SeleShot Documentation

Release 0.0.5

Bartosz Alchimowicz

April 19, 2014

Contents

SeleShot allows to get a screen shot of a whole webpage and a part of it.

Contents:

CHAPTER 1

Overview

Sometimes it is required to take a screen shot not of the whole web page but only a part of it. Such a screen shot can contains for example a single button or form.

Seleshot possibilities:

- Support for taking screenshot(s) of elements with a specified id(s)
- Support for taking screenshot(s) of elements with a specified xpath(s)
- Support for taking screenshot(s) of area.
- Support for taking screenshot(s) with highliting any elements

CHAPTER 2

Installation

The following libraries are required:

- python 2.7
- python-setuptools
- python-imaging 1.1.7-4
- selenium (tested with: 2.25.0-py2.7, 2.39.0-py2.7)

```
easy_install selenium
```

- **selenium-server-standalone**
  - Download the latest release
  - Run server from console with the following command:

```
java -jar selenium-server-standalone-2.31.0.jar
```

CHAPTER 3

Usage examples

3.1 Standalone application

```
seleshot.py -u http://www.python.org
seleshot.py -u http://www.python.org -f img.png
```

3.2 In code

The code below shows how to create a simple application which takes a screen shot of a webpage.

```
from seleshot import create

s = create()
s.get_screen(url="http://www.python.org").save(path)
s.close()
```

This code shows how to use cut_element and cut_area functions.

```
from seleshot import create

xpath =  ".//*[@id='mainnav']/ul/li"
id = "submit"
url = 'http://www.python.org'

s = create()
i = s.get_screen(url)

i.cut_element(id = id).save('cut1.png')
i.cut_element(xpath = xpath).save('cut2.png')

i.cut_area(height = 100).save("area1.png")
i.cut_area(200, 300, 250, 350).save('area2.png')
i.cut_area(200, 300, 250, 350).cut_area(60, 60, 50, 50).save("area3.png")

s.close()
```

This code shows how to use draw_dot and draw_frame functions.

```
from seleshot import create

s = create()
```

```
xpath = ".//*[@id='mainnav']/ul/li"
id = "submit"
url = 'http://www.python.org'

i = s.get_screen(url)

i.draw_frame(id = id,padding = 10, color='yellow',size= 5).save('frame1.png')
i.draw_frame(coordinates=(500,500,40,50), color='green').save('frame2.png')

i.cut_area(200, 300, 250, 350).draw_dot(coordinates = (50, 50), padding = 3, color = 'yellow', size =
    coordinates = (60, 20), padding = 4, color = 'red', size = 10).save('dot1.png')

i.draw_dot(id='touchnav-wrapper',padding= 10, size=100).save("dot2.png")
i.draw_dot(id='submit',padding= 1, size=3).save("dot3.png")
s.close()
```

This code shows how to use draw_image function.

```
from seleshot import create

s = create()
url = 'http://www.python.org'

i = s.get_screen(url)

i.cut_element(id = 'submit').save('cut1.png')
i.cut_element(xpath = ".//*[@id='mainnav']/ul/li").save('cut2.png')

i.draw_image(id = 'submit', padding = (0, 0), position = i.Position.OUTSIDE | i.Position.BOTTOM, file
i.draw_image(xpath = ".//*[@id='mainnav']/ul/li", padding = (15, 10), position = i.Position.OUTSIDE
i.draw_image(id = 'touchnav-wrapper', padding = (15, 10), position = i.Position.OUTSIDE | i.Position
i.draw_image(coordinates = (100, 200), padding = (0, 0), position = i.Position.OUTSIDE | i.Position.P

s.close()
```

This code shows how to use draw_zoom and draw_blur functions.

```
from seleshot import create

s = create()
url = 'http://www.python.org'

i = s.get_screen(url)

i.cut_element(id = 'submit').save('cut1.png')
i.cut_element(xpath = ".//*[@id='mainnav']/ul/li").save('cut2.png')

i.draw_blur(id = 'submit').save('blur1.png')
i.draw_blur(xpath = ".//*[@id='mainnav']/ul/li").save('blur2.png')

i.draw_zoom(id = 'submit', padding = (0, 5), position = i.Position.OUTSIDE | i.Position.BOTTOM, zoom
i.draw_zoom(xpath = ".//*[@id='mainnav']/ul/li", padding = (15, 10), position = i.Position.OUTSIDE |
i.draw_zoom(id = 'touchnav-wrapper', padding = (15, 10), position = i.Position.OUTSIDE | i.Position.I

i.draw_zoom(id = 'submit', padding = (0, 5), position = i.Position.OUTSIDE | i.Position.BOTTOM, zoom

s.close()
```

CHAPTER 4

API

ScreenShot(object):

get_screen(self, url = None):
Get specified screen(s)

Parameters **url** (*string*) – web page to capture (including http protocol, None to reuse loaded webpage)

Returns Screen shot

Return type ImageContainer

Raises Exception

close(self):
Close driver

ImageContainer(object):

Container for an image.

Possible positions:

•MIDDLE

•INSIDE

•OUTSIDE

•BORDER

•LEFT

•RIGHT

•TOP

•BOTTOM

Example of usage:

```
position = Position.OUTSIDE | Position.LEFT
```

cut_element(self, id = None, xpath = None):
Cut one element by id or xpath. After this operation you cannot cut more elements.

Parameters

- **id** (*string*) – id of a given element

- **xpath** (*string*) – xpath of a given element

Returns ImageContainer

Return type ImageContainer

Raises RuntimeError, ValueError

cut_area(self, x = 0, y = 0, height = None, width = None):

Cut area from a given point to a given size (in px)

Parameters

- **x** (*integer*) – x coordinate for a point
- **y** (*integer*) – y coordinate for a point
- **height** (*integer or None*) – height of an area
- **width** (*integer or None*) – width of an area

Returns ImageContainer

Return type ImageContainer

draw_dot(self, id = None, xpath = None, coordinates = None, padding = 0, color = None,

For id and xpath: Draw a red dot on a given position of a given element.

For coordinates: Draw a red dot in a given point (x, y)

Parameters

- **id** (*string*) – id of a given element
- **xpath** (*string*) – xpath of a given element
- **coordinates** (*tuple of integers (x, y)*) – coordinates = (x, y) - center of a dot
- **position** (*Position enum*) – position of a dot
- **padding** (*tuple of integers (x, y)*) – padding between dot and element
- **color** (*color object or string*) – color of dot
- **size** (*integer*) – size of dot

Returns ImageContainer

Return type ImageContainer

Raises ValueError

draw_frame(self, id = None, xpath = None, coordinates = None, padding = None, color = N

For id and xpath: Draw a frame around a given element

For coordinates: Draw a frame for a given coordinates

Parameters

- **id** (*string*) – id of a given element
- **xpath** (*string*) – xpath of a given element
- **coordinates** (*tuple of integers - (x, y, width, height)*) – coordinates for a frame - coordinates = (x, y, width, height) - middle of a dot
- **padding** (*tuple of integers (x, y)*) – padding between frame and element

- **color** (*color object or string*) – color of a frame (see PIL's documentation)
- **size** (*integer*) – size of frame (thickness)

Returns ImageContainer

Return type ImageContainer

Raises ValueError

draw_image(self, id = None, xpath = None, coordinates = None, position = Position.MIDD

For id and xpath: Draw an image on a given position of a given element.

For coordinates: Draw an image in a given point (x, y)

Parameters

- **id** (*string*) – id of a given element
- **xpath** (*string*) – xpath of a given element
- **coordinates** (*tuple of integers (x, y)*) – coordinates = (x, y) - center of an image
- **position** (*Position enum*) – position of an image
- **padding** (*tuple of integers (x, y)*) – padding between dot and element
- **filename** (*string*) – filename of the image file
- **image** (*Image object*) – reference to Image object

Returns ImageContainer

Return type ImageContainer

Raises ValueError

draw_zoom(self, id = None, xpath = None, coordinates = None, position = Position.MIDDL

For id and xpath: Draw a zoomed image on a given position of a given element.

For coordinates: Draw a zoomed element in a given point (x, y).

Parameters

- **id** (*string*) – id of a given element
- **xpath** (*string*) – xpath of a given element
- **coordinates** (*tuple of integers (x, y)*) – coordinates = (x, y) - center of a zoomed image
- **position** (*Position enum*) – position of a zoomed image
- **padding** (*tuple of integers (x, y)*) – padding between dot and element
- **zoom** (*integer*) – zoom size of an element

Returns ImageContainer

Return type ImageContainer

draw_blur(self, id = None, xpath = None):

Blur whole area of the screenshot except a given element.

Parameters

- **id** (*string*) – id of a given element
- **xpath** (*string*) – xpath of a given element

Returns ImageContainer

Return type ImageContainer

Raises RuntimeError, ValueError

save(self, filename):
Save to a filename

Parameters **filename** (*string*) – name of a file

Returns ImageContainer

Return type ImageContainer

is_cut(self):
If True, then there is possibility to cut an element. If False, then there is not possibility to cut any element.

Returns possibility to cut

Return type boolean