Practical 07 - Test and Load Test your WebService

Objectives:

- To interact with and test the WebService you created in Practical 06
- To load test the WebService you created in Practical 06

Tools to be used in this practical:

- Oracle Java JDK v1.7 (aka Java 7) - www.java.com (to support SOAPUI)
- SoapUI Tester - www.soapui.org
- Tomcat Application Server (assumed already running, and hosting your WebService application, as at the end of the last exercise)

Instructions:

1. Launch SOAPUI
    - 1.1. Locate folder where you unpacked SOAPUI, e.g. /home/usr/development/SoapUI-5.x.x
    - 1.2. Within that folder, locate the the bin sub-directory
    - 1.3. Within the bin subdirectory, locate and launch the soapui.sh item
    - 1.4. Wait for SOAPUI to launch, and the desktop to appear
    - 1.5. If the "SOAPPUI Starter Page" appears, close it ([x] in top left of Window)
2. Create a “New SOAP project” within SoapUI
    - 2.1. Within SoapUI, select "File", then "New SOAP project"
    - 2.2. In the "New SOAP Project" window:
        - 2.2.1.Set "Project Name" to "ShapeAreas"
        - 2.2.2.Set "Initial WSDL" to the WSDL URL you noted at the end of Practical 06.
        - 2.2.3.Make sure that "Create Requests" is ticked.
        - 2.2.4.Make sure that "Create TestSuite" is ticked.
        - 2.2.5.Make sure that "Relative Paths" is ticked.
        - 2.2.6.Check again that all the settings above are correct, the click <OK>
    - 2.3. When asked to "Save project ShapeAreas", choose a suitable location (e.g. /home/user/development) then click <Save>
    - 2.4. Wait for the project to finish building.
    - 2.5. SOAPUI will attempt to generate a TestSuite based on the WSDL of the WebService.
    - 2.6. When "Generate Test Suite" appears
        - 2.6.1.Ensure that "TestSuite" is set to <create>
        - 2.6.2.Ensure that "Style" is set to "One TestCase for each operation"

- 2.6.3. Under "Request Content", ensure that "Create new empty requests" is selected.
- 2.6.4. Ensure that under "Operations" all of the Operations are selected
- 2.6.5. Ensure the "Generate LoadTest" is *not* selected.
- 2.6.6. Click <OK>

2.7. Under "Generate TestSuite", the default name is suitable, just click <OK>

2.8. Save the project so far by clicking on the "Save" icon on the tool bar.

3. Interact with the WebService:
   - 3.1. Locate and open the "ShapeAreasSoapBinding" item.
   - 3.2. Within the "ShapeAreasSoapBinding" item, locate and open each operation item (of 3).
   - 3.3. Within each Operation, locate and double click the "Request1" item
   - 3.4. Set the input fields in the "Request" side
   - 3.5. Click on "Submit Request" (green arrow to top left of request)
   - 3.6. Wait for the response to be returned
   - 3.7. Examine the response.
   - 3.8. Try the tests attempting to feed in bad data, e.g. a string where a numeric input is expected.
     - 3.8.1. What do you think of the error handling of this WebService?
     - 3.8.2. Did you write any code to produce SOAP faults?
   - 3.9. Break the request XML (e.g. remove one of the closing tags), and submit.
     - 3.9.1. Is this what we expected?
     - 3.9.2. Repeat these steps for the other operations...
4. Add further tests to the ShapeAreasSoapBinding TestSuite.
   - 4.1. Locate and open the "ShapeAreasSoapBinding TestSuite" item.
   - 4.2. Within the "ShapeAreasSoapBinding TestSuite" item select any one of the three TestCases for your operations, and open it.
   - 4.3. Within the opened operation, locate and open the "Test Steps (1)" item.
   - 4.4. Within the "Test Steps (1)" item, locate and double click on the operation name item.
   - 4.5. In the request editor, set the inputs and execute to ensure that the test passes.
   - 4.6. Note the response time of this test.
   - 4.7. Save the project.
   - 4.8. Add a response time test to the test suite
     - 4.8.1. Click on the "Assertions" button (bottom left below the request.
     - 4.8.2. Click the "+" item to add an assertion
     - 4.8.3. In the "Add Assertion" window:
     - 4.8.4. Select the "SLA" category.
     - 4.8.5. Under "SLA" select the "Response SLA" item - read the description of this assertion.
     - 4.8.6. Click <Add>
     - 4.8.7. Under "Configure Response SLA Assertion"

4.8.7.1. Set "Specify desired response time" to a reasonable time (e.g. 150% of the response time noted in the test above)

4.8.7.2. Click <OK>

4.8.8. Save the project

4.8.9. Re-run the test a few time, noting the varying response times.

5. Load test your WebService.

5.1. Below the "Test Steps(1)" item of the TestCase you selected above, locate the "Load Tests (0)" item.

5.2. Right-click the "Load Tests (0)" item and select "New Load Test"

5.3. In the "Specify name of LoadTest", accept the default name by pressing <OK>

5.4. Wait a few seconds for the load test to build and open.

5.5. Note the settings on the load test, in particular:

5.5.1. Limit (the period the test will run for)

5.5.2. Threads (number of simultaneous requests)

5.5.3. Test Delay and Random (gaps between calls by each thread calling the web service, and variance in those gaps)

5.6. Launch the test (green arrow in top left of the "LoadTest" windows)

5.7. Note the max, min and average response times of the web service under load.

5.8. Click on the "Graph" icons beside the "Launch Test" green arrow to see a graphical representation of the progress of the test.

6. Crank up the test rate

6.1. Increase number of threads, decrease test delay.

6.2. Re-run the test.

6.3. Repeat until the SLA begins to be breached.

7. Add Additional Tests to the TestSuite to confirm that your WebService is working

7.1. Add a test to check for "if we put in bad data, we get back a SOAP fault"

7.1.1. Below the "Test Steps(1)" item of the TestCase you selected above, locate the "SOAP" item.

7.1.1.1. Right click the SOAP item, and select "Clone"

7.1.1.2. In the Clone TestStep window:

7.1.2. Note the available options

7.1.3. Set the name as you see fit

7.1.4. Click <OK>

7.1.5. Locate the Cloned SOAP test item, and double click it to edit it.

7.1.6. Put in valid data, then run the Cloned SOAP test item - confirm that it passes

7.1.7. Put in invalid data, then re-run the Cloned SOAP test item - does it pass or fail?

7.1.8. As we've put in invalid data, we expect (want?) it to fail with a SOAP fault.

7.1.8.1. Add an assertion of type "Compliance, Status and Standards" | "SOAP Fault"

7.1.8.2. Remove any existing assertion of type "Not SOAP Fault"

7.1.9. Run the test again - confirm that the test is passed (even though the WebService is returning a SOAP Fault)

7.2. Add a test to check for "if we put in good data, we get back a correct answer"

7.2.1. Clone another copy of on of the SOAP Test Items
7.2.2. Locate the Cloned SOAP test item, and double click it to edit it.
7.2.3. Put in valid data
7.2.4. Run the Cloned SOAP test item - confirm that it passes
7.2.5. Note the name of the Attribute holding the answer.
7.2.6. Look at the answer in the returned SOAP message - is it correct.
7.2.7. Add an assertion of type "Property Content" | "XPath Match"
7.2.8. Set the XPath Expression to "node()", and click "Select from Current" to set the Expected Result to the current returned result.
7.2.9. You can try the "Select from Current" and "Test" buttons
7.2.10. Also, note the other options
7.2.11. Click Save
7.2.12. Run this test again - confirm that the test is passed

8. Run the whole test case
   8.1. Locate the relevant test case item
   8.2. Double click on it to open in the editor
   8.3. You should be able to see each test step you have created
   8.4. Execute the test case (using the green "run" arrow button)
   8.5. Note the success or failure of the test case overall.