selenate Documentation

Release 0.2.0

William Mak

June 07, 2014

Contents

1 Installing **3**

2 Example **5**

3 Documentation **7**

3.1 Installation 7
3.2 Browser 7
3.3 Selenium Server 7
3.4 Selenate 7
3.5 Selenate Objects 8

Author William Mak

Selenate is a python wrapper around the Selenium Python Client Driver. It's main goal is to simplify the management of selenium server, and as well provide easy to use functions.

Currently Selenate is in version 0.3 and has much of the preliminary ground work complete and should be sufficiently usable, but is still missing neccesary functionality.

CHAPTER 1

Installing

```
pip install -U selenate
```

CHAPTER 2

Example

```
from selenate import Selenate
browser = Selenate() # Start the selenium server, and launch firefox
browser.get("http://www.google.com") # Tell firefox to goto google.com
browser.type_to("#gbqfq", "selenate\n") # Locate the css locator, and type
browser.quit() # close the browser, and end the selenium server
```

CHAPTER 3

Documentation

3.1 Installation

Selenate has dependencies on selenium, so at this point you should have installed selenate which in turn would have installed selenium. But if you haven't yet done that the command to install selenate is:

```
pip install -U selenate
```

and if for whatever reason this doesn't install selenium as well that command similarily follows as

```
pip install -U selenium
```

3.2 Browser

Currently selenate only supports firefox so please ensure that you have the most recent version of the browser installed.

3.3 Selenium Server

Selenium requires there to be a Selenium Server running in order to interact with a browser. The server used to be hosted at google code, but has since changed and is now found at googleapis.com. So to download the server navigate there, find the newest version which at the time of writing is 2.41 and download the file known as selenium-server-standalone-2.41.0.jar. Selenate includes automatic management of this server for you if you download this file under the name *selenium-server.jar* and store it in the same directory as your project files.

Now normally you would have to start the selenium server with something like:

```
java -jar selenium-server.jar
```

and you should try running this regardless to check that everything is working. Though for future runs Selenate will handle starting and stoppind this whenever you run a script, this is done in the declaration of the Selenate object

```
browser = Selenate()
```

3.4 Selenate

It's very easy to start a browser instance. First import selenate:

```
from selenate import selenate
```

Then start your browser:

```
browser = selenate()
```

In the background Selenate will check whether or not you have a selenium server running, and start one if you haven't. Then it will create a fresh session of firefox.

3.5 Selenate Objects

A selenate object is the current browser instance and contains everything required to begin automation.

Constructor:

class Selenate.(host, server) The host and server variables are assumed to be strings and neither are required variables..

- the host value defaults to "127.0.0.1" and can be changed to a proxy if needed.
- The server value defaults to "./selenium-server.jar" and should be changed to wherever the selenium server file is located or False if this is being handled elsewhere

Class methods:

classmethod *Selenate*. get (url) url is a strong which is a valid url for the browser to go to.

classmethod *Selenate*. find_element_by_locator (locator) locator is a string in the format "type=locator" where type is one of the following: 'css', 'class' or 'id'. locator may also just be a css identifier. This function will return an object of the Selenate Element class. The use of this class is explained in the Selenate Element Documentation For example:

```
browser = Selenate()
browser.get("http://www.github.com/wmak/selenate")
icon = browser.find_element_by_locator("css=.mega-octicon")
icon = browser.find_element_by_locator(".mega-octicon")
```

classmethod *Selenate*. click (locator) *locator* should be formatted exactly as seen from **find_element_by_locator** this will cause Selenate to click the element described by *locator*.

classmethod *Selenate*. type_to (locator, text) *locator* should be formatted exactly as seen from **find_element_by_locator** *text* is a string. This will cause selenate to enter *text* into the element described by *locator*

classmethod *Selenate*. quit () Closes the Selenate browser, and if Selenate was in charge of the selenium server kills that as well.