
Selenium configurator Documentation

Release 0.1.0

Pierre Verkest

June 20, 2016

1	Resources	3
2	intended persons	5
3	Quick overview	7
4	Known projects using this library	9
5	Contents	11
5.1	Installation	11
5.2	Reference documentation	11
5.3	How to contribute	14
5.4	API Documentation	14
	Python Module Index	19

This library provide an helper api for selenium launchers. it prepares multiple selenium web drivers from a configuration file. This aims to help launchers to run 1 task over multiple browsers through various selenium web drivers (local, grid, cloud provider...).

Resources

- [Documentation](#)
- [Issue Tracker](#)
- [Code](#)

intended persons

- Developers writing a configuration file which want to read a configuration format reference
- Contributors or developer that want to extend nor add new features
- Developer using directly this library to reuse this configuration format and wants to launch selenium test by itself.

Quick overview

This is what looks like a selenium configuration file to prepare 5 webdrivers (2 local Firefox with different configs / 1 local Chrome / 1 chrome on selenium Grid / 1 firefox on selenium Grid):

```
drivers:
- class: selenium_configurator.drivers.local.Chrome
  capabilities:
    cap1: Chrome_capability1
- class: selenium_configurator.drivers.local.Firefox
- class: selenium_configurator.drivers.local.Firefox
  capabilities:
    cap1: Firefox_capability1
- class: selenium_configurator.drivers.remote.Grid
  command_executor: http://grid.example.com:4444/wd/hub
  capabilities:
    cap1: grid_capability1
    cap2: grid_capability2
  request_drivers:
    - browserName: chrome
      platform: LINUX
      version:
        cap2: chrome_grid_capability2
    - browserName: firefox
      platform: LINUX
      version: "3.4"
      cap1: firefox_grid_capability1
global_capabilities:
  cap1: global_capability1
  cap2: global_capability2
  cap3: global_capability3
```

Known projects using this library

- [nose-selenium-multi-browser](#): is a nose plugin that duplicate a testCase as many times as there are configured browsers then those tests can be launched in parallels.
- [selenium-odoo-qunit](#): This is a CI launcher to run Odoo Qunit tests on multiple browsers at once using selenium.

Contents

5.1 Installation

5.1.1 Requirements

selenium-configurator required:

- **python 2.7**, I expect to support in the future python ≥ 3.4 .
- **selenium**, official selenium python binding
- **pyyaml**, yaml parser (which can parse json as well)

5.1.2 Install selenium-configurator

To install **selenium-configurator** from pypi in a virtualenv:

```
$ virtualenv sandbox
$ sandbox/bin/pip install selenium-configurator
```

To install from source:

```
$ virtualenv sandbox
$ git clone git@anybox/selenium-configurator.git
$ cd selenium-configurator
$ ../sandbox/bin/python setyp.py
```

5.2 Reference documentation

5.2.1 Configuration

File format

The configuration file can be a **Json** file or a **Yaml** file.

Structure documentation

Json structure:

```
{
  "drivers": [
    {
      "class": "path.to.the.config.driver.class"
    }, ...
  ],
  "global_capabilities": {
  }
}
```

Yaml structure:

```
drivers:
  - class: path.to.the.config.driver.class
  - ...

global_capabilities:
  ...
```

- **drivers** required: Array of *config driver*
- **global_capabilities** optional: Capabilities shared with all drivers, can be overload on each driver definition

Note: In the following documentation we are going to display only Json format

Configuration driver

Each selenium webdriver MUST have a *Configuration driver class* to later start the browser:

```
{
  "class": path.to.the.config.driver.class,
  "capabilities": {
    "foo": "bar",
  },
}
```

- **class** required: This is the absolute python import to the python configuration driver class
- **capabilities** optional: selenium capabilities to use for this webdriver overwrite `global_capabilities`

Warning: Depending the configuration driver class used, the driver configuration format MAY differ.

We can distinguish 2 kind of webdriver:

- **local:** using to launch browser on the same computer where the code is running
- **remote:** to run browser on other computers.

In the later case we can call it directly or using a service (Selenium Grid, Sauce, ...) that give us a single entry point to communicate with. In this case you will get an extra parameter to ask: is there / where is the browser with those capabilities `request_drivers` those services will give you an available browser to run your test on.

This library provide a list of configuration driver class to map common selenium webdriver:

- **Local drivers: Require selenium drivers installed for the given browser on**

the computer you are running this lib

- `selenium_configurator.drivers.local.Firefox`
- `selenium_configurator.drivers.local.Chrome`
- `selenium_configurator.drivers.local.Opera`
- `selenium_configurator.drivers.local.Safari`
- `selenium_configurator.drivers.local.Ie`
- `selenium_configurator.drivers.local.Phantomjs`

- **Remote drivers:**

- `selenium_configurator.drivers.local.Grid`

Extending configuration file

Not implemented yet! I would love to add the capability to a configuration file to extend an other configuration files to allow more flexibility over plateform constraint.

Confidential information

When using Grid service like Sauce / ... you may want to not store private token on this config file! I expect to manage this use cas when I'll need it the first time

5.2.2 Coding

Use this library

This is a simple example how to use this library:

```
from selenium_configurator.conf.configurator import Configurator

selenium_conf = Configurator.from_file('/path/to/selenium.json')
drivers = selenium_conf.get_drivers() # list of config drivers class

# to visit python.org website on each browser define on selenium.json
for driver in drivers:
    driver.selenium.get('https://www.python.org/')
    driver.quit()
```

For further information please have a look to the [API documentation](#)

Configuration driver class

Extending existing driver

5.3 How to contribute

5.3.1 Installation

Install **selenium-configurator** from source:

```
$ virtualenv sandbox
$ git clone git@anybox/selenium-configurator.git
$ cd selenium-configurator
$ ../sandbox/bin/python setup.py develop
```

5.3.2 Running test

To run unittest you will need an internet access, firefox webdriver and a grid to run it locally:

```
$ nosetests -s -v selenium_configurator/tests/
```

5.3.3 Setup local webdriver

- [chrome/chromium](#)
- [firefox](#)

5.3.4 Setup grid and nodes

You can easily setup a Grid using selenium docker container.

- [Selenium images on docker hub](#)
- [Docker files](#)

5.3.5 Code style

PEP8

5.4 API Documentation

5.4.1 Configurator

class `selenium_configurator.configurator.Configurator` (*config*)

This Configurator class is the entry point to prepare a selenium web driver list from a configuration string, file or object.

A short example to launch on a local Firefox would be:

```

from selenium_configurator.configurator import Configurator

configurator = Configurator(
    {
        'drivers': [
            {
                'class': 'selenium_configurator.drivers.local.Firefox',
            }
        ],
    }
)
drivers = configuration.get_drivers()
for driver in drivers:
    driver.selenium.get('https://anybox.fr/')
    driver.quit()

```

classmethod from_file (*path*)

Create a Configurator from a .yaml or .json file

Parameters *path* – to the config file

Returns An instance of Configurator

classmethod from_string (*yaml_conf*)

Create a Configurator from a string that expected to be a yaml or json format.

Parameters *yaml_conf* – A string

Returns An instance of Configurator

get_drivers ()

Get the list of driver instance from the configuration description provide to that configurator class

Returns A list of drivers instances that contains selenium instance ready to start,

5.4.2 Driver

class selenium_configurator.driver.**Driver** (*conf*, *global_capabilities=None*, *name=None*)

Abstract class to define and ensure to expose uniform public API

We mainly recommend to use *get_web_drivers* class method to instantiate Drivers. For instance remote web-drivers that use service like selenium Grid will return multiple instance of Driver..

classmethod *get_web_drivers* (*conf*, *global_capabilities=None*)

Class method to prepare Driver(s) instance according current Driver class and given settings.

Parameters

- **conf** – a dict that contains the Driver configuration
- **global_capabilities** – Capabilities shared over Drivers configuration

Returns a list of Driver

global_capabilities = None

Capabilities shared between drivers

name

Returns the technical name that describe the related webdriver

quit()

Prefer this way to `quit()` a selenium to properly re-open the browser in case it was closed

selenium

Get the instance of webdriver, it starts the browser if the webdriver is not yet instantiated

Returns a 'selenium instance <<http://selenium-python.readthedocs.org/>

api.html#module-selenium.webdriver.remote.webdriver>'

class `selenium_configurator.driver.DriverFactory`

This class parse the config dict to generate the list of Driver

5.4.3 Local

Local driver to launch browser on the current computer.

While instantiate a webdriver you can request for different [capabilities](#), to change available behavior of the browser.

class `selenium_configurator.drivers.local.Chrome` (*conf*, *global_capabilities=None*,
name=None)

Local Chrom/Chromium webdriver

Configuration settings:

```
{
    'class': 'selenium_configurator.drivers.local.Chrome',
    'capabilities': {
        ...
    }
}
```

[Specific capabilities](#)

class `selenium_configurator.drivers.local.Firefox` (*conf*, *global_capabilities=None*,
name=None)

Local Firefox webdriver

Configuration settings:

```
{
    'class': 'selenium_configurator.drivers.local.Firefox',
    'capabilities': {
        ...
    }
}
```

[Specific capabilities](#)

class `selenium_configurator.drivers.local.Ie` (*conf*, *global_capabilities=None*, *name=None*)

Local Internet Explorer webdriver

Configuration settings:

```
{
    'class': 'selenium_configurator.drivers.local.Ie',
    'capabilities': {
        ...
    }
}
```

[Specific capabilities](#)

```
class selenium_configurator.drivers.local.Opera(conf, global_capabilities=None,
                                              name=None)
```

Local Opera webdriver

Configuration settings:

```
{
    'class': 'selenium_configurator.drivers.local.Opera',
    'capabilities': {
        ...
    }
}
```

```
class selenium_configurator.drivers.local.Phantomjs(conf, global_capabilities=None,
                                                    name=None)
```

Local phantomjs webdriver

Configuration settings:

```
{
    'class': 'selenium_configurator.drivers.local.Phantomjs',
    'capabilities': {
        ...
    }
}
```

```
class selenium_configurator.drivers.local.Safari(conf, global_capabilities=None,
                                                name=None)
```

Local Safari webdriver

Configuration settings:

```
{
    'class': 'selenium_configurator.drivers.local.Safari',
    'capabilities': {
        ...
    }
}
```

Specific capabilities

5.4.4 Remote

```
class selenium_configurator.drivers.remote.Grid(grid_conf, desired_capabilities,
                                                global_capabilities=None, name=None)
```

Grid Driver class to use over browsers over a Selenium Grid

It give you the availability to request multiple browsers against one Grid server.

Configuration settings to get 1 Chrome and 1 Firefox over a Grid service:

```
conf = {
    'class': 'selenium_configurator.drivers.remote.Grid',
    'capabilities': {
        # Capabilities shared with request browser and overload
        # general capabilities
    },
    "request_drivers": [
        {
            "browserName": "firefox",
            "platform": "LINUX",
```

```
        "version": "",
        # other capabilities
    },
    {
        "browserName": "chrome",
        "platform": "LINUX",
        "version": "",
    },
]
}
from selenium_configurator.driver.remote import Grid
driver_list = Grid.get_web_drivers(conf)
```

Available request drivers capabilities

classmethod `get_web_drivers` (*conf*, *global_capabilities=None*)

Prepare 1 selenium driver instance per request browsers

Parameters

- **conf** –
- **global_capabilities** –

Returns

S

`selenium_configurator.configurator`, [14](#)
`selenium_configurator.driver`, [15](#)
`selenium_configurator.drivers.local`, [16](#)
`selenium_configurator.drivers.remote`,
[17](#)

C

Chrome (class in selenium_configurator.drivers.local), 16
Configurator (class in selenium_configurator.configurator), 14

D

Driver (class in selenium_configurator.driver), 15
DriverFactory (class in selenium_configurator.driver), 16

F

Firefox (class in selenium_configurator.drivers.local), 16
from_file() (selenium_configurator.configurator.Configurator
class method), 15
from_string() (selenium_configurator.configurator.Configurator
class method), 15

G

get_drivers() (selenium_configurator.configurator.Configurator
method), 15
get_web_drivers() (selenium_configurator.driver.Driver
class method), 15
get_web_drivers() (selenium_configurator.drivers.remote.Grid
class method), 18
global_capabilities (selenium_configurator.driver.Driver
attribute), 15
Grid (class in selenium_configurator.drivers.remote), 17

I

Ie (class in selenium_configurator.drivers.local), 16

N

name (selenium_configurator.driver.Driver attribute), 15

O

Opera (class in selenium_configurator.drivers.local), 16

P

Phantomjs (class in selenium_configurator.drivers.local),
17

Q

quit() (selenium_configurator.driver.Driver method), 15

S

Safari (class in selenium_configurator.drivers.local), 17
selenium (selenium_configurator.driver.Driver attribute),
16
selenium_configurator.configurator (module), 14
selenium_configurator.driver (module), 15
selenium_configurator.drivers.local (module), 16
selenium_configurator.drivers.remote (module), 17