Słowa kluczowe jak góry lodowe

czyli rzecz o bibliotekach testowych

Marcin Kowalczyk

marcin.kowalczyk@tieto.com

Spis treści

Dlaczego słowa kluczowe są jak góry lodowe, po co tworzyć własne biblioteki testowe

Krótkie 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
- Popularne narzędzie korzystające z tej metody: Robot Framework

1	${start} =	Set Variable	Hello world TESTWAREZ2014 !
2	Log	${start}	

Słowa kluczowe jak góry lodowe

- Gdzie ma być ciężar logiki testu? Są różne podejścia

Zalety testowania w oparciu o słowa kluczowe

- Co dostarcza nam framework testowy?
  - Składnię testów opisaną słowami kluczowymi
  - Logowanie
  - Obsługę wyjątków rzucanych w bibliotekach testowych
  - Możliwość łatwego dodawania własnych bibliotek

Wady testowania w oparciu o słowa kluczowe

- Mało przejrzysta i efektywna składnia testów zbudowanych ze słów kluczowych gdy mamy do czynienia ze złożonym przypadkiem testowym
- „Programowanie w tabelkach” często zniechęca inżynierów testów do korzystania z narzędzia

Cel: wykorzystać zalety, **wyeliminować wady**

Przykład klasycznego podejścia

- Większość wbudowanych generycznych bibliotek testowych często dostarcza funkcjonalność, w której ciężar logiki testu leży po stronie słów kluczowych
- „Programowanie w tabelkach” często zniechęca inżynierów testów
- Przykład: szukanie ilości wystąpień litery „b” w liście zawierającej litery

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						

Przeniesienie ciężaru logiki testu w kierunku biblioteki

- W bardzo prosty sposób można napisać własną bibliotekę i zdefiniować własne słowo kluczowe
- Zysk: 5 słów kluczowych mniej w skrypcie, reużywalność

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
```

Przykład z życia wzięty

- Przykład: 16 linijek używających Selenium2Library

1	${welcomeMsg} =	Set Variable	Hello world Testwarez 2014 !
2	Open Browser	http://127.0.0.1:8080	browser=ff
3	Sleep	2	
4	Click Element	link=Examples	
5	Sleep	2	
6	Click Element	link=JSP Examples	
7	Sleep	2	
8	Click Element	xpath=/html/body/p[4]/table	
9	Sleep	2	
10	Wait Until Page Contains Element	name=foo	timeout=10
11	Input Text	name=foo	text=${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		

Przeniesienie funkcjonalności Selenium do biblioteki

- Nowe słowo kluczowe używające Selenium2Library

1	Run Tomcat Example	http://127.0.0.1:8080	Hello world Testwarez 2014 !

```
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()
```

Przeniesienie funkcjonalności Selenium do biblioteki c.d.

- Nowe słowo kluczowe używające Selenium WebDriver

1	Run Tomcat Example2	http://127.0.0.1:8080	Hello world Testwarez 2014 !

```
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()
```

Gdy brak funkcjonalności testowej tworzymy własny „keyword”

- 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)

Praktyczne aspekty tworzenia własnych bibliotek testowych

Czym jest biblioteka testowa?

- Biblioteka to 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

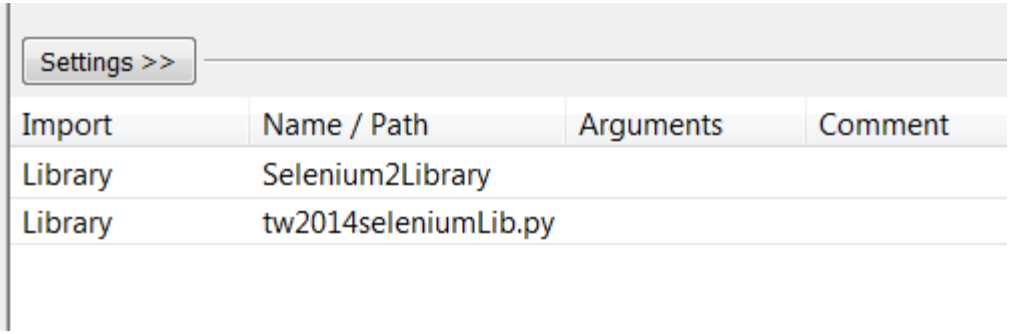
Jakie języki wspiera Robot Framework?

- Python – najbardziej naturalna forma pisania pluginów, gdyż Robot Framework jest też pisany w Pythonie. Biblioteki są interpretowane przez pyBot i jyBota
- Java – trochę bardziej skomplikowana forma, do wykorzystania jeśli jest wyraźny powód do użycia Javy. Biblioteki interpretowane tylko przez jyBota. Wymaga instalacji Jythona.

Tworzenie biblioteki i import w skrypcie

- Biblioteka to po prostu moduł Pythonowy (plik nazwa.py)
- Widoczna jeśli jest w tym samym katalogu co skrypt

  lub
- W dowolnej lokalizacji wskazanej przez PYTHONPATH (zmienna systemowa wskazująca, gdzie znajdują się biblioteki Pythonowe



- Większość opublikowanych bibliotek znajduje się na PyPI (Python Package Index) – do zainstalowania z sieci jedną komendą

Mapowanie funkcji na słowa kluczowe

- Podstawowym założeniem jest fakt, że silnik Robot Frameworka mapuje nazwy funkcji bibliotecznych na słowa kluczowe
- Słowa kluczowe mogą przyjmować argumenty i zwracać wartości jak zwykłe funkcje
- Przykład mapowania:

Nazwa metody pythonowej	Zamapowane słowo kluczowe
jakas_nazwa_funkcji	Jakas Nazwa Funkcji
jakasNazwaFunkcji	Jakas Nazwa Funkcji
click_element	Click Element

Drukowanie do logu i ustawianie negatywnego rezultatu

- Drukowanie do logu jest niezwykle proste – framework przechwytuje standardowe wyjście
- Aby ustawić test w stanie negatywnym wystarczy rzucić dowolny wyjątek - zostanie obsłużony przez framework

Akcja	Jak	Przykład
Drukowanie do logu	Drukowanie na standardowe wyjście	print „cos drukujemy”
Ustawianie negatywnego rezultatu	Rzucenie dowolnego wyjątku	Raise Exception(„jakis komunikat”)

Dokumentowanie słów kluczowych

- Libdoc - narzędzie robotowe do dokumentowania bibliotek testowych
- Dokumentuje słowa kluczowe i komentarze

tw2014seleniumLib

Library scope: global
Named arguments: supported

Introduction

Przykladowa biblioteka testowa na potrzeby konferencji Testwarez 2014 !

Shortcuts

Run Tomcat Example · Run Tomcat Example 2

Keywords

Keyword	Arguments	Documentation
Run Tomcat Example	*url, welcomeMsg*	Uruchamia przyklady z Tomcata uzywajac robotowej biblioteki Selenium2Library. Przyklad uzycia: Run Tomcat Example | http://127.0.0.1:8080 | Hello world Testwarez !
Run Tomcat Example 2	*url, welcomeMsg*	Uruchamia przyklady z Tomcata uzywajac biblioteki Selenium WebDriver. Przyklad uzycia: Run Tomcat Example2 | http://127.0.0.1:8080 | Hello world Testwarez again !

Altogether 2 keywords.
Generated by libdoc on 2014-08-22 12:24:08.

Kilka słów o bibliotekach w Javie

- JavalibCore – stworzona przez twórców Robota biblioteka umożliwiającą w miarę łatwe pisanie pluginów w Javie
- Dostarcza 2 klasy do wyboru po których powinny dziedziczyć nowe biblioteki:
  - ClassPathLibrary
  - AnnotationLibrary
- Jeśli brak wyraźnego powodu do skorzystania z Javy lepiej pisać biblioteki w Pythonie

Dziękuję za uwagę

Czas na pytania i pokaz praktyczny.

Knowledge.
Passion.
Results.

Marcin Kowalczyk

marcin.kowalczyk@tieto.com