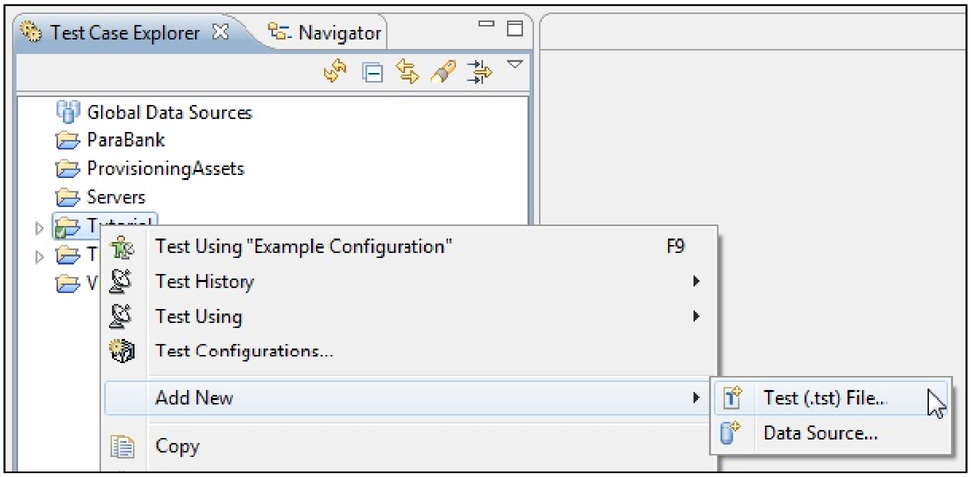
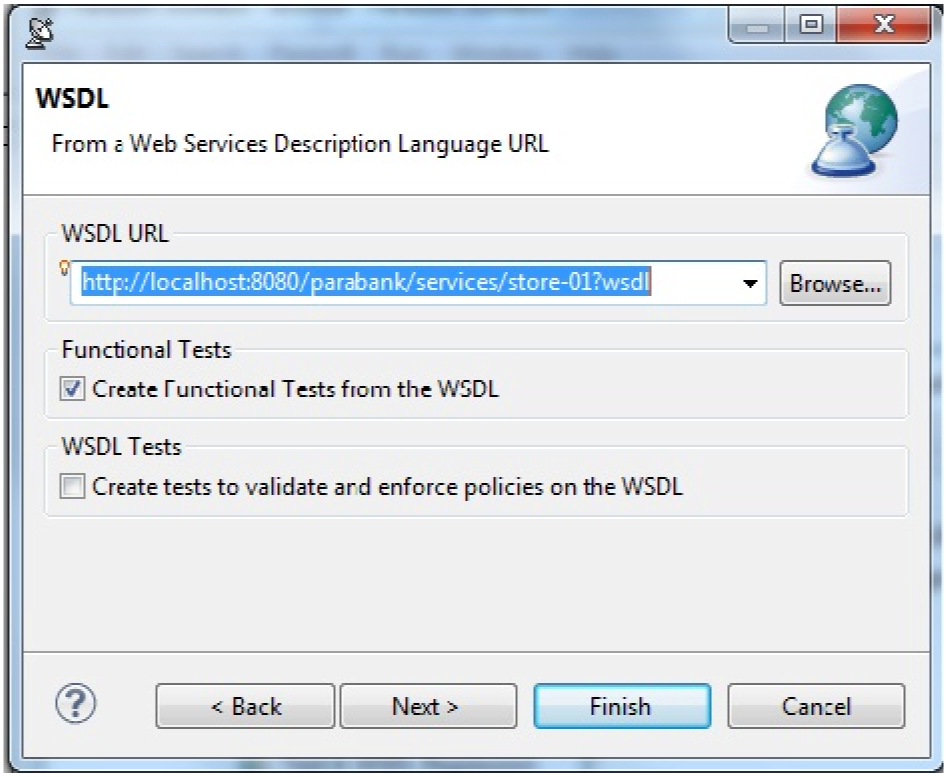
Creating Test Suites for Unit Tests

For this example, we add a new Test (.tst) file to the project created in the previous lesson.

1. Right-click the project from the previous exercise, then choose **Add New> Test (.tst) File** from the shortcut menu.



2. Enter a name for the file (e.g., `functional test lesson`), then click **Next**.
3. Select **SOA> WSDL**, then click **Next**.
4. Select `http://localhost:8080/parabank/services/store-01?wsdl` from the **WSDL URL** field.
5. Ensure that **Create Functional Tests from the WSDL** is enabled.



WSDL

From a Web Services Description Language URL

WSDL URL

http://localhost:8080/parabank/services/store-01?wsdl

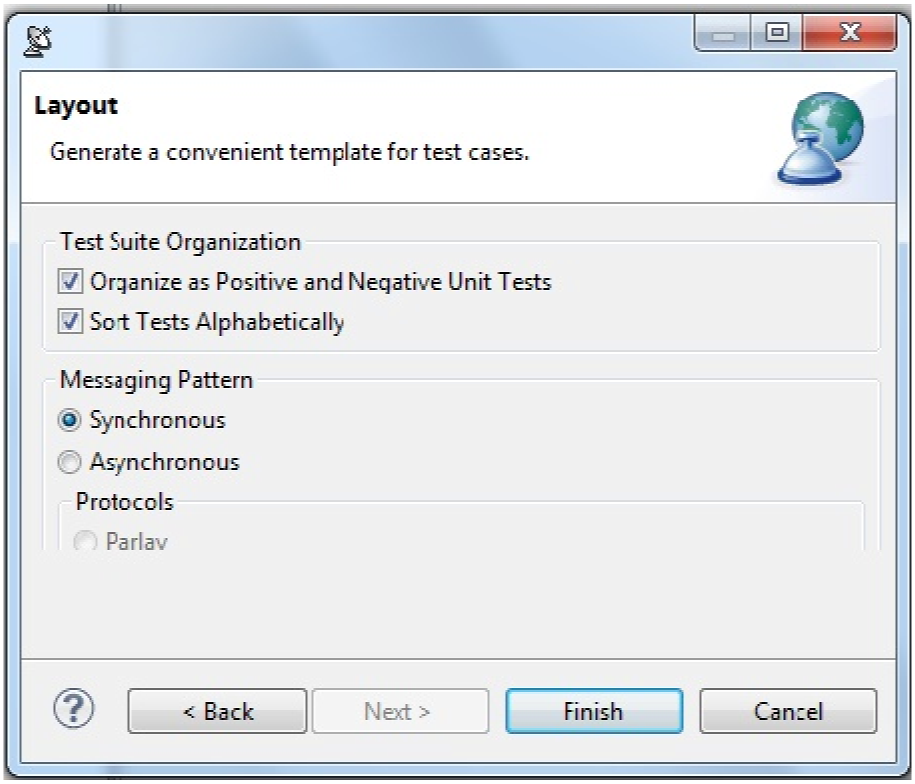
Functional Tests

☑ Create Functional Tests from the WSDL

WSDL Tests

□ Create tests to validate and enforce policies on the WSDL

6. Click **Next** to accept the default settings until you reach the **Layout** dialog.
7. Enable **Organize as Positive and Negative Unit Tests**.



Layout

Generate a convenient template for test cases.

Test Suite Organization

☑ Organize as Positive and Negative Unit Tests

☑ Sort Tests Alphabetically

Messaging Pattern

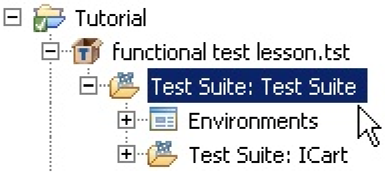
● Synchronous

○ Asynchronous

Protocols

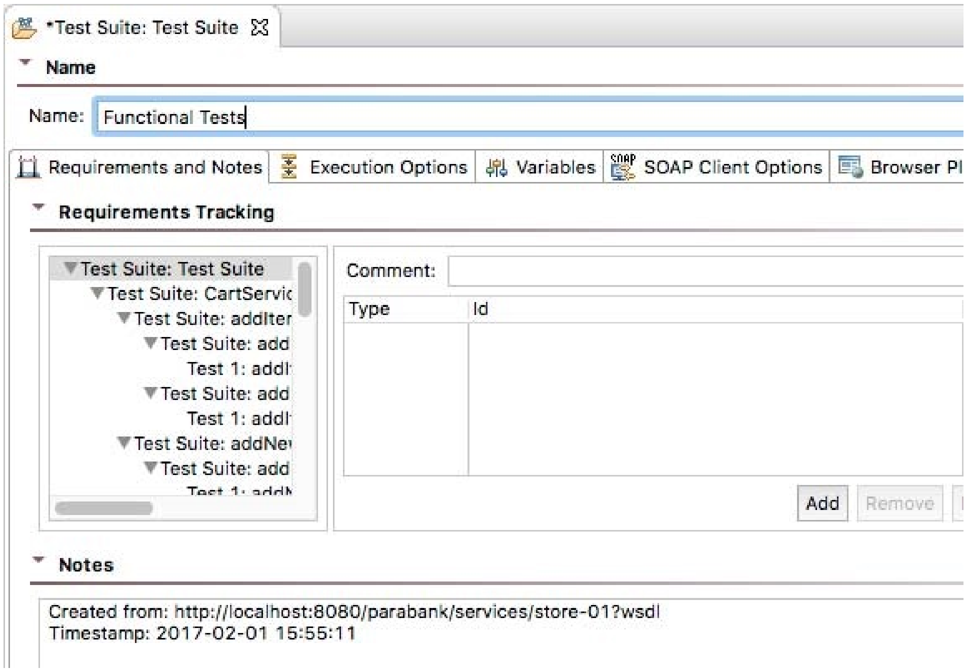
○ Parlay

8. Click the **Finish** button. The newly created test suite displays in the **Test Case Explorer**.



- Tutorial
  - functional test lesson.tst
    - Test Suite: Test Suite
      - Environments
      - Test Suite: ICart

9. Double-click the new **Test Suite: Test Suite** node to open the test suite configuration panel and rename the test suite to `Functional Tests` in the **Name** field.



*Test Suite: Test Suite

Name

Name: Functional Tests

Requirements Tracking

- Test Suite: Test Suite
  - Test Suite: CartServic
    - Test Suite: addIter
      - Test Suite: add
        - Test 1: addI
      - Test Suite: add
        - Test 1: addI
    - Test Suite: addNe
      - Test Suite: add
        - Test 1: addN

Comment:

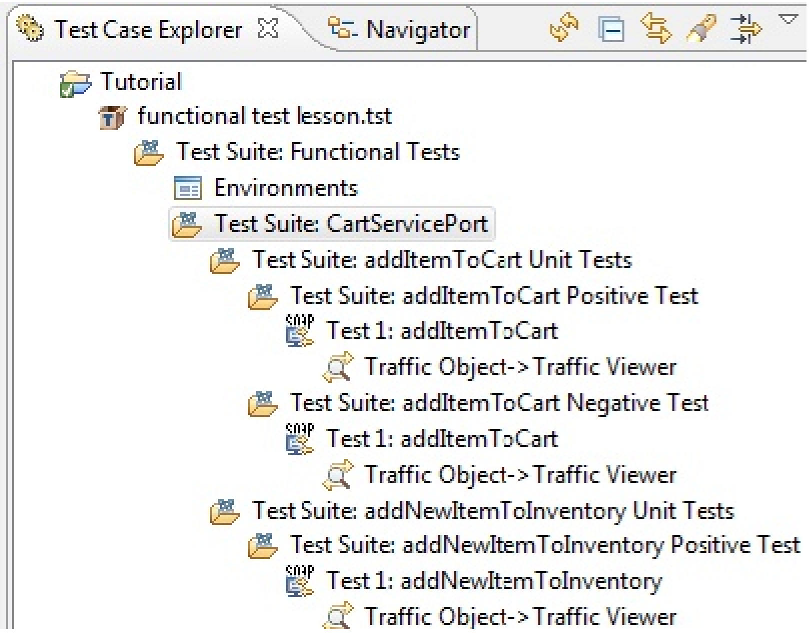
Type	Id

Notes

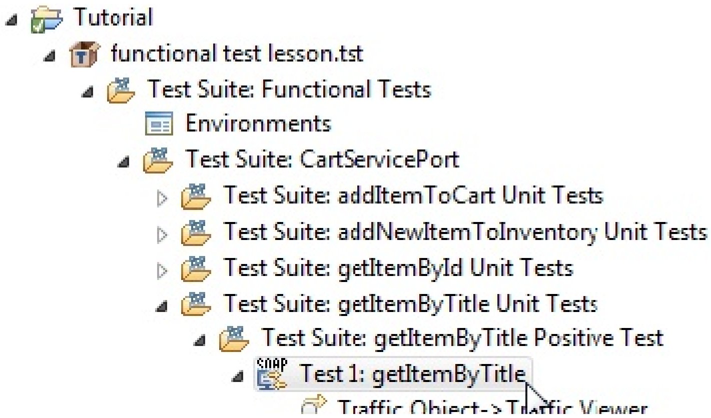
Created from: http://localhost:8080/parabank/services/store-01?wsdl
Timestamp: 2017-02-01 15:55:11

10. Save the file.

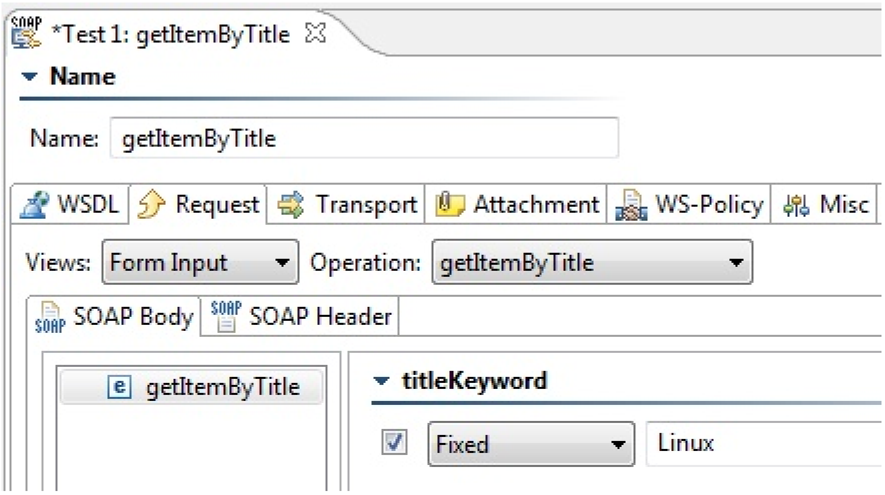
11. Expand the **Functional Tests** test suite to show the **Test Suite: CartServicePort** node, which includes seven other test suites that test each operation of the WSDL.
12. Right-click the **Test Suite: CartServicePort** node, then choose **Expand All**. This will display all seven test suites, as well as every test within each test suite.

    
    Test Case Explorer
Navigator
Tutorial
functional test lesson.tst
Test Suite: Functional Tests
Environments
Test Suite: CartServicePort
Test Suite: addItemToCart Unit Tests
Test Suite: addItemToCart Positive Test
Test 1: addItemToCart
Traffic Object->Traffic Viewer
Test Suite: addItemToCart Negative Test
Test 1: addItemToCart
Traffic Object->Traffic Viewer
Test Suite: addNewItemToInventory Unit Tests
Test Suite: addNewItemToInventory Positive Test
Test 1: addNewItemToInventory
Traffic Object->Traffic Viewer

    Each of the seven test suites contain both a positive and negative test for each operation. Why create both positive and negative tests? Because it is important to test situations where we send expected data as well as unexpected data to the server.
13. Double-click the **Test Suite: getItemByTitle Positive Test> Test 1: getItemByTitle** node to open the test configuration panel.

    
    Tutorial
functional test lesson.tst
Test Suite: Functional Tests
Environments
Test Suite: CartServicePort
Test Suite: addItemToCart Unit Tests
Test Suite: addNewItemToInventory Unit Tests
Test Suite: getItemById Unit Tests
Test Suite: getItemByTitle Unit Tests
Test Suite: getItemByTitle Positive Test
Test 1: getItemByTitle
Traffic Object->Traffic Viewer

14. Choose the **Request> SOAP Body** tabs in the test configuration panel.
15. Enable the option in the **titleKeyword** entry field and choose **Fixed** from the drop-down menu.
16. Enter `Linux` in the empty field and save the file. We will be searching for books with keyword **Linux**.



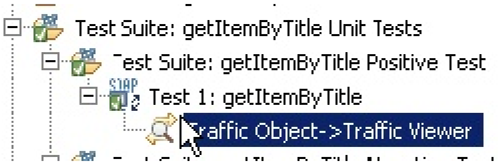
Name: getItemByTitle

Views: Form Input

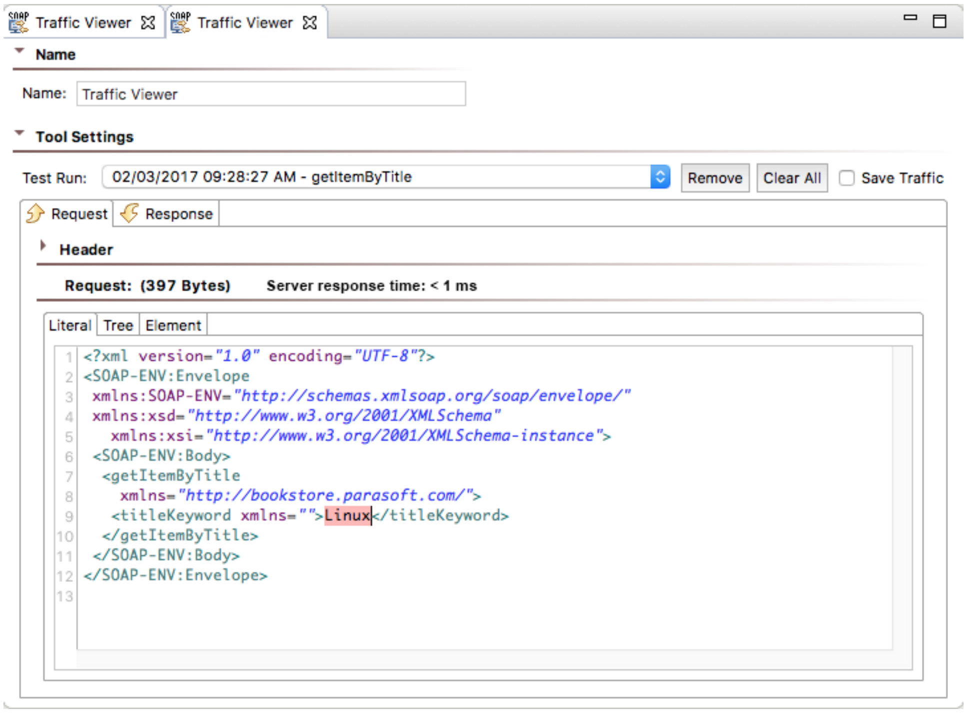
Operation: getItemByTitle

☑ Fixed Linux

17. Select the **Test 1: getItemByTitle** node and click the **Test** tool bar button. The getItemByTitle operation is invoked with the parameter **Linux**.
18. Expand the **Test 1: getItemByTitle** node and double-click the **Traffic Object-> Traffic Viewer** node underneath.


Test Suite: getItemByTitle Unit Tests
Test Suite: getItemByTitle Positive Test
Test 1: getItemByTitle
Traffic Object->Traffic Viewer

The Traffic View opens and displays the traffic that was logged from the test run.



Name: Traffic Viewer

Test Run: 02/03/2017 09:28:27 AM - getItemByTitle

□ Save Traffic

Request: (397 Bytes) Server response time: < 1 ms

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <SOAP-ENV:Body>
  <getItemByTitle
    xmlns="http://bookstore.parasoft.com/">
   <titleKeyword xmlns="">Linux</titleKeyword>
  </getItemByTitle>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

We now have a functional test that tests the getItemByTitle operation of our web service with a single input value. The same sequence of actions can be done to create functional tests for the other operations defined within the WSDL.