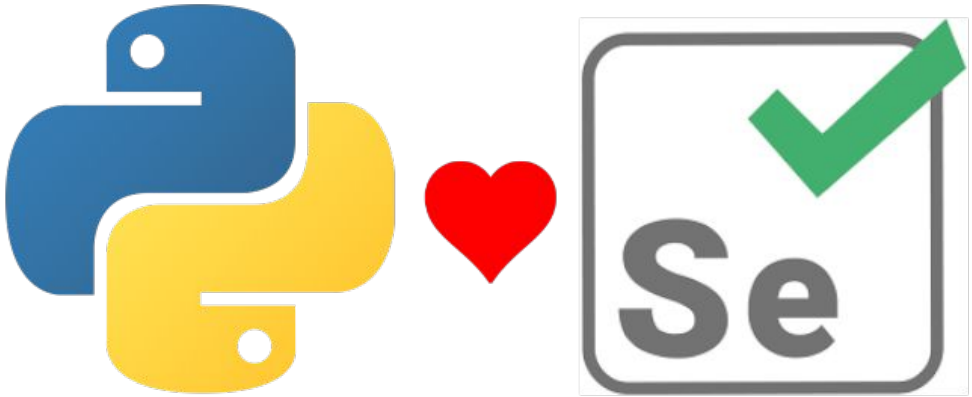


16 - Splinter

Roteiro

- Sobre o Splinter
- Instalação
- Browser / WebDriver
- Navegação
- Interagindo com elementos
  - Encontrar
  - Mouse
  - Teclado
- Matchers
- Frames / Janelas
- Nem tudo são flores

picpay.me/dunossauro | apoia.se/livedepython

picpay.me/dunossauro | apoia.se/livedepython

```
dunossauro at babbage in ~/git/apoiase on master*
$ python apoiadores.py
ADEMAR Peixoto       ALBERTO TAVARES     Acássio Araújo      Adão Oliveira       Alan Soder          Alex Lima
Alexandre Harano     Alexandre Santos    Alexandre Sá        Alexandre Tsuno     Alexandre Villares  Alynne Ferreira
Alysson Oliveira     Amanda Magalhães    Amaziles Carvalho   Andre Rodrigues     Antonio Ribeiro     Bernardo Fontes
Bianca Rosa          Bruno Oliveira      Bruno Rocha         CARLOS SANTOS       Caio Nascimento     Carlos Cardoso
Carol Souza          Cleiton Souza       Davi Ramos          Davi Lima           Diego Guimarães     Dilenon Delfino
Douglas Bastos       EDUARDO PAZZI       Edson Braga         Eduardo Nunes       Elias Soares        Emerson Lara
Eugenio Mazzini      Everton Alves       Fabrício Coelho     Fernando Vier       Filipe Cruz         FlavKaze FlavKaze
Franklin Silva       Fábio Serrão        Gabriel Simonetto   Gabriela Santiago   Gabrielly Andrade   Geandreson Costa
Gladson Menezes      Hélio Neto          Isaac Ferreira      Johnny Tardin       Jonatas Oliveira    Jonatas Leon
Jonatas Oliveira     Jones Leite         Jones Leite         Jones Lourenço      Jorge Plautz        José Prado
Jovan Costa          João Lugão          Juan Gutierrez      LEONARDO CRUZ       Lucas Mendes        Lucas Neris
Lucas Polo           Lucas Valino        Luciano Ratamero    Luiz Bruno          Luiz Lima           Maiquel Leonel
Marcela Campos       Marcello Benigno    Mateus Braga        Melisa Campagnaro   Octavio Sydow       Otavio Carneiro
Patrick Gomes        Paulo Tadei         Rafael Peixoto      Rafael Dias         Reinaldo Silva      Renan Gomes
Renan Moura          Renne Rocha         Ricardo Schalch     Rodrigo Ferreira    Rodrigo Vaccari     Régis Tomkiel
Thiago Araujo        Thiago Bueno        Tiago Cordeiro      Tyrone Damasceno    Valdir Junior       Vicente Marcal
Victor Geraldo       Vinícius Bastos     Vinícius Ferreira   Vítor Zanoni        William Oliveira    Willian Lopes
Willian Lopes        Willian Rosa        Wladimir Falcão     Falta você          Falta você          Falta você
```

Sample code

```
from splinter import Browser

with Browser() as browser:
    # Visit URL
    url = "http://www.google.com"
    browser.visit(url)
    browser.fill('q', 'splinter - python acceptance testing for web applications')
    # Find and click the 'search' button
    button = browser.find_by_name('btnG')
    # Interact with elements
    button.click()
    if browser.is_text_present('splinter.readthedocs.io'):
        print("Yes, the official website was found!")
    else:
        print("No, it wasn't found... We need to improve our SEO techniques")
```

https://splinter.readthedocs.io

O que é o splinter?

https://github.com/cobrateam/splinter

Selenium em camadas

browser.open('url')

Selenium em camadas

browser.visit('url')

SPLINTER

SELENIUM

DRIVER A | DRIVER B | DRIVER C

BROWSER A | BROWSER B | BROWSER C | BROWSER D

Instalação

pip install splinter

Browser / WebDriver

Manipulando o webdriver

O splinter fornece uma API **muito simples** para manipular o browser e facilita as chamadas de import

```
from splinter import Browser

b = Browser()
b = Browser(headless=True)
b = Browser(incognito=True)
```

Manipulando o webdriver

O splinter fornece uma API **muito simples** para manipular o browser e facilita as chamadas de import

```
from splinter import Browser

b = Browser()
b = Browser(headless=True)
b = Browser(incognito=True)
```

firefox por padrão

Manipulando o webdriver

Para alternar entre os browsers não é necessário mudar a forma dos imports, o que facilita um bocado

```
from splinter import Browser


b = Browser('firefox')
b = Browser('chrome')
```

Manipulando o webdriver

Webdrivers remotos respeitam a mesma API

```
from splinter import Browser

b = Browser(
    driver_name='remote',
    browser='firefox',
    command_executor='http://127.0.0.1:4444/wd/hub',
    desired_capabilities={'platform': 'LINUX'},
)
```

Manipulando o webdriver

Webdrivers remotos respeitam a mesma API

```
from splinter import Browser

b = Browser(
    driver_name='remote',
    browser='firefox',
    command_executor='http://127.0.0.1:4444/wd/hub',
    desired_capabilities={'platform': 'LINUX'},
)
```

DEFAULT

Possíveis problemas

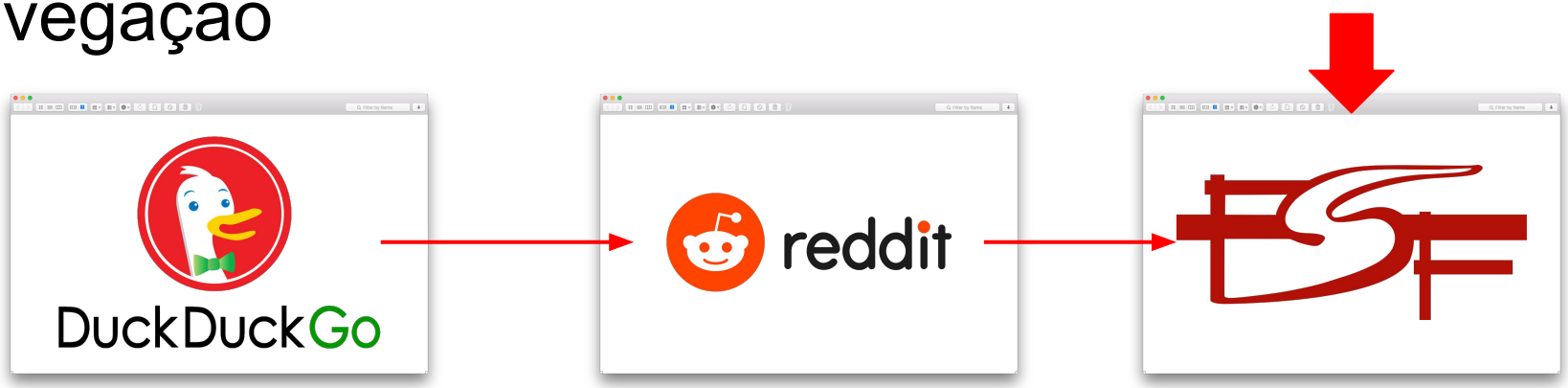
Caso você tenha problemas com a importação e o uso de paths para os executáveis

```
from splinter import Browser

executable_path = {'executable_path': '</path/to/chrome>'}
browser = Browser('chrome', **executable_path)
```

Navegação

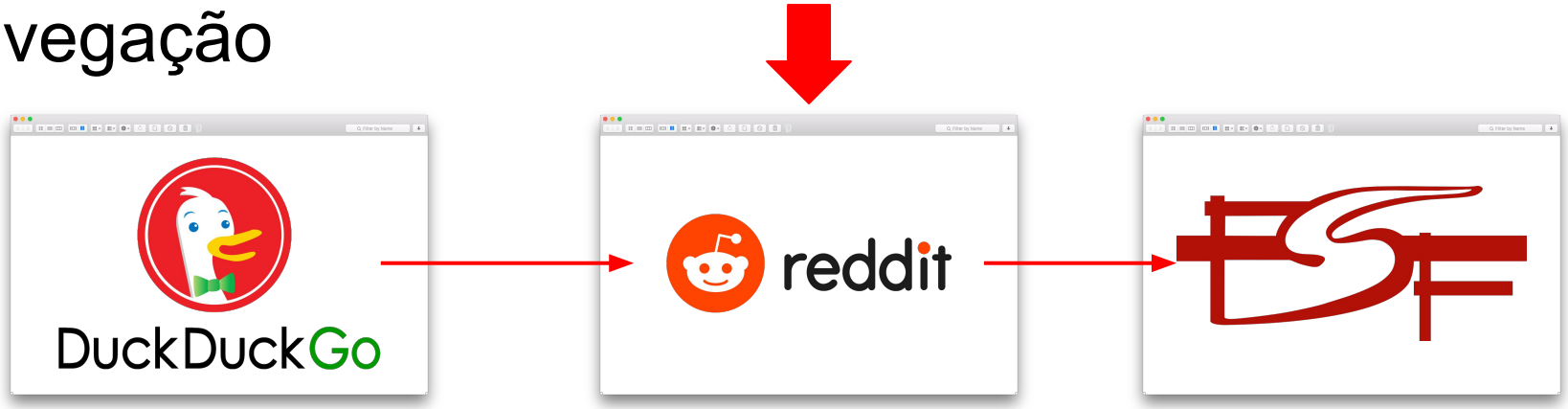
Navegação



Histórico	
pos	**url**
2	http://ddg.gg
1	http://redit.com
0	https://www.fsf.org/

```
>>> browser.visit('http://fsf.org')
```

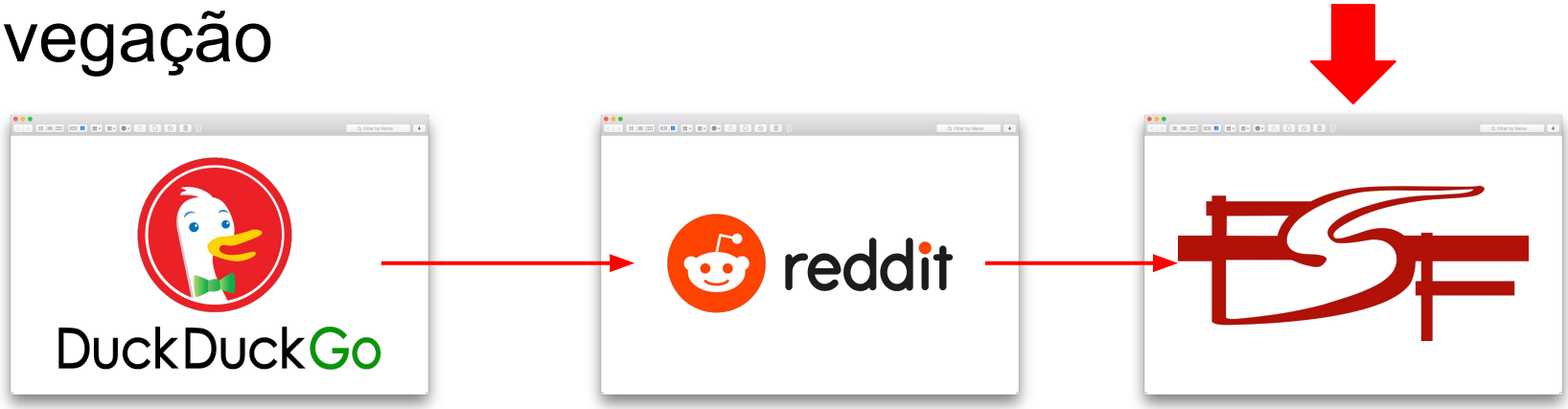
Navegação



Histórico	
pos	**url**
2	http://ddg.gg
1	http://redit.com
0	https://www.fsf.org/

```
>>> browser.visit('http://fsf.org')
>>> browser.back()
```

Navegação



Histórico	
pos	**url**
2	http://ddg.gg
1	http://reddit.com
0	https://www.fsf.org/

```
>>> browser.visit('http://fsf.org')
>>> browser.back()
>>> browser.forward()
```

Navegando com Splinter

- get() -> visit()
- back()
- forward()
- current_url -> url
- title
- page_source -> html

Interagindo com os elementos

Interagindo com elementos

O splinter nos permite fazer as mesmas buscas e interações que o selenium nos permite, porém com uma API mais simplificada. Por exemplo, procurar “live de python” no google

```
from splinter import Browser

b = Browser()
b.visit('http://google.com')
b.find_by_css('[name="q"]').type('Live de python')
b.find_by_name('btnK').click()
```

Interagindo com elementos

O splinter nos permite fazer as mesmas buscas e interações que o selenium nos permite, porém com uma API mais simplificada. Por exemplo, procurar “live de python” no google

```
from splinter import Browser

b = Browser()
b.visit('http://google.com')
b.find_by_css('[name="q"]').type('Live de python')
b.find_by_name('btnK').click()
```

Codando um tikinho

https://selenium.dunossauro.live/aula_07.html

Como estamos fazendo?

```
from splinter import Browser

b = Browser()

b.visit('https://selenium.dunossauro.live/aula_07.html')

b.find_by_id('nome').type('eduardo')
b.find_by_id('email').type('eduardo@duno.live')
b.find_by_id('senha').type('1234')
```

Fill e Type

Type usa a **low level API** do selenium. Onde faz **keyUp** e **keyDown**. Tem lugares onde precisamos usar type.

Porém, em formulários, é necessário que o **input** tenha um nome (**name**) e nesse caso a opção **fill** pode nos ajudar.

```
>>> b.fill(name', 'text')
```

Codando + um tikinho

https://selenium.dunossauro.live/aula_07.html

Procurando elementos

Selenium	Splinter
find_element_by_id()	find_by_id()
find_elements_by_css_selector()	find_by_css()
find_elements_by_tag_name()	find_by_tag()
find_elements_by_xpath()	find_by_xpath()
find_elements_by_class_name()	...
find_elements_by_link_text()	...
find_elements_by_partial_link_text()	...
find_elements_by_name()	find_by_name()
...	find_by_text()
...	find_by_value()

Procurando Links

Selenium	Splinter
find_elements_by_link_text()	...
find_elements_by_partial_link_text()	...

```
b.links.find_by_text()
b.links.find_by_partial_text()
b.links.find_by_href()
b.links.find_by_partial_href()
```

Procurando Links

Selenium	Splinter
find_elements_by_link_text()	...
find_elements_by_partial_link_text()	...

```
b.find_by_css('selector').find_by_text()
b.find_by_css('selector').find_by_partial_text()
b.find_by_css('selector').find_by_href()
b.find_by_css('selector').find_by_partial_href()
```

Lazy “Finds”

```
elements = b.find_by_css('seletor')

elements.fist
elements.last
elements[1]  # slice
```

Codando + um tikinho

https://selenium.dunossauro.live/aula_07.html

Mouse

Eventos com mouse

O splinter nos ajuda a ter acesso a recursos do **ActionChains** (lowlevel) de maneira bem simples

```
caixa = b.find_by_id('caixa')

caixa.click()
caixa.mouse_over()     # AC
caixa.double_click()   # AC
caixa.right_click()    # AC
caixa.mouse_out()      # AC
```

Eventos com mouse

Temos facilitadores para o click, assim como tínhamos com **fill**.

```
b.click_link_by_href('href')
b.click_link_by_id('id')
b.click_link_by_text('text')
b.click_link_by_partial_text('text')

element = b.find_by_id('id')
element.check()
element.unckeck()
```

Codando + um tikinho

https://selenium.dunossauro.live/caixinha.html

Matchers

https://splinter.readthedocs.io/en/latest/matchers.html

Matchers

Matchers no splinter são um casamento entre duas funcionalidades. Os waits originais do selenium redesenhados e o esquema de assertivas que normalmente fazemos na mão.

Matchers

Por exemplo, se quisermos saber, se um determinado texto está, ou não, na tela.

```
from splinter import Browser

b = Browser()

b.visit('https://selenium.dunossauro.live/aula_09_a.html')

b.is_text_present('texto')
b.is_text_not_present('texto')
```

Matchers

Por exemplo, se quisermos saber, se um determinado texto está, ou não, na tela.

```
from splinter import Browser

b = Browser()

b.visit('https://selenium.dunossauro.live/aula_09_a.html')

b.is_text_present('texto')
b.is_text_not_present('texto')
```

boolean

Matchers

Por exemplo, se quisermos saber, se um determinado elemento está vísivel, ou não.

```
from splinter import Browser

b = Browser()

b.visit('https://selenium.dunossauro.live/aula_09_a.html')

# is_element_present_by_*
# is_element_not_present_by_*

b.is_element_present_by_css('i.dropped')
b.is_element_note_present_by_name('name')
```

Matchers

Por exemplo, se quisermos saber, se um determinado elemento está, ou não, na tela.

```
from splinter import Browser

b = Browser()

b.visit('https://selenium.dunossauro.live/aula_09_a.html')

# is_element_present_by_*
# is_element_not_present_by_*

b.is_element_present_by_css('i.dropped')
b.is_element_note_present_by_name('name')
```

boolean

Matchers

Por exemplo, se quisermos saber, se um determinado elemento está, ou não, na tela.

```
from splinter import Browser

b = Browser()

b.visit('https://selenium.dunossauro.live/aula_09_a.html')

# is_element_present_by_*
# is_element_not_present_by_*

b.is_element_present_by_css('i.dropped')
b.is_element_note_present_by_name('name')
```

*** = seletor**

Matchers

Por exemplo, se quisermos saber, se um determinado elemento está, ou não, na tela. (Somente XPATH e CSS selector)

```
from splinter import Browser

b = Browser()

b.visit('https://selenium.dunossauro.live/aula_09_a.html')

is_element_visible_by_css('seletor')
is_element_visible_by_xpath('//xpath')
```

bollean

Nem tudo são flores

Nem tudo são flores

- Não tem erros “explícitos”
- Não oferece suporte nativo a todos os browsers
  - somente, Chrome e Firefox
  - https://splinter.readthedocs.io/en/latest/contribute/writing-new-drivers.html