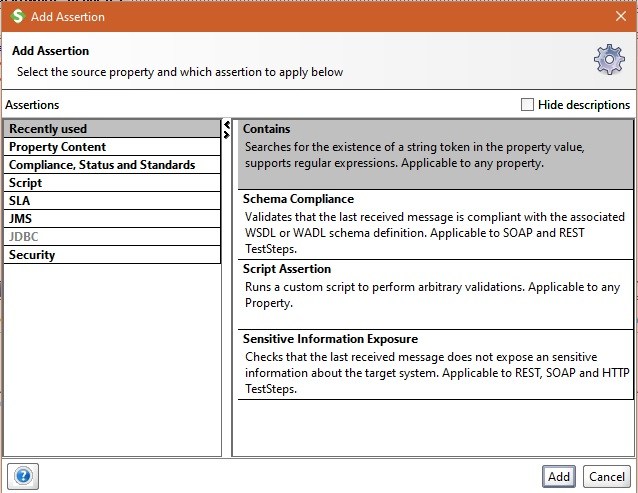
[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot14.jpg)

**SoapUI Log:**SoapUI tool provides the feature using which we can check the SoapUI log as shown in the below image. This will provide the step by step events that take place while making Web Service call.

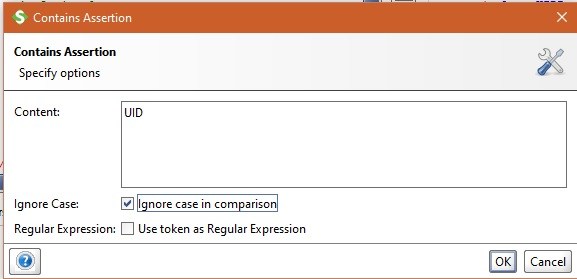
[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-Project-screenshot15.jpg)

## Brief Introduction to Assertions:

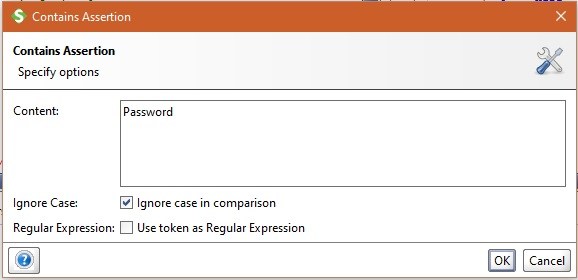
Click on the “Add Assertion” option which can be added to each test case, in this tutorial lets add assertion through which we can check “UID” and “Password” is present in the response message.

[](https://i0.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-assertion-1.jpg)

Here in the Content space add “UID” and check the checkbox for Ignore case. This will be an Assertion “Contains UID” added to the TestCase1.

[](https://i2.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-assertion-2.jpg)

Similarly add another assertion “Contains Password” to the TestCase1 as indicated in below image.

[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-assertion-3.jpg)

Now run the TestCase1 and as we know the response message will have both UID and Password present in the XML message therefore both assertions are going to pass successfully and the output will look like as below.

[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2016/02/SoapUI-assertion-4.jpg)

Likewise there are following assertions available in SoapUI.

* Schema compliance: It validates the last received message as response is WSDL schema compliant or not. It is applicable to both SOAP and RESTful Web services.
* Script assertions: Runs a custom script to run arbitrary validations.
* Sensitive information exposure: This to make sure that the last response message does not contain any sensitive information about the target system.