

Python na automatização de testes

<https://github.com/elyezer/test-automation-examples>

Sobre mim

- Formado em Engenharia da Computação no Inatel em Agosto de 2011
- Engenheiro de Qualidade Senior na Red Hat desde dezembro de 2013
- Experiência com Python desde 2008, e desde 2013 profissionalmente
- Github: [@elyezer](#)

O que é Python

Python é uma linguagem de programação que permite você trabalhar rapidamente e integrar seus sistemas de forma efetiva.

Ele é poderoso... e rápido; Pode ser usado em conjunto com outras tecnologias; Multi-plataforma; É amigável e fácil de aprender; É Aberto.

Quem usa Python?

- Internet: YouTube, Globo.com, Bitly, Mozilla...
- Computação gráfica: Disney, AutoDesk...
- Desktop: Dropbox, OpenOffice...
- Operações: Red Hat, Google, Fedora...
- Enterprise: IBM, Oracle, OpenERP...
- Games, computação científica, segurança, etc...

Por quê criar testes automatizados

- Garantir a qualidade
- Descobrir regressões facilmente
- Evitar trabalho manual
- Obter resultados rápidos
- Permitir a integração contínua

Ferramentas

- unittests - criação de testes unitários
- DDT (data-driven tests) - multiplicador de testes unitários
- Requests - biblioteca HTTP
- Fabric - utilitário SSH para deploy de aplicações e tarefas de administração de sistema

Ferramentas

- Selenium - framework de testes para aplicações web
- Splinter - Camada de abstração em cima do Selenium
- Servidor de integração contínua. Exemplo: Jenkins

Unittests

```
1 def add(a, b):
2     """Sum a + b"""
3     return a + b
4
5
6 def sub(a, b):
7     """Subtract a - b"""
8     return a - b
9
10
11 def mul(a, b):
12     """Multiply a * b"""
13     return a * b
14
15
16 def div(a, b):
17     """Divide a / b"""
18     return a / b
```

```
1 import unittest
2 import calculator
3
4
5 class CalculatorTestCase(unittest.TestCase):
6     def test_add(self):
7         self.assertEqual(calculator.add(40, 2), 42)
8
9     def test_sub(self):
10        self.assertEqual(calculator.sub(40, 2), 38)
11
12    def test_mul(self):
13        self.assertEqual(calculator.mul(40, 2), 80)
14
15    def test_div(self):
16        self.assertEqual(calculator.div(40, 2), 20)
17
18    def test_add_strings(self):
19        self.assertEqual(calculator.add('a', 'b'), 'ab')
20
21    def test_fail(self):
22        self.assertEqual(calculator.sub(2, 4), 0)
```


Unittest - test runners

```
$ python -m unittest unittest_example
```

```
...F..  
===== FAIL: test_fail (unittest_example.CalculatorTestCase) =====  
-----  
Traceback (most recent call last):  
  File "unittest_example.py", line 22, in test_fail  
    self.assertEqual(calculator.sub(2, 4), 0)  
AssertionError: -2 != 0  
-----  
Ran 6 tests in 0.001s  
  
FAILED (failures=1)
```

Unittest - test runners

\$ nosetests unittest_example.py

```
...F..
```

```
=====
```

```
FAIL: test_fail (examples.unittest_example.CalculatorTestCase)
```

```
-----
```

```
Traceback (most recent call last):
```

```
  File "/Users/elyezer/code/test-automation-examples/examples/unittest_example.py", line 22, in test_fail
    self.assertEqual(calculator.sub(2, 4), 0)
```

```
AssertionError: -2 != 0
```

```
-----
```

```
Ran 6 tests in 0.001s
```

```
FAILED (failures=1)
```

Unittest - test runners

\$ py.test unittest_example.py

```
===== test session starts =====
platform darwin -- Python 2.7.5 -- py-1.4.25 -- pytest-2.6.3
collected 6 items

unittest_example.py ...F..

===== FAILURES =====
_____ CalculatorTestCase.test_fail _____

self = <examples.unittest_example.CalculatorTestCase testMethod=test_fail>

    def test_fail(self):
>         self.assertEqual(calculator.sub(2, 4), 0)
E         AssertionError: -2 != 0

unittest_example.py:22: AssertionError
===== 1 failed, 5 passed in 0.04 seconds =====
```

DDT

```
1 import unittest
2 import calculator
3 from ddt import ddt, data
4
5
6 @ddt
7 class CalculatorDDTTestCase(unittest.TestCase):
8     @data(
9         (40, 2, 42),
10        (2, 2, 4),
11        (20, 30, 50)
12    )
13    def test_add(self, test_data):
14        a, b, c = test_data
15        self.assertEqual(calculator.add(a, b), c)
```

Requests

```
1 import requests
2 import urlparse
3
4 GITHUB_API_URL = 'https://api.github.com/'
5 ROBOTTELO_COLLABORATORS_URL = urlparse.urljoin(
6     GITHUB_API_URL, '/repos/SatelliteQE/robottelo/collaborators')
7
8 response = requests.get(ROBOTTELO_COLLABORATORS_URL)
9 results = response.json()
10
11 print 'Robottelo collaborators:'
12 for collaborator in results:
13     print '* {0} - {1}'.format(
14         collaborator['login'], collaborator['html_url'])
```

Requests - output

```
$ python requests_example.py
Robottelo collaborators:
* alda519 - https://github.com/alda519
* elyezer - https://github.com/elyezer
* Ichimonji10 - https://github.com/Ichimonji10
* JacobCallahan - https://github.com/JacobCallahan
* kbidarkar - https://github.com/kbidarkar
* omaciel - https://github.com/omaciel
* sg hai - https://github.com/sg hai
* sthirugn - https://github.com/sthirugn
* tkolhar - https://github.com/tkolhar
```

Fabric

```
1  from fabric.api import run
2
3
4  def host_type():
5      run('uname -s')
```

Fabric - output

```
$ fab -H localhost,elyezer.com host_type
[localhost] Executing task 'host_type'
[localhost] run: uname -s
[localhost] Passphrase for private key:
[localhost] out: Darwin
[localhost] out:

[elyezer.com] Executing task 'host_type'
[elyezer.com] run: uname -s
[elyezer.com] out: Linux
[elyezer.com] out:

Done.
Disconnecting from elyezer.com... done.
Disconnecting from localhost... done.
```


Selenium

```
1  from selenium import webdriver
2
3  browser = webdriver.Firefox()
4  browser.maximize_window()
5  browser.get('https://www.python.org/')
6
7  element = browser.find_element_by_css_selector('.introduction > p')
8  print element.text
9
10 browser.close()
```

Splinter

```
1  from splinter import Browser
2
3  browser = Browser()
4  browser.driver.maximize_window()
5  browser.visit('https://www.python.org/')
6
7  element = browser.find_by_css('.introduction > p')
8  print element.text
9
10 browser.quit()
```

Por quê usar o Splinter?

Por ser uma camada de abstração torna mais fácil escrever testes.

Splinter:

```
browser.fill('username', 'janedoe')
```

Selenium:

```
elem = browser.find_element.by_name('username')  
elem.send_keys('janedoe')
```

Aplicação de exemplo

```
1  from flask import Flask, flash, jsonify, render_template, request
2
3  app = Flask(__name__)
4  app.secret_key = '42 is the answer'
5
6
7  @app.route('/', methods=['GET', 'POST'])
8  def index():
9      if request.method == 'POST':
10         name = request.form['name'].strip()
11         if len(name) == 0:
12             flash(u'Please provide a name', 'danger')
13         else:
14             flash(u'Welcome {0}'.format(request.form['name']), 'success')
15     return render_template('index.html')
```

Aplicação de exemplo

```
18 @app.route('/api/echo', methods=['POST'])
19 def api():
20     name = request.form['name'].strip()
21     if len(name) == 0:
22         return jsonify(error=u'Please provide a name')
23     else:
24         return jsonify(response=name)
25
26 if __name__ == '__main__':
27     app.debug = True
28     app.run(host='0.0.0.0')
```

Teste da API

```
1 import unittest
2 import urlparse
3 import requests
4
5
6 SERVER_URL = 'http://localhost:5000/'
7 ECHO_API_URL = urlparse.urljoin(SERVER_URL, '/api/echo')
8
9
10 class APITestCase(unittest.TestCase):
11     def test_success(self):
12         name = 'Python Day'
13         response = requests.post(ECHO_API_URL, data={'name': name})
14         data = response.json()
15         self.assertIn('response', data)
16         self.assertEqual(data['response'], name)
17
18     def test_failure(self):
19         response = requests.post(ECHO_API_URL, data={'name': ''})
20         data = response.json()
21         self.assertIn('error', data)
22         self.assertEqual(data['error'], u'Please provide a name')
```

Teste da UI



```
1 import unittest
2 from splinter import Browser
3
4 SERVER_URL = 'http://localhost:5000/'
5
6
7 class APITestCase(unittest.TestCase):
8     def setUp(self):
9         self.browser = Browser()
10
11     def tearDown(self):
12         self.browser.quit()
13
14     def test_success(self):
15         name = 'Python Day'
16         self.browser.visit(SERVER_URL)
17         self.browser.fill('name', name)
18         self.browser.find_by_tag('button').click()
19         if not self.browser.is_text_present(u'Welcome {0}'.format(name)):
20             self.fail('Unable to find success message')
21
22     def test_failure(self):
23         self.browser.visit(SERVER_URL)
24         self.browser.fill('name', '')
25         self.browser.find_by_tag('button').click()
26         if not self.browser.is_text_present(u'Please provide a name'):
27             self.fail('Unable to find failure message')
```



















Automatização na Prática

- Robottelo: a test suite which exercises The Foreman
- Automation tools: A set of tools to help automating virtual machines to install Foreman and test it using Robottelo

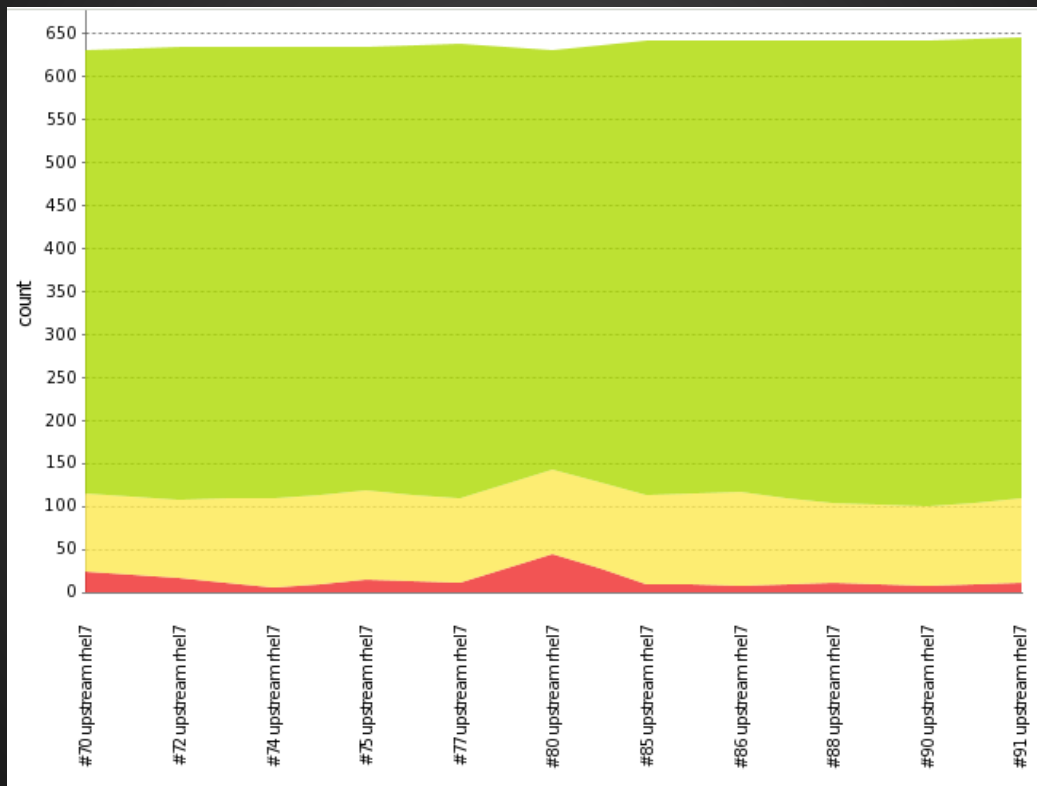
Automatização na Prática

- Jenkins

Configuration Matrix	rhel65	rhel7
downstream		
upstream		
iso		

Configuration Matrix		api	cli	ui
downstream	rhel65			
	rhel7			
upstream	rhel65			
	rhel7			
iso	rhel65			
	rhel7			

Automatização na prática



Referências

- [Introdução a linguagem Python](#)
- [Requests](#)
- [Fabric](#)
- [Selenium](#)
- [Splinter](#)
- [Robottelo](#)
- [Automation-tools](#)

Referências

- [DDT](#)
- [Jenkins](#)

Perguntas?

Obrigado