Magazine

Automat do raportowania błędów

Autor: Jacek Okrojek
O autorze: Jacek Okrojek, absolwent Wydziału Fizyki Technicznej, Informatyki i Matematyki Stosowanej Politechniki Łódzkiej, specjalizacja Sieci i Systemy Teleinformatyczne, tester, test leader, freelance developer, od ponad 6 lat zajmuje się testowaniem i tworzeniem oprogramowania w kraju i za granicą w firmie Ericpol Telecom orazjako niezależny konsultant, prowadził i uczestniczył w testach na poziomie podstawowym, funkcyjnym, integracyjnym i systemowym.

Intermediate
Level

3
Magazine Number

Testowanie oprogramowania
Section in the magazine

Wprowadzenie

Selenium to popularny projekt opensource wspomagający automatyzacje testowania aplikacji web. Jego wykorzystanie nie musi się ograniczać tylko do tego zadania. Możliwości jakie

posiada narzędzie mogą być użyte do automatyzacji wielu innych zadań. W artykule pokażę w zarysie jak przy jego pomocy zautomatyzować proces raportowania błędów, gdy korzystamy z popularnych systemów lub rozwiązania dedykowanych wykorzystujących interface WWW.

Automatyczne wykonywanie testów, mimo pewnych ograniczeń, jest szeroko wykorzystywane w wielu organizacjach. Ostatnim punktem tego procesu jest raportowanie błędów, które opiera się automatyzacji i wymaga ingerencji testera. Częsta praktyka jaką jest wykonywanie nocnych testów sprawia, że następny dzień spędzamy na przeglądaniu wyników i raportowaniu błędów. Jest to zadanie żmudne i bądźmy szczerzy mało interesujące. Jego przyspieszenie, choć nie łatwe byłoby miłym i również cennym usprawnieniem.

Ogólne założenia

Użycie przedstawionego rozwiązania będzie możliwe o ile aktualnemu systemowi testującemu, gdy test zakończy się niepowodzeniem, będziemy mogli zlecić wywołanie opisanej aplikacji i przekazanie jej potrzebnych danych. Nie powinno być to problemem w większości przypadków, Selenium posiada drivery dla kilku najpopularniejszych języków programowania. Opisane rozwiązane będzie zaimplementowane w Python'ie. Nic nie stoi na przeszkodzie by to zmienić i użyć języka programowania, który najbardziej nam odpowiada.

W pracy testera języki skryptowe jak Python czy Perl sprawdzają się lepiej niż języki wymagające kompilacji. Tworzenie prostych narzędzi jest w nich dużo szybsze, nieoceniona jest tez możliwość wykorzystania interpretera w procesie tworzenia pomocnych aplikacji jak i testowania głównego systemu.

Zadaniem aplikacji jest sprawdzić czy błąd nie był już zaraportowany a następnie, gdy jest to uzasadnione zaraportowanie niepokojącego incydentów. Kluczowy dla efektywności działania systemu jest pierwszy krok, czyli sprawdzenie czy informacja o błędzie nie istniej już w bazie. Dobrą praktyką, która często jest już stosowana w wielu organizacjach jest stosowanie konwencji przy raportowaniu błędów.

Z uwagi na specyfikę testowanych systemów prezentowane rozwiązanie to tylko zarys implementacji. Skupiłem się na przygotowaniu środowiska, dobrych praktyk i prezentacji rozwiązań typowych problemów jakie możecie napotkać przy dostosowywaniu rozwiązania do własnych potrzeb. Wykorzystywany w przykładach system raportowania błędów Google jest jednym z prostszych i nie wymaga dodatkowego omawiania.

Przygotowanie środowiska

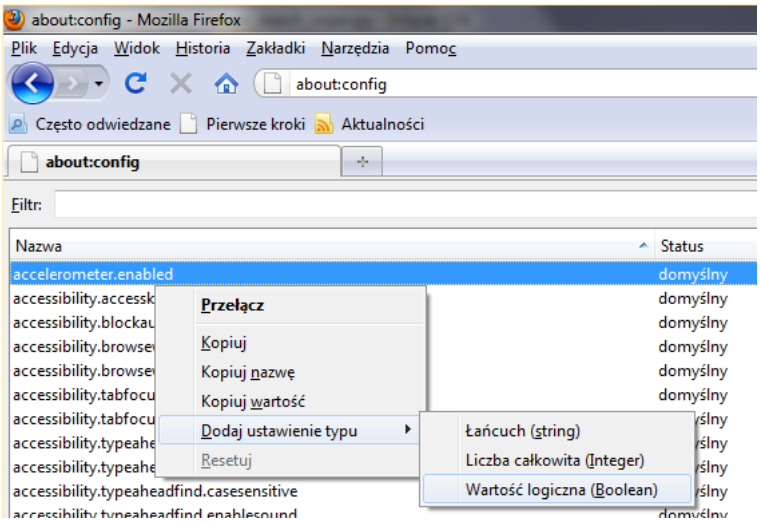
Opisany proces dotyczy systemu Winows i może wymagać modyfikacji dla systemów UNIX/Linuxowych. Główne składniki jakie będą potrzebne to:

- Python 2.6 lub nowszy – do pobrania z http://python.org, korzystam z wersji PortablePython (http://portablepython.com) –jest wzbogacona o dodatkowe biblioteki i nie wymagająca instalacji przez co można z niej korzystać w systemach gdzie nie mamy uprawnień administratora
- Selenium RC – http://seleniumhq.org/
- Firefox – korzystam z wersji 3.5

oraz dodatkowo:

- Remember Certificate Exception (https://addons.mozilla.org/pl/firefox/addon/10246/?src=api)
- Firebug - http://getfirebug.com przyda sie przy odnajdywaniu elementów na stronie
- Selenium IDE

Po zapisaniu na dysku instalacji/rozpakowaniu wszystkich podstawowych elementów możemy przystąpić do przygotowania przeglądarki. Zaczniemy dodając katalog, w którym zainstalował się Firefox do zmiennej systemowej PATH. Choć do pracy z Selenium nie jest to wymagane to wygodne i sprawdzone rozwiązanie. Firefox ma możliwość tworzenia odrębnych profili użytkownika, w rama których można określić m.in jakie opcje i ustawienia maja być aktywne oraz jakie dodatki maja być zainstalowane. Startując przeglądarkę z linii poleceń z opcją –P (firefox –P) będziemy mieli dostęp do tej funkcji. Tworzymy nowy profil i wybieramy katalog, w którym ma być utworzony. Po wystartowaniu przeglądarki z nowym profilem w polu adresu wpisujemy “about:config” i zatwierdzamy klawiszem Enter. Klikając prawym przyciskiem w dowolnym miejscu ekranu dodajemy zmienne:



extensions.update.notifyUser (type=boolean; value=false)

extensions.newAddons (type=boolean; value=false)

Pozostaje jeszcze tylko wprowadzić drobne zmiany w ustawieniach. Z menu wybieramy Narzędzia/Opcje... następnie z zakładki Karty odznaczamy wszystko, z zakładki Treść odznaczamy „Zablokuj wyskakujące okna” i na zakładce Bezpieczeństwo wybieramy Ustawienia i odznaczamy wszystkie ostrzeżenia. W nowo utworzonym profilu warto również zainstalować wymienione wyżej dodatki. Przydadzą się nam w dalszej pracy.

Sprawdzamy czy incydent jest już zaraportowany

Proces raportowania problemów należy rozpocząć od sprawdzenie czy problem nie został zaraportowany wcześniej**.** Posłużymy się Selenium IDE do stworzenia szkieletu, który dalej będziemy modyfikować. Jeśli zainstalowaliśmy dodatek Selenium IDE możemy go uruchomić z menu Narzędzia. W tym momencie wszystkie nasze aktywności na stronie są nagrywane.

W polu adresu wpisujemy adres strony pozwalającej przeszukiwać zaraportowane błędy. W prezentowanym przykładzie będzie to http://code.google.com/p/selenium/issues/advsearch. W polu 'Search within' wybieramy All isssues. Nastepnie do pola 'with **all** of the words' wpisujemy '**Fault in line'.** Następnie naciskamy przycisk Search. W tym momencie wyłączamy nagrywanie naciskając czerwony, okrągły przycisk w prawej górnej części okna Selenium IDE i eksportujemy skrypt (Plik/Export Test Case As...).

Dobrą praktyką, która często jest już stosowana w wielu organizacjach jest stosowanie konwencji przy raportowaniu błędów. Konwencja zwykle dzieli błędy na kilka klas i określa m.in. jak tytułować raporty i jakie informacje w nich umieszczać. Ułatwia to raportowanie, późniejszą analizę problemu i oczywiście przeszukiwanie raportów Zalety tej praktyki widać szczególnie w dużych i rozproszonych organizacjach.

Możemy teraz otworzyć skrypt w swoim ulubionym IDE. Jest on zapisany jako unittest pythona, nie do końca mi to odpowiada więc sugeruje zmodyfikować go do prostszej postaci

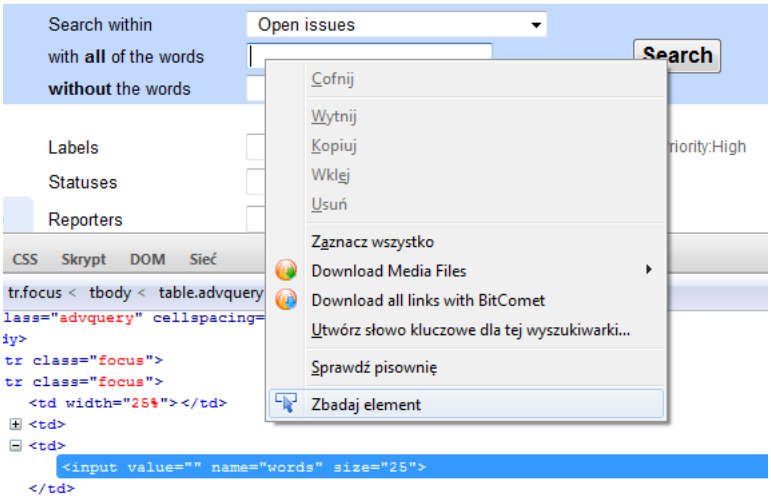
```
searchedWords = 'TC_ID'

sel = selenium("localhost", 4444, "*firefox",
"http://code.google.com/")
sel.start()

sel.open("http://code.google.com/p/selenium/issues/advsearch")
sel.wait_for_page_to_load("30000")
sel.select("//div[@id='maincol']/form/table/tbody/tr[1]/td[3]/select", "label=regexp:\\sAll issues")
sel.type("words", searchedWords)
sel.click("btn")
res = sel.get_html_source()
```

W trzeciej linii przekazujemy driverowi selenium adres i port Selenium RC servera i domenę, z której będziemy korzystać. Następnie startujemy poprzez serwer startujemy przeglądarkę. Kolejnymi krokami są polecenie otwarcie strony i oczekiwanie na jej załadowanie. Są to czynności, które nie będą różnić się przy pracy z innymi bug trakcer'ami.

Pozostaje wpisanie danych do odpowiednich pól i zlecenie pokazania wyników. Selenium dysponuje kilkoma sposobami umożliwiającymi lokalizacje elementów na stronie. Najprostszym jest wykorzystanie ich atrybutu `name` Jeśli nie chcemy mozolnie przeczesywać kodu HTML strony wygodnie posłużyć się dodatkiem Firebug. Klikając prawym przyciskiem na interesujący element i wybierając opcje 'Zbadaj element' możemy szybko poznać jego atrybuty. Ostatnią czynnością jest pobranie kodu strony z rezultatami wyszukiwania. Stronę taką należy przeanalizować i zdecydować czy zarejestrować incydent.



Testujemy

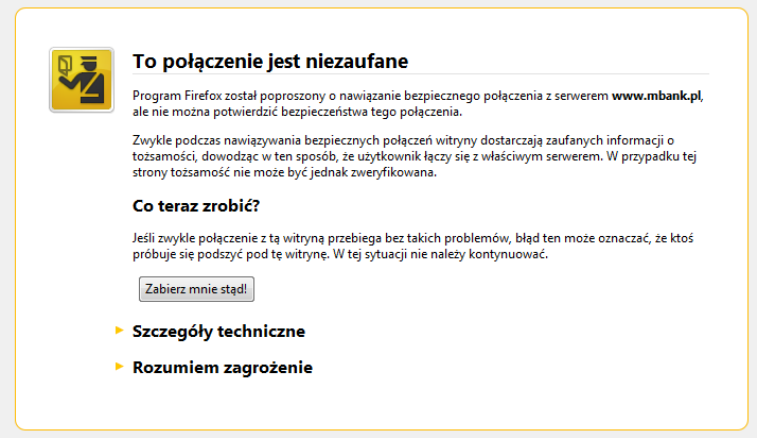
Przed uruchomieniem naszego skryptu musimy wystartować server Selenium RC. Przechodzimy do katalogu gdzie znajduje się rozpakowane Selenium RC, przechodzimy do podkatalogu selenium-server i startujemy serwer wydając polecenie

java –jar selenium-server.jar –firefoxProfileTemplate *scieżka_do_profilu*

Po chwili powinny uruchomić się dwa okna przeglądarki i w jednym z nich powinniśmy zobaczyć wyniki wyszukiwania.

Raportowanie incydentów

Podobnie jak przy wyszukiwaniu elementów posłużymy się Selenium IDE by stworzyć podstawowy szkielet. Po rozpoczęciu nagrywania przechodzimy na stronę http://code.google.com/p/selenium/issues/list i rozpoczynamy proces rejestracji błędu. Pierwszym krokiem jest zalogowanie się do systemu następnie przechodzimy do strony umożliwiającej wpisanie danych o napotkanym problemie. Należy wziąć pod uwagę, możliwość automatycznego zapamiętywania haseł i logowania na tym etapie. Problemem jaki możemy napotkać jest autoryzacja z wykorzystaniem certyfikatów SSL. Selenium każdorazowo będzie wymagało potwierdzenia certyfikatu co mija się z koncepcją automatyzacji.



Rozwiązaniem tego problemu jest instalacja pluginu Remember Certificate Exception. Jego zadanie to przeprowadzenie nas przez ten proces automatycznie. Łącząc wyszukiwanie i raportowanie błędów w jedną całość zakończymy prace nad naszym rozwiązaniem. Jako ostatnie polecenie w skrypcie powinno znaleźć się zamknięcie przeglądarki instrukcją:

sel.stop()

Możemy ją pominąć dając testerowi możliwość ostatecznej decyzji o zaraportowaniu błędu.

Podsumowanie

Automatyzacja związana jest z dużym nakładem pracy przy jej wdrażaniu i jak zawsze należy się zastanowić nad korzyściami i wadami jej stosowania. W zależności od specyfiki projektu prezentowane rozwiązanie może zyskać różne oceny. Dotyczy to zarówno samej koncepcji automatycznego raportowania błędów jak i zaprezentowanej implementacji. Diabeł tkwi w szczegółach i to one zdecydują jak bardzo może być ono przydatne. Wykorzystanie Selenium RC pozwala ukryć szczegóły protokołu HTML, charakteryzuje się dużą prostotą łatwością wprowadzania zmian. Wadą może być ewentualna niestabilność serwera. Warto na pewno przyjrzeć się na pewno narzędziu jakim jest Selenium.