PAPRYQARZ
WE TEST WITH TASTE

(jeszcze) efektywniejsze wykorzystanie keywordów w testowaniu

czyli jak skutecznie pisać własne słowa kluczowe

O autorze

- Od ponad 10 lat w jakości oprogramowania
- Własne funkcjonalności testowe na licencji open source
- Pracuje w firmie Meelogic

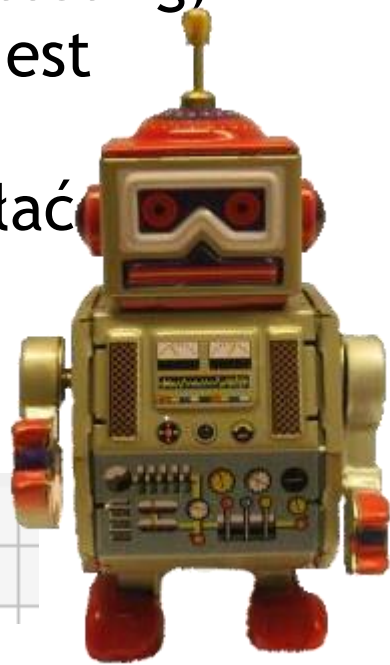
meelogic

LGPL Free Software Free as in Freedom

Wprowadzenie

- Testowanie w oparciu o słowa kluczowe jest powszechną metodą testową (keyword driven testing)
- Słowo kluczowe reprezentuje akcję testową i jest interpretowane przez framework testowy
- Posiada własną składnię w której można wywołać zewnętrzny kod (przez keywordy właśnie!)
- Popularne narzędzie open source: Robot Framework

1	${start}	Set Variable	papryQArz jest OK !
2	Log	${start}	



Co nam dostarcza framework testowy?

- Składnię testów opisaną słowami kluczowymi wraz z IDE
- Logowanie
- Obsługę wyjątków
- Możliwość łatwego dodawanie własnych bibliotek (czyli zbiorów słów kluczowych)

Ograniczenia składni

- Mało przejrzysta składnia przy złożonych problemach
- Brak zagnieżdżonych pętli
- Brak „while”
- Itd.

Pułapka

- Wady przesłaniają zalety
- Rezygnacja z narzędzia i poszukiwanie kolejnego rozwiązania
- Tworzenie własnego narzędzia – czasochłonne i ryzkowne

Rozwiązanie

Wykorzystać zalety narzędzia i wyeliminować wady poprzez przeniesienie logiki testu ze składni testu do keywordów napisanych języku programowania

PAPRYQARZ

Przypadek teoretyczny

Szukanie ilości wystąpień litery „b” w liście zawierającej litery (9 linijek)

checkIfListContainsSomeValue

Settings >>

1	**Log**	Let's define a list of a,b & c and check how many "b" are there				
2	@{letters} =	**Create List**	a	b	c	
3	${howManyB} =	**Set Variable**	0			
4	${howManyB} =	**Convert To Integer**	${howManyB}			
5	**Log**	${howManyB}				
6	:FOR	${i}	IN	@{letters}		
7		**Log**	${i}			
8		**Run Keyword If**	'${i}'=='b'	Set Test Variable	${howManyB}	${howManyB + 1}
9	**Log**	${howManyB}				
10						

PAPRYQARZ

Przypadek teoretyczny poprawiony

Logika testu przeniesiona do Pythona

checkIfListContainsSomeValue_fromLibrary

Settings >>

1	Log	Let's define a list of a,b & c and check how many "b" are there				
2	@{letters} =	Create List	a	b	c	
3	${howManyB} =	How Many Chars In List	${letters}	b		
4	Log	${howManyB}				
5						

```
def how_many_chars_in_list(charsList, charToBeFound):
    howManyChars = 0
    for i in charsList:
        if i==charToBeFound:
            howManyChars += 1
    return howManyChars
```

PAPRYQARZ

Przykład: Selenium2Library

16 linijek

1	${welcomeMsg} =	Set Variable	Hello world
2	Open Browser	http://127.0.0.1:8080	browser=ff
3	Sleep	2	
4	Click Element	link=Examples	
5	Sleep	2	
6	Click Element	link=JSP Examples	
7	Sleep	2	
8	Click Element	xpath=/html/body/p[4]/table	
9	Sleep	2	
10	Wait Until Page Contains Element	name=foo	timeout=10
11	Input Text	name=foo	text=${welcomeMsg}
12	Sleep	2	
13	Click Element	xpath=/html/body/blockquote	
14	${dispMsg}=	Get Text	xpath=/html/body/blockquote
15	Should Contain	${dispMsg}	${welcomeMsg}
16	Close Browser		

PAPRYQARZ

Przykład poprawiony

1	Run Tomcat Example	http://127.0.0.1:8080	Hello world

```
import Selenium2Library as sl
import time
import string

def runTomcatExample(url,welcomeMsg):
    seleniumObj = sl.Selenium2Library()
    seleniumObj.open_browser(url,'ff')
    seleniumObj.click_element('link=Examples')
    time.sleep(2)
    seleniumObj.click_element('link=JSP Examples')
    time.sleep(2)
    seleniumObj.click_element('xpath=/html/body/p[4]/table[1]/tbo
    seleniumObj.wait_until_page_contains_element('name=foo')
    seleniumObj.input_text('name=foo',welcomeMsg)
    time.sleep(2)
    seleniumObj.click_element('xpath=/html/body/blockquote/form/i
    foundText = seleniumObj.get_text('xpath=/html/body/blockquote
    print welcomeMsg
    print foundText
    if string.find(foundText,welcomeMsg) > -1:
        print "Expected text has been found"
    else:
        raise Exception("Expected text has NOT been found")
    seleniumObj.close_browser()
```

Tylko 1 linijka!
Reszta w Keywordzie

Czy w takim razie potrzebujemy framework? TAK !

PAPRYQARZ

Przykład poprawiony raz jeszcze

1	Run Tomcat Example2	http://127.0.0.1:8080	Hello world

Tylko 1 linijka!
Reszta w keywordzie

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

def runTomcatExample2(url,welcomeMsg):
    browser = webdriver.Firefox()
    browser.get(url)
    time.sleep(2)
    browser.find_element_by_link_text('Examples').click()
    time.sleep(2)
    browser.find_element_by_link_text('JSP Examples').click()
    time.sleep(2)
    browser.find_element_by_xpath('/html/body/p[4]/table[1]/tbody
    element = WebDriverWait(browser, 10).until(EC.presence_of_ele
    time.sleep(2)
    for i in range(10):
        element.send_keys(Keys.BACKSPACE)
    element.send_keys(welcomeMsg)
    time.sleep(2)
    browser.find_element_by_xpath('/html/body/blockquote/form/inp
    foundText = browser.find_element_by_xpath('/html/body/blockqu
    time.sleep(2)
    print welcomeMsg
    print foundText
    if string.find(foundText,welcomeMsg) > -1:
        print "Expected text has been found"
    else:
        raise Exception("Expected text has NOT been found")
    browser.quit()
```

Kiedy jeszcze warto pisać własne keywordy ?

- Chcemy wykorzystać wszystkie zalety frameworka testowego ale nie ma potrzebnej funkcjonalności testowej
- Tworzymy własną generyczną bibliotekę testową

lub

- Tworzymy własną specyficzną bibliotekę testową dedykowaną do naszego środowiska testowego (trudno oczekiwać, że istnieje skoro ma być dedykowana do środowiska)

PAPRYQARZ
WE TEST WITH TASTE

kilka słów o praktycznych aspektach tworzenia słów kluczowych

na przykładzie Robot Framework

Co to jest biblioteka testowa?

- Biblioteka to najczęściej po prostu zbiór funkcji w wybranym języku używanych przez framework testowy
- Najczęściej używany wzorzec projektowy: adapter (ang. *wrapper*). Przykład: robotowa biblioteka Selenium2Library – stanowi warstwę pośredniczącą między Selenium a Robotem

PAPRYQARZ

To łatwe !

„Creating your own test libraries is a breeze.”

robotframework.org

Jakie języki wspiera Robot Framework

- Python – najbardziej naturalna forma pisania pluginów, gdyż Robot Framework jest też pisany w Pythonie.
- Java – trochę bardziej skomplikowana forma, do wykorzystania jeśli jest wyraźny powód do użycia Javy.

PAPRYQARZ
WE TEST WITH TASTE

dziękuję za uwagę

częśc praktyczna i czas na pytania