Building the Testing Framework

Team Lucky 7

Laura Barber, Venessa Johansen-Barrera, Michael Blackburn

Introduction

We built a testing framework for ReadtheDocs, the open source documentation project. As part of the initial construction for our testing framework we tested two methods in the nginx-smoke-test python file, passing urls from lists to methods and check whether or not they are served by nginx, django, or perl. Additionally we will pass invalid urls to check for false positives.

Our initial attempts to test the method were thwarted by unconventional naming. Our test file, nginx-smoke-test.py, uses dashes in its name, which are invalid for import in python. We were able to create a proxy file that effectively changed the dashes to underscores, allowing the file to be imported for access to its methods. Our current problem, as our framework exists, is that the proxy file uses an absolute path for conversion, and attempts to convert this to a relative path have been unsuccessful. Using the absolute path, however, is functional at this time, and testing was completed in this way.

Description of Framework

We have a script *runAllTests.sh* that will call a python file *test.py.* The python file imports a proxy file (*nginx_smoke_test.py)* for *nginx-smoke-test.py* because the character ‘-’ is considered invalid syntax for a file name in python. *test.py* will then fetch from a list several urls and feed them through the method we wish to test.

For each test case the results will be output into an html file with the test number, the test input, the expected outcome and the actual outcome. In addition, there will be a header showing which method is being tested for the proceeding test cases.

Once the tests have finished running the *output.html* file that is stored in */temp* will be opened in the system’s default browser.

How-to

Navigate to the top level TESTING directory of the project (Lucky7_TESTING) in the terminal. In order to run the tests type in *./scripts/runAllTests.sh.* The tests should then run automatically and the results will be opened in the default browser.

Tested Items

The items will be tested and the results will be output in the format outlined below:

test number
requirement tested
method tested

test inputs

expected outcome

Test Cases

1

requirement: url is served by nginx and return 200 OK
method: **served_by_nginx**(url)
test input: https://pip.readthedocs.org/en/latest/
expected outcome: True

2

requirement: url is served by nginx and return 200 OK
method: **served_by_nginx**(url)
test input: http://docs.fabfile.org/en/latest/
expected outcome: True

3

requirement: url is served by nginx and return 200 OK
method: **served_by_nginx**(url)
test input: http://cs.cofc.edu
expected outcome: False

4

requirement: url is served by django and return 200 OK
method: **served_by_django**(url)
test input: https://readthedocs.org/security/
expected outcome: True

5

requirement: url is served by django and return 200 OK
method: **served_by_django**(url)
test input: http://www.microsoft.com
expected outcome: False

6

requirement: url is served by django and return 200 OK
method: **served_by_django**(url)
test input: https://cs.cofc.edu
expected outcome: False

Chapter 3

Verifying the Test Cases

In order to verify the server headers of the tested urls outside of the methods we're testing and determine our oracle, we will use https://www.whatismyip.com/server-headers-check/.

We will then compare the actual results returned by the tested method to the expected results, and through this comparison determine the success or failure of each test.

Moving Forward

Going forward, our objective will be to convert the path from absolute to relative, and to include other methods for further testing. We will also want to introduce more control for the tests, i.e. having the user input what method they would like to test via terminal and returning the appropriate results.