Python+Selenium environment Setup instructions

For Windows Users

1. Install python + selenium-webdriver.
   a. Download active-python from here: http://www.activestate.com/activepython/downloads, and then install it in your computer.
   b. In cmd, enter command: pip install selenium.
2. Install webdriver for all the browsers (I took the path examples of my own computer, you should refer to it and install them in the correct path on your own computer)：
   Chrome：Install chromedriver.exe[I can provide you with it if you couldn't download it from internet] under C:\Program Files (x86)\Google\Chrome\Application
   IE：Install IEdriver following instructions in http://code.google.com/p/selenium/wiki/InternetExplorerDriver under C:\Program Files (x86)\Internet Explorer.
   Safari：Install selenium-server-standalone-2.39.0.jar[I can provide you with it if you couldn't download it from internet] under C:\Program Files (x86)\Safari
3. Edit system variables.
   Right-click on Computer, choose Properties > Advanced system settings > Environment Variables > System variables.
   Find the value 'PATH', click on Edit，add the path of the browers' webdrivers after the existing value of PATH, separate them with ';', for example:
   C:\Python27\;C:\Python27\Scripts;%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Google\Chrome\Application;C:\Program Files (x86)\Internet Explorer;C:\Program Files (x86)\Safari
4. As for Safari，we need to add a new system variable.
   Right-click on Computer, choose Properties > Advanced system settings > Environment Variables > System variables.
   Click on New，In Variable name field, enter: SELENIUM_SERVER_JAR, in Variable value field, enter: C:\Program Files (x86)\Safari\selenium-server-standalone-2.39.0.jar。
5. As for IE, you need to add http://helpx.adobe.com as the Trusted site:
   In IE, choose Tools > Internet Options > Security > Trusted sites > Sites. Enter http://helpx.adobe.com in Add this website to the zone field, and click Add.
6. If you don't have java environment in your system, you need to install java and jdk to your system.
7. Put the py file and the config in the same folder, for example, put them here:
   D:\python

For Mac Users

Install Python bindings for Selenium from: https://pypi.python.org/pypi/selenium

ChromeDriver can be download from: https://code.google.com/p/chromedriver/downloads/list

Here is a instruction how to install ChromeDriver in Mac: http://damien.co/resources/how-to-install-chromedriver-mac-os-x-selenium-python-7406

Test Automation Specification

1. Unzip PythonCode folder
2. The PythonCode folder contains cqstart.py, which is the main method executes all test cases under cqtestcases folder
3. When the test is done, a test report will be generated by using HTMLTestRunner
4. Under cqtestcases folder, there is a subfolder called cqpkg, it contains the _init_.py file; All initializations of PythonCode are setup here in this file, including what webdriver to use (Chrome,IE,Safari or Firefox), Test URL, login username, password and some predefined functions.
5. When the test start, a new browser instance will open. The test cases under cqtestcases will execute one by one according to the alphabetic order
6. Once the test is finished, all the error cases during the test will be collected and running in a new page. (This function is not complete yet)
7. After all test cases have been tested, test report will generated under PythonCode folder called "CQAutotestReport_(browser name).html", the test report and cmd window will show the published page URL as well as test results.
8. In test report, the detail information of error case will provide a screenshot of the page where the error occurs. User just need to click the "Error Screenshot", it will link to the image

Command-line Instructions

In cmd change directory to the location where you unzip your PythonCode folder, then enter: python cqstart.py or just go to the file directory and double-click the cqstart.py file

This command will start the automation test of all components and templates we have implemented in both Chrome and Firefox. However if you want to test just some specific test cases in a specific browser, then you could use the following command:

-b browsername: e.g. –b Chrome

this indicate the test is running in Chrome only

Valid Brwosername input: Chrome, Firefox, IE, Safari

-t templatename: e.g. –t helppage

this indicate the test is running help page template only

Valid templatename input: helppage, helpcenter, topics

-c casename: e.g. –c text

this indicate the test is focus on text component only

Valid casename input:

agenttext	codeblock	definition	flashgeneric	flashvideo	fromanexpertarticle
note	heading	image	imageandtext	minitoc	fromanexpertvideo
procedure	reference	relatedlink	remark	table	text
tip	accordion	chl	divider	multicolumn	dropdownmenu
tabnavigation	tabcontent	chlrawhtml	wematomfeed	wembulletin	wemrssfeed
iframe					

-s servername: e.g. –s qa

this means the test is running in qa server

Valid servername input: qa

qa: http://learn-author-qa01.corp.adobe.com:4502/siteadmin#/content/help/en/qe/test-docs/sinotypeselenium

If '–s' command is missing, the default server will be used, the default server is:

http://learn-author-dev01.corp.adobe.com:4502/siteadmin#/content/help/en/qe/test-docs/sinotypeselenium

Template Coverage

So far, we have covered following template:

help page

In help page template we have implement following components:

agent text	code block	definition	flash generic	flash video	from an expert article
note	heading	image	image and text	mini toc	from an expert video
procedure	reference	related link	remark	table	text
tip	accordion	chl container	divider	multicolumn	dropdown menu
tab navigation	tab content	chl raw html	wem atom feed	wem bulletin	wem rss feed
iframe					

help center

All components

content type container

Covered by creating folder

more help manual

All components

more help

All components

top topics

All components

Known issues

There is an issue with the automation tool which we don't know the exact reason for it. We there are too many components created in one page using automation tool, some components might failed, but if you run the failed components in a separate page, it will work fine. Our solution to this is to limit number of component in each page, if the component still not working properly, the error handling function will collect these components and run them in a new created page. A separate test report will be generated by running components that didn't work properly.