**- Framework{Compatible for ubuntu and windows} -**

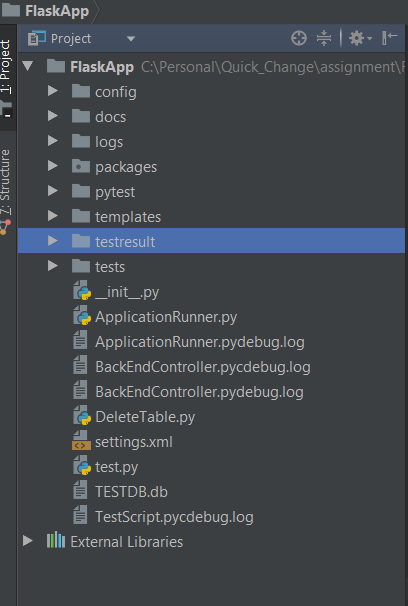
[Pre-requisite]

1. Install python 2.7 (if python3.6.5 is used,the code will work in virtual environment).

Python path is set to /usr/bin/python

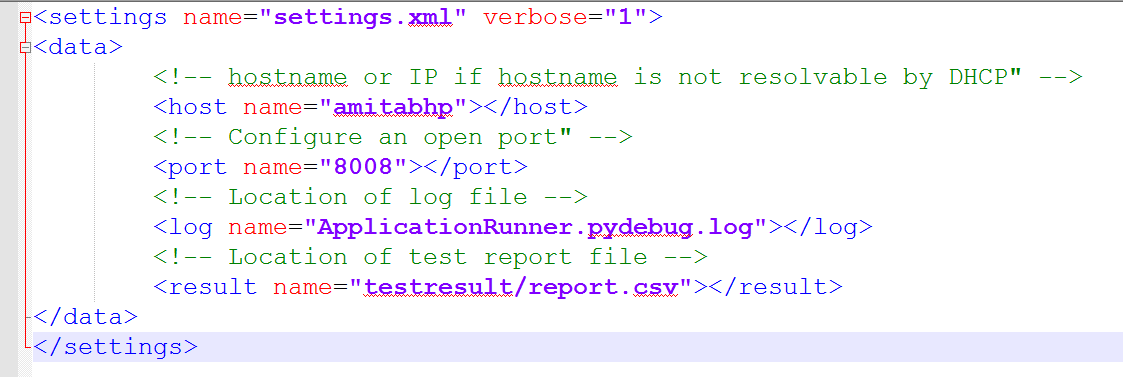
1. sudo pip install –upgrade pip
2. sudo pip install flask
3. sudo apt-get install sqlite3 libsqlite3-dev
4. sudo pip install –upgrade pip
5. sudo pip install pytest
6. sudo pip install selenium
7. add chromedriver to the pytest path.
8. chmod +x <script>
9. Update site-packages if anything is missing.
10. tar -xvf FlaskApp.tar

Below is hierarchy maintained under FlaskApp directory, ApplicationRunner.py is the driver script under this directory; packages (all reusable library modules/functions), Config(\*.xml for tests),templates(\*.html files for web view),tests(python test scripts),pytest, logs(script logs) are placed as part of the framework architecture.



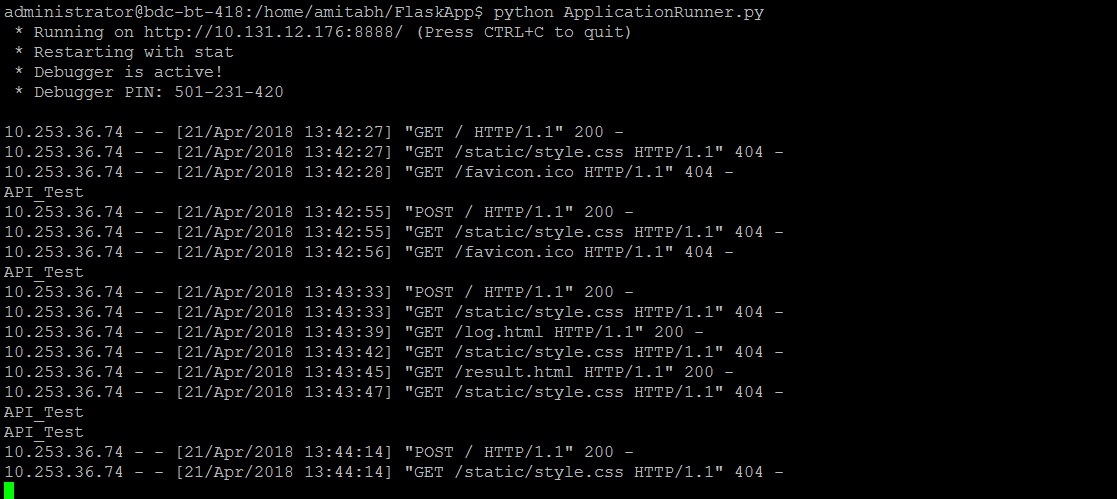
Instructions:

1. Update the hostname and port in the settings.xml file.



1. Start the web server

$ python ApplicationRunner.py



1. Access the webpage

http://<hostaname>:<port>/

1. Open Logs

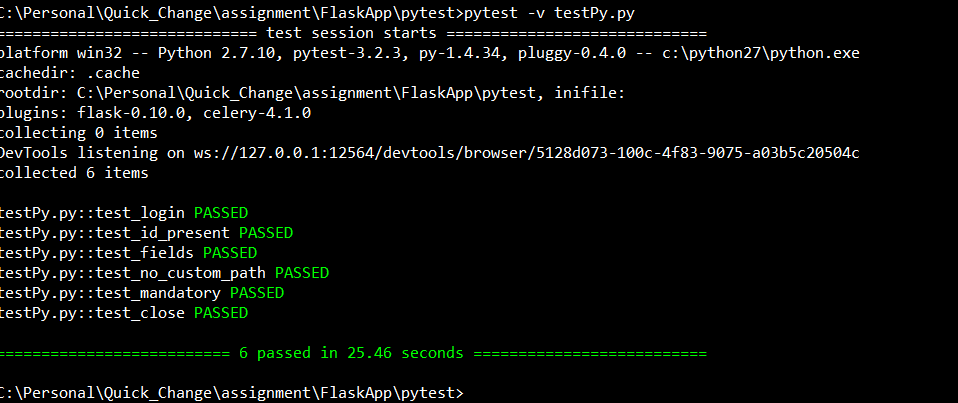
http:// <hostname>:<port>/log.html

1. Open Test Result

http://<hostaname>:<port>/result.html

1. Pytest framework for automating the tests. [Pre-requisite: selenium Webdriver]. I prefer to execute python in a separate machine.

pytest -v testPy.py



**Framework Design…..**

1. MVC architecture is used for the development approach.
2. Database is used to store/retrieve data time to time.
3. Python Flask is used as a web hosting server.
4. WSGI is used as gateway to all post/get requests.
5. Ninja templates are used for rendering UI templates.
6. packages, templates, scripts, config files and driver/database scripts are created to make the system reusable and extensible.
7. \*.xml files are created to provide settings related stuff.
8. Easily refactored for future enhancement.

**RCA (Root Cause Analysis) of memory leak:**

Sometimes the memory leak problem happens in linux systems and works fine in windows. I will follow these steps to do the RCA.

1. The command line utilities **top** or **htop** should be the first stop for any CPU or memory load investigation
2. Add logging statements to the code to measure memory usage over time, or at critical junctures of a long-running process. This can help isolate the critical section of the code that’s causing the memory issue.

$ python

Python 2.7.12 (default, Dec 4 2017, 14:50:18)

[GCC 5.4.0 20160609] on linux2

Type "help", "copyright", "credits" or "license" for more information.

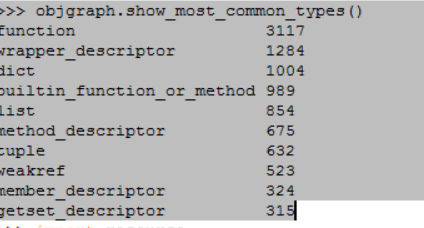
>>> import resource

>>> print 'Memory usage: %s (kb)' % resource.getrusage(resource.RUSAGE\_SELF).ru\_maxrss

Memory usage: 7108 (kb)

>>>

1. Query for the objects currently in memory. & find out the process id of the memory hogging python process.



1. sudo apt-get install libc6-dev libc6-dbg python-gi libglib2.0-0-dbg python-ply

sudo gdb -p <pid of the process>

>generate-core-file

# this will save a .core file, which you can then examine in gdb

sudo gdb python myfile.core -x ~/gdb-heap-commands