Além do Webdriver e Page Objects

Charles Kilesse
@chkile

Gustavo Fonseca
@gustavo7lagoas

Agenda

Selenium webdriver

Test frameworks

Page Objects

Factory de Page Objects

“Services” de dados

“3 As”

(powered by pseudo-python)

Agenda

Implicit Wait X Explicit Wait

Camadas de Abstração

Abordagem Estrutural X Abordagem Funcional

Navegação entre páginas

(powered by pseudo-python)

DRY, YAGNI and KISS whenever you can.

ref: http://talkingabouttesting.com/2015/06/08/testes-de-aceitacao-automatizados/

BÁSICO

Selenium Webdriver

“Permite a automação de navegadores. É isso!”

ref: https://github.com/seleniumhq/selenium

ref: www.seleniumhq.org

ref: selenium-python.readthedocs.org/

```python
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Firefox()
driver.get("http://www.python.org")
assert "Python" in driver.title
elem = driver.find_element_by_name("q")
elem.send_keys("pycon")
elem.send_keys(Keys.RETURN)
assert "No results found." not in driver.page_source
driver.close()
```

ref: http://selenium-python.readthedocs.org/en/latest/getting-started.html

Test Frameworks

O Selenium ajuda com o navegador.

Alguém precisa fazer os testes.

```
class PythonOrgSearch(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Firefox()

    def test_search_in_python_org(self):
        driver = self.driver
        driver.get("http://www.python.org")
        self.assertIn("Python", driver.title)
        elem = driver.find_element_by_name("q")
        elem.send_keys("pycon")
        elem.send_keys(Keys.RETURN)
        assert "No results found." not in diver.page_source

    def tearDown(self):
        self.driver.close()
```

ref: http://selenium-python.readthedocs.org/en/latest/getting-started.html

Page Objects

Separação entre código de teste e código específico da página.

Repositório único dos serviços e operações fornecidos pela página

ref: http://docs.seleniumhq.org/docs/06_test_design_considerations.jsp

```
class PageObject:
    locators = { "name_input": (By.ID, "name"),
                 "email_input": (By.CLASS, "email"),
                 "submit_button": (By.CLASS, "submit_form") }

    def fill_name_input(text):
        element = driver.find_element(locators["name_input"])
        element.send_keys(text)

    def fill_email_input(text):
        element = driver.find_element(locators["email_input"])
        element.send_keys(text)

    def click_submit():
        element = driver.find_element(locators["submit_button"])
        element.click()
        return ConfirmationPage
```

Problemas!

Como organizar o código dos testes?

E quando tenho muitos Page Objects?

Como manipulo os dados do que quero testar?

INTERMEDIÁRIO

Múltiplos Page Objects

Utilizar o padrão de projeto Factory

Agora temos um Page Object Factory

Vamos ver como fica?

```
class PageObjectsFactory:
    page_map = {
        "home": HomePage,
        "form": FormPage,
        "confirmation": ConfirmationPage
    }

    def create(page_key):
        return page_map[page_key]()
```

Dados para testes

Utilizar o Padrão de Projeto Service que cria e manipula os dados pro seu teste

E este caso como fica?

```
class UserService:
    def create_user(name, email):
        data = {
            "username": name,
            "email": email
        }

        api_layer.put("/user", data)
        return User(name, email)
```

Organização do código

Modelo 3As

Arrange, Act, Assert

Aqui também temos um exemplo

```
class TestCases:
    def test_user_form_submit():
        # Arrange
        user = user_service.create_user("Charles",
        "charles.kilesse@outlook.com")

        # Act
        form_page = page_factory.create("form")
        form_page.fill_name_input(user.name)
        form_page.fill_email_input(user.email)
        confirmation_page = form_page.click_submit()
        confirmation_message = confirmation_page.get_message()

        # Assert
        assertEqual(confirmation_message, "Yay! Much Success!")
```

E o teste lá do início?

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Firefox()  TEST FRAMEWORK
driver.get("http://www.python.org")  PAGE OBJECT
assert "Python" in driver.title  TEST FRAMEWORK
DO YOU EVEN 3AS?
elem = driver.find_element_by_name("q")  PAGE OBJECT
elem.send_keys("pycon")  PAGE OBJECT
elem.send_keys(Keys.RETURN)  PAGE OBJECT
assert "No results found." not in driver.page_source  TEST FRAMEWORK
driver.close()  TEST FRAMEWORK
```

AVANÇADO

Implicit X Explicit Waits

Simplicidade X Controle

Camadas de abstração

Page Objects

DOM Elements

Domain

Behavior

Domain Specific Language (DSL)

Abordagem estrutural ou funcional?

Funcional:

```
login(username, password)
ou
login_as(user)
```

Estrutural:

```
enter_name(username)
enter_password(password)
click_login()
```

E a navegação entre as páginas?

Page Object retorna outro Page Object

ou navegação implícita

ou eu crio um objeto de navegação?

BDD?

Documentação do sistema.

Abstração das regras de negócio.

Testes são efeito colateral.

Palestra sobre abstrações http://seleniumsimplified.com/2014/11/automation-abstractions-page-objects-and-beyond-conference-talk/

Melhores Práticas Selenium

https://seleniumhq.github.io/docs/best.html#best_practices

OBRIGADO!

Charles Kilesse

charles.kilesse@outlook.com

@chkile

Gustavo Fonseca

gustavo7lagoas@gmail.com

@gustavo7lagoas