Team6

Deliverable 3

Introduction

The overall framework for the automated testing of OpenMRS's API has been built. It is encompassed mostly through the master script. The script successfully compiles the necessary dependencies, takes in test cases, executes the drivers with command line arguments, takes the output, and compares it with the corresponding oracle to create a comprehensive test report.

The majority of the script, and testing is complete. Apart from the addition of more test cases, the testing framework must adapt to the dependencies being placed in the project folder, as well as moving the drivers to the appropriate location.

Framework Architecture

The script runs through several phases to prepare for the testing, initialize testing, and produce a report. This process goes as follows:

Script Structure:

Preparation: The script compiles the OpenMRS with Maven and prepares the dependencies and drivers for the test.

Load: The script takes in a list of test cases available from testing from the *testCases* directory. These will be all the tests that the framework will loop through to complete its complete API testing.

Selection: The script selects an individual test, reads the test name, what it's testing, the information needed to locate and call the oracle and driver, and retrieves the input needed to pass to the driver via command-line arguments.

Testing: The driver (already compiled) is found within the *testCaseExecutables* directory and is called using the inputs from the test file. The dependencies are located within the *src* directory and are imported by the driver, as well as the class being tested. The output is returned from the driver and stored by the script.

Writeback: The driver writes back the calculated output from the driver to an output file in the *temp* directory. The file is named based on the test case name.

Comparison: From the home directory, the script locates the output file from *temp* and the oracle from *oracles* and compares the files directly. The way the oracle and output are formatted allows for the file contents to be easily compared. The success status of the test is then written to the "testReport.html" in the *reports* directory.

Repetition: The steps from **Load** to **Comparison** are repeated for all available tests until a complete report is developed.

Production: The report is finalized by the script, with correct HTML formatting and is then displayed to the user's web browser.

Test Cases

TEST BEGIN
==========
Test Being Run:
DoubleRangeTest1
Testing Range Function for within bounds
OpenMRS Util Component
contains()
DoubleRangeTest
1000.02 9999.1 3000
Success
==========
==========
Test Being Run:
DoubleRangeTest2
Testing Range Function for exclusive upper bound
OpenMRS Util Component
contains()
DoubleRangeTest
1000.02 9999.1 9999.2
Success
==========
==========
Test Being Run:
DoubleRangeTest3
Testing Range Function for exclusive upper bound
OpenMRS Util Component
contains()
DoubleRangeTest
1000.02 9999.1 9999.1
Success
==========
==========
Test Being Run:
DoubleRangeTest4
Testing Range Function for exclusive lower bound
OpenMRS Util Component
contains()

DoubleRangeTest
1000.2597 9999.1 1000.259001
Success
==========
==========
Test Being Run:
DoubleRangeTest5
Testing Range Function for inclusive lower bound with zeros
OpenMRS Util Component
contains()
DoubleRangeTest
0 175.63 0
Success
==========
==========

The method documentation states:

```
* BUG: this method should return false if both ends of the range are null.
* It currently returns true in this case.
*
* checks whether a double is in this range
* @param      d the Double to check for in this range
* @return  true if d is in this range, false otherwise
* @should return true if parameter is in range
* @should return false if parameter is not in range
* @should return false if parameter is equal to high
* @should return true if parameter is equal to low
* @should return false if parameter is lower than low
* @should return false if both low and high are null
```

Our oracles state:
==> DoubleRangeTest1Oracle.txt <==
true
==> DoubleRangeTest2Oracle.txt <==
false
==> DoubleRangeTest3Oracle.txt <==
false
==> DoubleRangeTest4Oracle.txt <==
false

==> DoubleRangeTest5Oracle.txt <==
true

The script compares the oracle with the resulting output of the function being tested. In these five cases that would be the boolean from the contains() method. A Success means the oracle matched the boolean in our oracle, a Failure would mean the booleans didn't match. This will be expanded to test other methods with oracles likely being something other than a boolean for some variety in our tests.

How to Test

Pull the code from Github into an empty directory. From the home directory *TestAutomation/scripts*, the command './runAllTests.sh' will initialize the test. From that point on, the test will be fully automated, from preparation to completion. The conclusion of the test will result in the test report being opened in a web browser. This report will list each test, what is being tested, the input, and if test produced the correct output.

Conclusion

Our framework is detailed and is almost fully implemented. The rest of our development will involve specifying the other 20 test cases needed and implementing them with drivers into our framework, as well as correcting directories. Overall, our progress is good and our outcome looks successful.