



Technologia i rozwiązania

Selenium

Automatyczne testowanie aplikacji



Prashanth Sams

[PACKT] open source*
PUBLISHING community experience distilled

Tytuł oryginału: Selenium Essentials

Tłumaczenie: Jakub Hubisz

ISBN: 978-83-283-3039-9

Copyright © Packt Publishing 2015

First published in the English language under the title 'Selenium Essentials' - 9781784394332.

Polish edition copyright © 2017 by Helion SA. All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/selata>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

O autorze	7
O korektorze merytorycznym	8
Przedmowa	9
Rozdział 1. Selenium IDE	13
Odtwarzanie WebDriver	17
Priorytety lokatorów	19
Unikanie eksportu Selenium	20
Schówek Selenium IDE	22
Testy sterowane danymi	23
Metody JavaScript zdefiniowane przez użytkownika	23
Funkcje JavaScript w Selenium IDE	26
Proste wywołanie JavaScript	27
Przewijanie kółkiem myszy	28
Parametryzacja przy wykorzystaniu tablic	28
Selenium Builder	29
Nagrywanie i odtwarzanie	30
Testy sterowane danymi	31
Selenium Builder w chmurze	33
Podsumowanie	34
Rozdział 2. Testy na wielu przeglądarkach z wykorzystaniem Selenium WebDriver	35
Testy kompatybilności z wykorzystaniem Selenium WebDriver	36
TestNG	37
Testy Selenium w chmurze dla wielu przeglądarek	40
SauceLabs	40
BrowserStack	42
TestingBot	43

Testy w przeglądarce bezinterfejsowej	45
PhantomJS	45
HTMLUnitDriver	47
Zmienianie interfejsów użytkownika	48
Przeglądarka Firefox	49
Przeglądarka Chrome	49
Testy na konkretnych wersjach przeglądarki Firefox	50
Testy z niestandardowego profilu Firefox	51
Testy z niestandardowego profilu Chrome	52
Podsumowanie	52
Rozdział 3. Funkcje Selenium WebDriver	53
Podstawowe funkcje WebDriver	54
Lokalizowanie elementów	55
Funkcje elementów WebElements	60
Nawigacja	63
Ciasteczka	64
Funkcje okna	66
Funkcje wybierające	70
Obsługa alertów i okien wyskakujących	72
Akcje myszy i klawiatury	74
Podsumowanie	80
Rozdział 4. Selenium WebDriver — najlepsze praktyki	81
Obsługa stron wykorzystujących Ajax	82
Metoda isElementPresent	82
Oczekiwanie	83
Oczekiwanie jawne	84
Limity czasu	87
Wzorzec Page Object	88
Klasa PageFactory	90
Adnotacja @FindBy	92
Adnotacja @FindBy	93
Klasa EventFiringWebDriver	93
Przykład sterownika uruchamiającego zdarzenia	99
Obsługa ramek iframe	103
Obsługa okien wyskakujących systemu operacyjnego i przeglądarki przy wykorzystaniu Java Robot	105
Profil Firefox do pobierania plików	109
Klasa JavascriptExecutor	110
Skrolowanie strony	111
Podświetlanie elementów	112
Otwieranie nowego okna przeglądarki	113
Kolektor błędów JavaScript	114
Podsumowanie	116

Rozdział 5. Frameworki Selenium WebDriver	117
Programowanie sterowane zachowaniem	118
Framework BDD Cucumber	118
Framework sterowany danymi JXL API	127
Zapis i odczyt arkusza Excela	127
Proste testy sterowane danymi	129
Testowanie sterowane danymi z wykorzystaniem biblioteki	131
Testowanie sterowane danymi z wykorzystaniem TestNG i adnotacji @dataProvider	134
Framework sterowany danymi Apache POI	136
Model HSSF — arkusz binarny	138
Model XSSF — arkusz SpreadsheetML (.xlsx)	141
Model SS — arkusze binarne i SpreadsheetML	143
Framework sterowany danymi z pliku tekstowego	144
Testy sterowane danymi z wykorzystaniem TestNG i adnotacji @dataProvider	
— plik tekstowy	146
Framework sterowany danymi z pliku właściwości	148
Testy sterowane danymi z wykorzystaniem TestNG i adnotacji @dataProvider	
— plik właściwości	151
Framework sterowany danymi CSV	153
Framework sterowany słowami kluczowymi	155
Framework hybrydowy	157
Podsumowanie	159
Skorowidz	161

Selenium IDE

Selenium IDE (ang. *Integrated Development Environment* — zintegrowane środowisko programistyczne) jest narzędziem typu *open source*, pozwalającym nagrywać i odtwarzać skrypty Selenium. Jest zintegrowane z przeglądarką Firefox i stanowi jej rozszerzenie. Jest odświeżonym narzędziem do automatyzacji testów interfejsów sieciowych korzystających z różnego rodzaju lokatorów na stronie. Lokatory mogą być oparte na atrybutach lub strukturze i mogą zawierać identyfikator, nazwę, łącze, ścieżkę XPath, CSS i DOM. IDE zawiera całą funkcjonalność Selenium, pozwalającą użytkownikom na ręczne nagrywanie, odtwarzanie, edycję i debugowanie testów w przeglądarce. Akcje użytkownika na stronie mogą zostać nagrane i wyeksportowane w dowolnym spośród najpopularniejszych języków programowania, takich jak Java, C#, Ruby i Python.

Program Selenium Builder jest alternatywnym narzędziem dla Selenium IDE, również pozwalającym nagrywać i odtwarzać działania na stronie internetowej. Jest rozszerzeniem przeglądarki Firefox, podobnym do Selenium IDE, ale posiadającym również pewne unikalne funkcjonalności niedostępne w Selenium IDE. Selenium Builder jest narzędziem udostępnianym przez Sauce Labs, który wykonuje testy na Sauce Cloud za pomocą interfejsu Selenium Builder.

W tym rozdziale dowiesz się o:

- Możliwościach nagrywania i odtwarzania dostępnych za pośrednictwem Selenium IDE.
- Funkcjach Selenium IDE.
- Testach sterowanych danymi w Selenium IDE.
- Funkcjach JavaScript w Selenium IDE.
- Nagrywaniu i odtwarzaniu w Selenium Builder.
- Testach sterowanych danymi w Selenium Builder.
- Selenium Builder w chmurze.

Selenium IDE to rozszerzenie przeglądarki Firefox, pozwalające nagrywać i odtwarzać aplikacje sieciowe. Pozwala jednak na więcej niż tylko nagrywanie i odtwarzanie. Pułapki pozwalają na debugowanie poleceń IDE krok po kroku w trakcie ich wykonywania. IDE posiada trzy różne typy paneli — są to: lewy panel, panel przypadków testowych i panel logowania/referencji/elementów UI/zwijany.




Selenium IDE możesz uruchomić za pomocą menu *Narzędzia: Narzędzia/Selenium IDE*. IDE można otworzyć również za pomocą kombinacji klawiszy *Ctrl+Shift+S* lub za pomocą ikony dostępnej w prawym górnym rogu przeglądarki Firefox. Ikona Selenium została pokazana na poniższym zrzucie ekranowym.



Po uruchomieniu IDE w **lewym panelu** utworzony zostanie nowy przypadek testowy bez nazwy. Aby rozpocząć nowy przypadek testowy, wybierz *File/New Test Case* lub wykorzystaj skrót klawiszowy *Ctrl+N*.

Aby rozpocząć nagrywanie skryptów testowych, kliknij okrągłą, czerwoną ikonę na pasku odtwarzania. Domyślnie przycisk nagrywania będzie aktywny, a skrypty testowe są nagrywane w **Selenese**, specjalnym języku podobnym do HTML. Pasek odtwarzania został przedstawiony na poniższym zrzucie ekranowym.



Suwak *Fast-Slow*  pozwala regulować szybkość wykonywania testu; przycisk *Play All*  pozwala wykonać wszystkie przypadki testowe w zestawie, gdzie zestaw stanowi zbiór przypadków testowych; a przycisk *Play*  pozwala uruchomić aktualny przypadek testowy. Przycisk *Pause/Resume* wstrzymuje na chwilę wykonywanie i pozwala je wznowić w dogodnej chwili.

Panel *Test Case* wyświetla wszystkie nagrane kroki w kolumnach *Command* (polecenie), *Target* (cel) i *Value* (wartość). Kolumna *Command* instruuje IDE, co ma zrobić, i może posiadać trzy różne typy wartości:

- akcje,
- akcesory,
- sprawdzenia.

Selenium IDE posiada listę swbudowanych poleceń, pozwalających wykonywać testy zgodnie z oczekiwaniami. Dodawanie do Selenium IDE poleceń zdefiniowanych przez użytkownika jest możliwe i może być wykonane poprzez rozszerzanie istniejących metod JavaScript. Polecenie może być jednego z trzech powyżej podanych typów. Podczas generowania skryptów omawiane polecenia można łatwo edytować i zamieniać na polecenia alternatywne.

Polecenia akcji pozwalają modyfikować stan aplikacji i mogą być typu `action` (akcja) i `actionAndWait` (akcja i oczekiwanie). Polecenia akcji posiadające sufiks `AndWait` pozwalają stronie w pełni się załadować przed uruchomieniem kolejnego polecenia.

Przykłady poleceń akcji to: `open`, `type`, `typeAndWait`, `select`, `selectAndWait`, `check`, `checkAndWait`, `click` i `clickAndWait`.

Aksesory wykrywają stan aplikacji i zapisują go w zmiennej; `store`, `storeText` i `storeValue` to polecenia wykorzystywane do przechowywania wartości. Na poniższym zrzucie ekranowym `search` to zmienna, a `prashanth sams` to szukane słowo. Następnie wartość jest pobierana i wykorzystywana jako parametr akcji, `${search}`. Powyższy scenariusz przedstawiony został na poniższym zrzucie ekranowym.

Command	Target	Value
<code>open</code>	<code>/?gfe_rd=cr&ei=...</code>	
<code>store</code>	<code>prashanth sams</code>	<code>search</code>
<code>type</code>	<code>css=#gbqfq</code>	<code>\${search}</code>

Sprawdzenia weryfikują stan aplikacji poprzez porównanie z oczekiwanym wynikiem. Dostępne są w trzech różnych trybach: *assert*, *verify* i *waitFor*. *Assert* w przypadku nieudanego sprawdzenia zatrzymuje wykonywanie testu, *verify* w przypadku nieudanego sprawdzenia pozwala na kontynuację testu, a *waitFor* oczekuje na wystąpienie określonego stanu i kończy się porażką po przekroczeniu dozwolonego czasu. Domyślnie dozwolony czas ustawiony jest na 30 sekund. W Selenium IDE dozwolony czas można skonfigurować za pomocą menu *Options*.

Pola *Target* pozwalają IDE na identyfikowanie elementów, a ogólna składnia pola *Target* jest następująca:

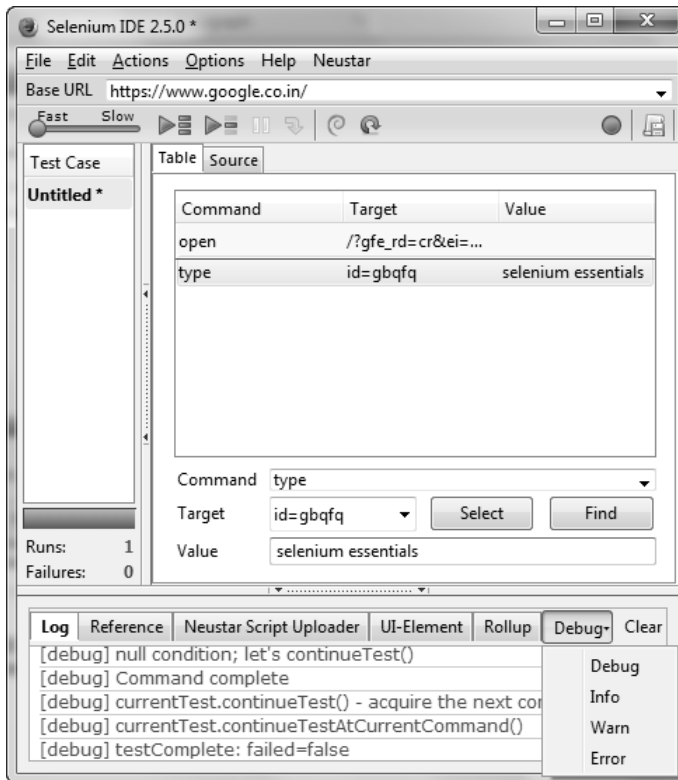
```
locatorType = argument
```

Oto przykładowy *Target*:

```
css=#gbqfq
```

Panel *Log/Reference/UI-Element/Rollup* znajduje się u dołu IDE i został przedstawiony na zrzucie ekranowym zamieszczonym na następnej stronie.

Ten panel pozwala użytkownikowi przeglądać między innymi zalogowane informacje, odniesienia do komend oraz elementów *UI-Element* i *Rollup*. Po zainstalowaniu wtyczki Neustar dla Selenium IDE wraz z innymi zakładkami zostanie pokazana zakładka *Neustar Script Uploader*. Narzędzie *Neustar WPM* (wcześniej znane pod nazwą *Browsermob*) służy do zarządzania testami wydajnościowymi i obciążeniowymi aplikacji sieciowych.



Logowanie zapisuje wszystkie kroki wykonywania testu i wykorzystywane jest głównie do celów debugowania. Menu *Debug* w dolnym panelu zawiera listę opcji: *Info*, *Debug*, *Warn* i *Error*. Opcje te pozwalają filtrować wiadomości dotyczące statusu, ostrzeżeń i błędów i pozwalają poprawić czytelność listy.

Po kliknięciu wiersza panelu *Test Case* zakładka *Reference* będzie zawierać dokładne objaśnienie poleceń IDE dla klikniętego przypadku testowego. W przypadku poleceń zdefiniowanych przez użytkownika zakładka *Reference* nie będzie zawierała żadnych informacji. *Rollup* wykonuje zestaw poleceń w jednym kroku; taki zestaw poleceń może być wielokrotnie wykorzystywany w ramach przypadku testowego. Więcej informacji na temat zakładek *UI-Element* i *Rollup* znajdziesz w dokumentacji dostępnej w menu *Help/UI-Element Documentation*.

Podczas nagrywania skryptów testowych Selenium IDE zapewnia opcje bazujące na interfejsie dla każdego kliknięcia prawego przycisku myszy na elementach strony WWW. Aby to osiągnąć, kliknij prawym przyciskiem na stronie i wybierz *Show All Available Commands* (pokaż wszystkie dostępne polecenia). Poniższy zrzut ekranowy jest skutkiem takiego działania.

open /?gfe_rd=cr&ei=wZ3OU7qXNcbN8geQn4CoCg&gws_rd=ssl
assertTitle Google
assertValue
assertText //div[@id='gb']/div/div +YouGmailImages+YouSearchYouTubeMapsPl...
assertTable
assertElementPresent //div[@id='gb']/div/div
verifyTitle Google
verifyValue
verifyText //div[@id='gb']/div/div +YouGmailImages+YouSearchYouTubeMapsPla...
verifyTable
verifyElementPresent //div[@id='gb']/div/div
waitForTitle Google
waitForValue
waitForText //div[@id='gb']/div/div +YouGmailImages+YouSearchYouTubeMaps...
waitForTable
waitForElementPresent //div[@id='gb']/div/div
storeTitle Google
storeValue
storeText //div[@id='gb']/div/div +YouGmailImages+YouSearchYouTubeMapsPla...
storeTable
storeElementPresent //div[@id='gb']/div/div

Odtwarzanie WebDriver

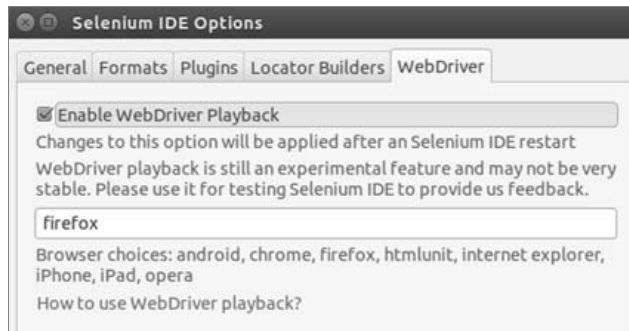
Funkcjonalność odtwarzania WebDriver w Selenium IDE pozwala na uruchamianie testów na dowolnej z najpopularniejszych przeglądarek: Chrome, Firefox, HtmlUnit, Internet Explorer i Opera. Domyślnie funkcjonalność odtwarzania WebDriver jest wyłączona i nieaktywna. Aby uruchamiać skrypty Selenium IDE, musisz włączyć ustawienia odtwarzania WebDriver.

Uruchom Selenium IDE i wybierz opcję *Options...* z menu *Options*. Przejdź do zakładki *WebDriver* i zaznacz pole *Enable WebDriver*. Teraz ponownie uruchom Selenium IDE, aby włączyć funkcjonalność odtwarzania WebDriver. Po zmianie nazwy przeglądarki ponowne uruchomienie Selenium IDE nie jest konieczne. Omawiane tu kwestie zostały przedstawione na zrzucie ekranowym na następnej stronie.

Warunki odtwarzania WebDriver

Poniżej znajdują się wymagania, które muszą być spełnione, aby móc włączyć funkcjonalność odtwarzania WebDriver.

- Pobranie najnowszej wersji biblioteki (JAR) Selenium Server.
- Instalacja Javy do uruchomienia serwera Selenium Server.
- Pobranie najnowszych sterowników dla popularnych przeglądarek (*chromedriver*, *IEDriver* itd.).

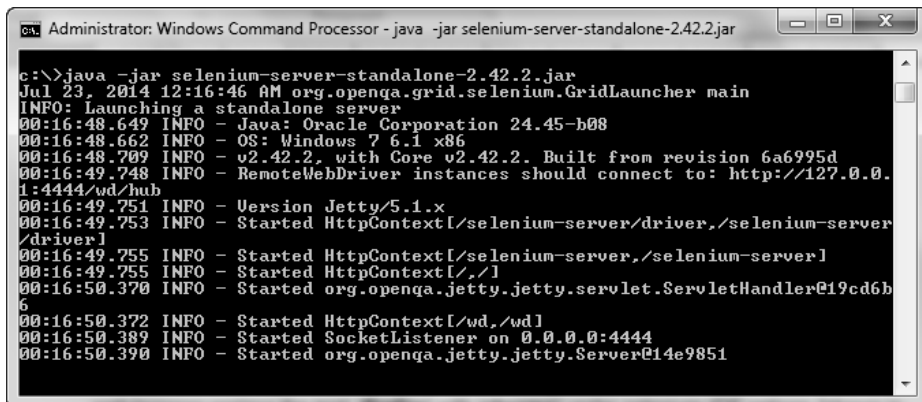


Selenium Server może być uruchomiony ręcznie za pomocą terminala lub wiersza poleceń. Otwórz terminal lub wiersz poleceń, znajdź plik JAR Selenium Server i uruchom polecenie:

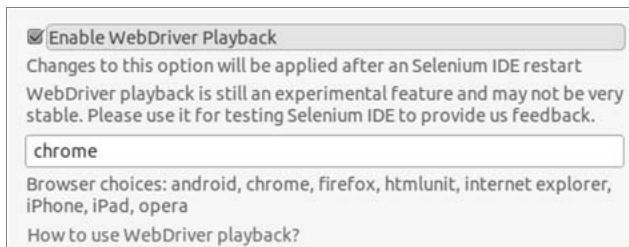
```
java -jar selenium-server-standalone-<version-number>.jar
```

Teraz możesz uruchomić polecenie:

```
java -jar selenium-server-standalone-2.44.0.jar
```



Aby uruchomić testy za pomocą WebDriver, kliknij przycisk *Play* w Selenium IDE. Aby uruchamiać testy w przeglądarce Chrome, zamień tekst *firefox* na *chrome* w opcjach Selenium IDE, zgodnie z poniższym zrzutem ekranowym.



Konieczne jest ustawienie ścieżki ChromeDriver dla Twojej stacji roboczej. Najnowszy sterownik ChromeDriver możesz pobrać pod adresem <http://chromedriver.storage.googleapis.com/index.html?path=2.9/>.

Poniżej podano czynności pozwalające ustawić ścieżkę rozszerzenia ChromeDriver na różnych platformach.

Na komputerze z systemem Windows:

1. Kliknij dwukrotnie i otwórz okno *Ten komputer*.
2. Kliknij prawym przyciskiem w dowolnym miejscu okna i wybierz *Właściwości*.
3. Kliknij *Zaawansowane ustawienia systemu*.
4. Kliknij przycisk *Zmienne środowiskowe* na zakładce *Zaawansowane*.
5. W sekcji *Zmienne systemowe* wybierz zmienną o nazwie Path i kliknij przycisk *Edytuj*.
6. Rozpakuj teraz pobrany plik *ChromeDriver* i skopiuj ścieżkę lokalizacji.
7. Wklej skopiowaną ścieżkę do zmiennej Path (w sekcji *Zmienne systemowe*) i kliknij przycisk *OK*.

Na komputerze z systemem Linux:

Otwórz terminal i uruchom polecenia:

```
$ wget http://chromedriver.storage.googleapis.com/2.7/chromedriver_linux64.zip
$ Unzip chromedriver_linux64.zip
$ cp chromedriver /usr/local/bin
$ chmod +x /usr/local/bin/chromedriver
```

Na komputerze Mac:

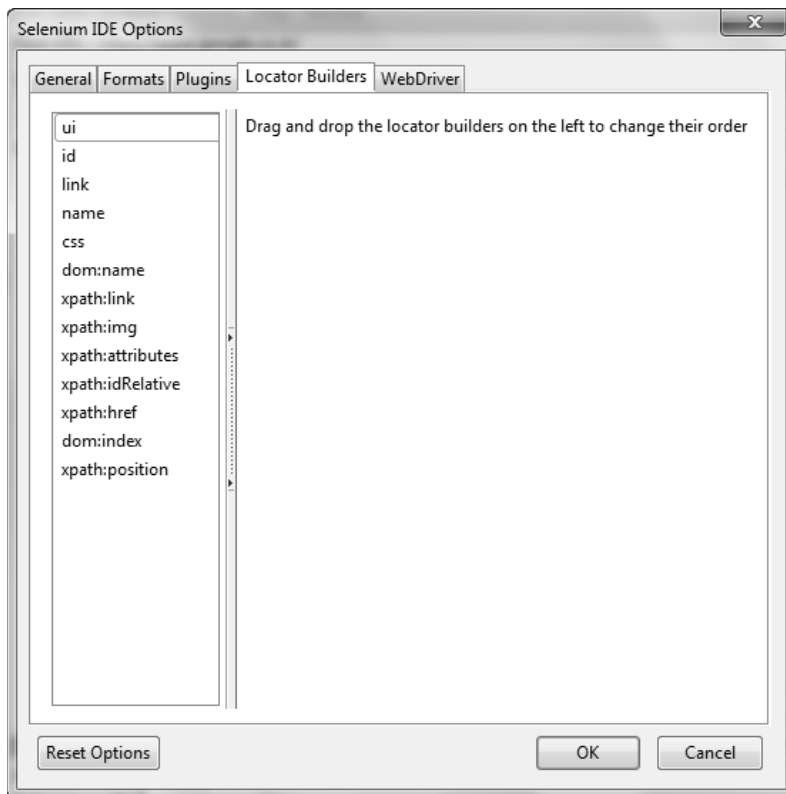
1. Rozpakuj spakowaną paczkę (chromedriver_mac32.zip).
2. Przekopiuj sterownik do katalogu */usr/bin*.
3. Zostaniesz poproszony o podanie hasła administratora; wprowadź je, aby ustawić ścieżkę.

Priorytety lokatorów

Priorytetowanie pozwala ustalać priorytety lokatorów podczas nagrywania skryptów. Funkcjonalność ta pozwala ustawić wysoki priorytet do generowania skryptów zgodnie z preferencjami użytkownika odnośnie do lokatorów. Na przykład zmieniając kolejność *css locator* z piątej na pierwszą pozycję, sprawimy, że elementy będą tworzone za pomocą CSS, czyli zmienna *locatorType* będzie domyślnie ustawiona na CSS.

Przykładem może być `CSS = argument`.

Uruchom Selenium IDE, wybierz *Options...* z menu *Options* i przejdź do zakładki *Locator Builders*. Panel po lewej będzie zawierał listę dostępnych kreatorów lokatorów, takich jak *ui*, *id*, *link*, *name*, *css*, *dom:name*, *xpath:link*, *xpath:img*, *xpath:attributes*, *xpath:idRelative*, *xpath:href*, *dom:index* i *xpath:position*. Lista ta została przedstawiona na poniższym zrzucie.



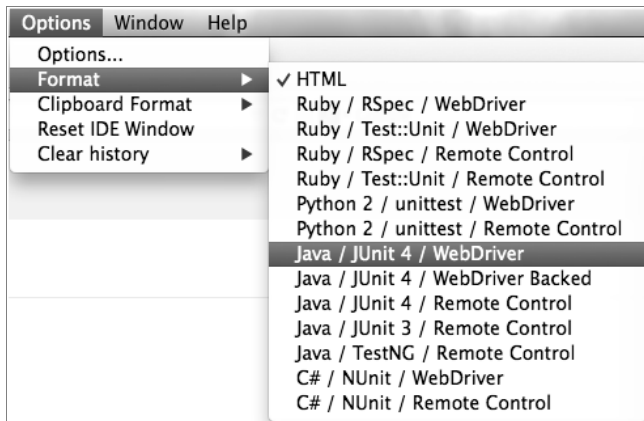
Aby zmienić kolejność konstruktorów lokatorów, możesz je przeciągać. Możesz teraz kliknąć przycisk *OK* i zrestartować Selenium IDE, aby zmiany weszły w życie. Aby wrócić do domyślnych ustawień Selenium IDE, kliknij przycisk *Reset Options* znajdujący się w lewym dolnym rogu okna.

Unikanie eksportu Selenium

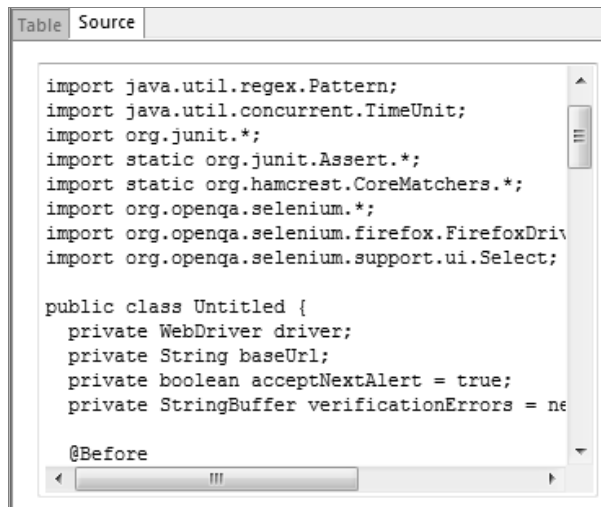
Eksportowanie przypadków testowych za każdym razem może przeszkadzać użytkownikowi. Selenium IDE zawiera świetną funkcję pozwalającą uniknąć problemów z eksportem. Kliknięcie przycisku *Source* pod panelem przypadku testowego wyświetli aktualny przypadek testowy w języku *Selenese*. Selenium IDE przekształca przypadek testowy z języka *Selenese* na preferowany przez użytkownika format, na przykład, *Java/JUnit4/WebDriver*.

Uruchom Selenium IDE i wybierz *Options...* z menu *Options*. Upewnij się, że opcja *Enable experimental features* (włącz funkcje eksperymentalne) jest włączona, i kliknij przycisk *OK*. Kliknij opcję *Format* w menu *Options* i wybierz preferowany format.

Możesz wybrać na przykład *Java/JUnit4/WebDriver*, zgodnie ze zrzutem ekranowym poniżej. Aby zmiany zostały wprowadzone, musisz zrestartować Selenium IDE. Selenium IDE nie wspiera eksportu w TestNG z wykorzystaniem WebDriver (*Java/JUnit4/WebDriver*).

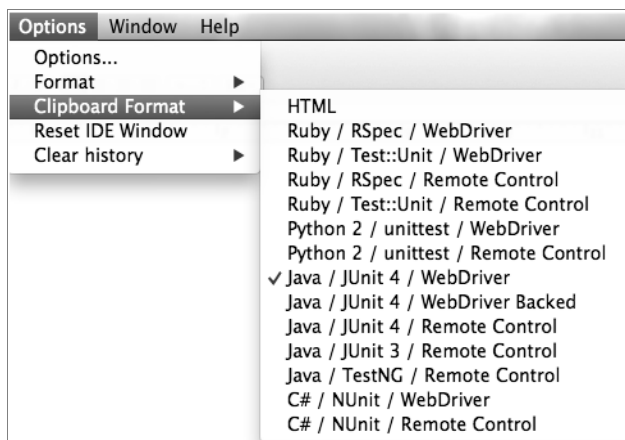


Po wyłączeniu funkcji *Table* zakładka automatycznie zmieni się na widok *Source* (źródła), zgodnie z poniższym zrzutem ekranowym.



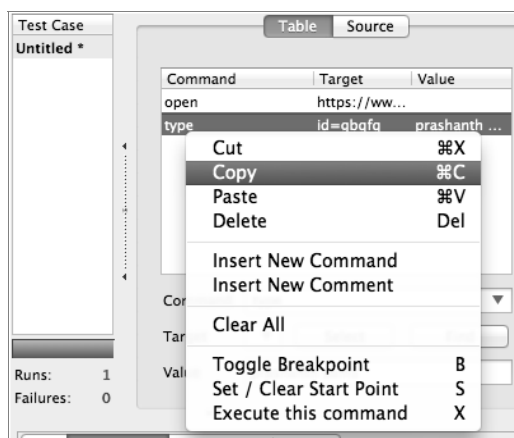
Schówek Selenium IDE

Kopiowanie fragmentów kodu za pośrednictwem *Clipboard Format* (schówka) jest jednym z najszybszych sposobów pozyskania wygenerowanych skryptów. Fragment kodu może zawierać jedną lub dwie linie kodu. Poniższy zrzut ekranowy wyświetla różne typy formatów eksportu dostępne w opcji *Clipboard Format*.



Uruchom Selenium IDE, najedź myszą na opcję *Clipboard Format* w menu *Options* i wybierz preferowaną kombinację formatów.

Przykładem kombinacji formatów jest *Java/JUnit4/WebDriver*. Domyślnym formatem jest format HTML. Poniższy zrzut ekranowy przedstawia kopiowanie wiersza z panelu *Test Case*.



Skopiuj wiersz z panelu *Test Case* (zgodnie z powyższym zrzutem ekranowym) i wklej jako fragment kodu.

```
driver.findElement(By.id("gbqfq")).clear();
driver.findElement(By.id("gbqfq")).sendKeys("prashanthsams");
```

Testy sterowane danymi

Parametryzacja to część techniki sterowania danymi. Pozwala pozyskać wartości wsadowe ze źródła zewnętrznego. Testy sterowane danymi są wykorzystywane do weryfikacji faktycznych i oczekiwanych wartości ze źródła zewnętrznego. Selenium IDE odgrywa znaczącą rolę w parametryzacji, ponieważ operuje na różnych zestawach permutacji i kombinacji. Zobaczmy, jak wykorzystać plik JavaScript jako źródło danych dla testów sterowanych danymi. Poniżej przedstawiona została składnia JavaScript dla parametryzacji.

```
varname = "wartość"
```

Na przykład utwórz plik JavaScript (*Datasource.js*) zawierający następujące słowa kluczowe:

```
Search1 = "PrashanthSams"
Search2 = "Selenium Podstawy"
```

Uruchom Selenium IDE i wybierz *Options...* z menu *Options, Option/Options...* Przejdź teraz do pola *Selenium IDE extensions* (rozszerzenia Selenium IDE) i wybierz plik *.js* utworzony wcześniej (*Datasource.js*), zgodnie z pierwszym zrzutem ekranowym zamieszczonym na następnej stronie.

Uruchom teraz powtórnie Selenium IDE, aby zmiany weszły w życie. Inicjalizujesz, przechowujesz i pobierasz wartości z pliku *.js* przy wykorzystaniu polecenia *storeEval*, zgodnie z drugim zrzutem ekranowym zamieszczonym na następnej stronie.

Search1 i *Search2* to zmienne pobierające odpowiadające im słowa kluczowe z pliku JavaScript. Wartości te są zapisywane powtórnie w nowych zmiennych *GoogleSearch1* i *GoogleSearch2*, zgodnie z drugim zrzutem ekranowym zamieszczonym na następnej stronie.

Metody JavaScript zdefiniowane przez użytkownika

Akcje IDE, akcesory i sprawdzenia mogą być definiowane przez użytkownika i dostosowywane. Aby to osiągnąć, użytkownik powinien dodać metody JavaScript do prototypu obiektu Selenium i prototypu obiektu PageBot. Selenium IDE weryfikuje metody zdefiniowane przez użytkownika przy starcie. Rozszerzenia Selenium Core w menu *Options...* wspierają ładowanie plików JavaScript zdefiniowanych przez użytkowników.

General | Formats | Plugins | Locator Builders | WebDriver

Encoding of test files

Default timeout value of recorded command in milliseconds (30s = 30000ms)

Selenium Core extensions (user-extensions.js)

Selenium IDE extensions

Tips for extensions: Close and reopen Selenium IDE window to make changes effect. You can specify multiple files separated by commas.

Remember base URL

Record assertTitle automatically

Record absolute URL

Activate developer tools

Visual assist (restart of Selenium IDE is required)

Enable experimental features

Disable format change warning messages

Start recording immediately on open

Table | Source

Command	Target	Value
open	/?gfe_rd=cr&ei=djWq...	
storeEval	Search1	GoogleSearch1
type	id=gbqfq	\${GoogleSearch1}
click	id=gbqfb	
open	/?gfe_rd=cr&ei=djWq...	
storeEval	Search2	GoogleSearch2
type	id=gbqfq	\${GoogleSearch2}
click	id=gbqfb	

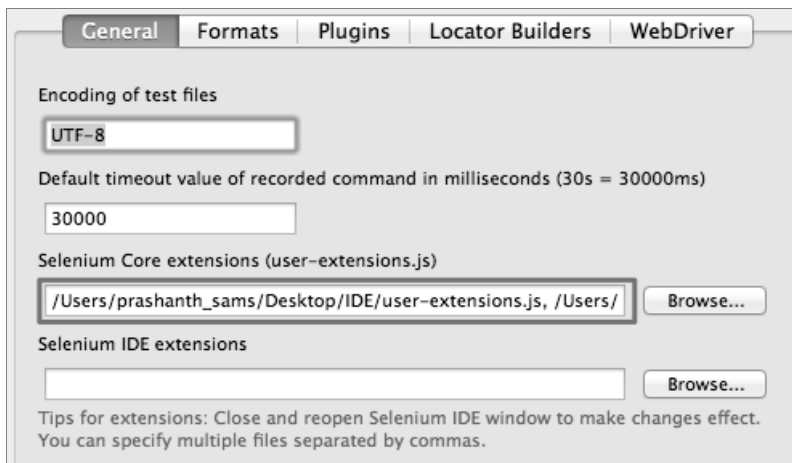
Command

Target

Value

Rozważmy przykład krok po kroku.

1. Przejdź do strony <https://sites.google.com/site/seleniumworks/selenium-ide-data-driven> i pobierz pliki JavaScript:
 - *datadriven.js*
 - *goto_sel_ide.js*
 - *user-extensions.js*
2. Uruchom Selenium IDE i wybierz *Options...* z menu *Options, Option/Options...* Przejdź do pola *Selenium Core extensions* i prześlij pliki (*user-extensions.js*, *goto_sel_ide.js* i *datadriven.js*) zgodnie z poniższym zrzutem ekranowym. Następnie zrestartuj Selenium IDE, aby zmiany weszły w życie.



3. W Selenium IDE plik XML jest wykorzystywany jako źródło danych do przechowywania wartości, natomiast *datadriven.js* został zaprojektowany do wspierania formatu XML.

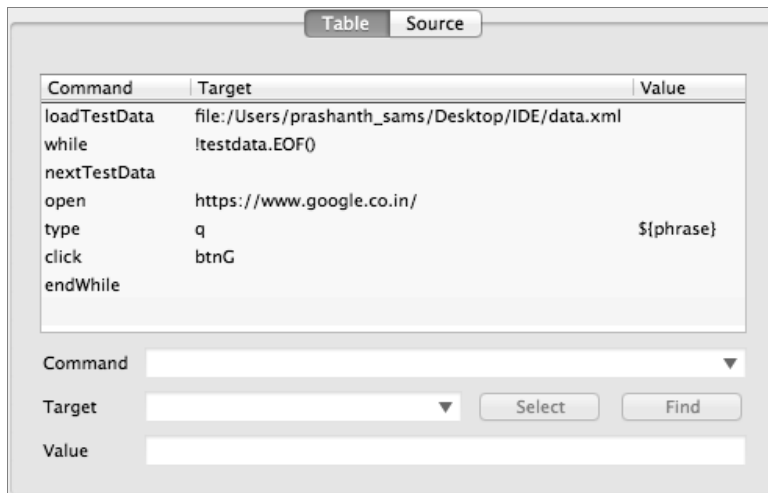
Oto przykład źródła danych w formacie XML:

```
<testdata>
  <test varname="wartość" />
  <test varname="wartość" />
  <test varname="wartość" />
</testdata>
```

Utwórz plik XML z rozszerzeniem *.xml* (*data.xml*). *varname* jest nazwą zmiennej, a wartość odnosi się do słowa kluczowego w znaczniku *<test>*. Przykład takiego pliku:

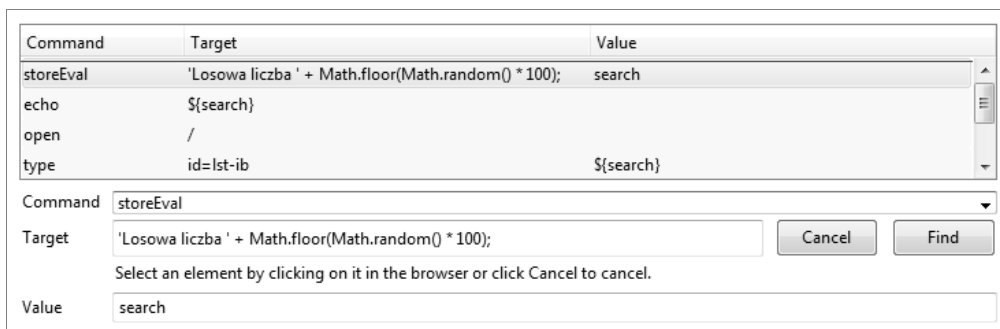
```
<testdata>
  <test phrase=" Selenium Automatyczne testowanie aplikacji " />
  <test phrase="seleniumworks.com" />
  <test phrase="prashantsams" />
</testdata>
```

4. Przyjrzyjmy się dokładnie poniższym poleceniom: `loadTestData` jest poleceniem zdefiniowanym przez użytkownika, pobierającym źródło danych XML, `while` i `endWhile` to pętla, natomiast `nextTestData` sprawdza dane z kolejnego wiersza źródła danych. Użytkownik może dodać dowolną liczbę metod JavaScript. Poniższy zrzut ekranowy przedstawia ten krok.



Funkcje JavaScript w Selenium IDE

Oprócz definiowania poleceń i wprowadzania ich za pośrednictwem *user-extensions.js* Selenium IDE umożliwia również użytkownikom tworzenie zapytań lub funkcji JavaScript bezpośrednio w polu *Target*. Uruchommy na przykład wyszukiwanie Google dla losowej liczby od 1 do 100.



Poniższy kod HTML pozwoli Ci przekonwertować poszczególne kroki na skrypt gotowy do uruchomienia.

```

<tr>
  <td>storeEval</td>
  <td>'Losowa liczba ' + Math.floor(Math.random() * 100);</td>
  <td>search</td>
</tr>
<tr>
  <td>echo</td>
  <td>${search}</td>
  <td></td>
</tr>
<tr>
  <td>open</td>
  <td></td>
  <td></td>
</tr>
<tr>
  <td>type</td>
  <td>id=1st-ib</td>
  <td>${search}</td>
</tr>

```

Proste wywołanie JavaScript

Domyślne polecenie runScript jest bardzo potężne i pozwala wywoływać proste funkcje JavaScript bezpośrednio z IDE, na przykład javascript{alert("Hello!")}.

Zobaczymy, jak możemy wyłączyć aktywne pole tekstowe i włączyć nieaktywne pole tekstowe, korzystając z kodu:

```

document.getElementsByName('****')[0].setAttribute('disabled', '')
document.getElementsByName('****')[0].removeAttribute('disabled');

```

Command	Target	V...
open	https://accounts.google.com/	
runScript	document.getElementsByName("Passwd")[0].setAttribute('disabled', '')	
runScript	document.getElementsByName("Passwd")[0].removeAttribute('disabled');	

Przewijanie kółkiem myszy

Zdarzenie `scroll` jest aktualnie niedostępne w Selenium IDE. Jednak rozszerzenie *user-extensions.js* zawiera metodę JavaScript pozwalającą przewijać stronę internetową.

Przejdź na stronę <https://sites.google.com/site/seleniumworks/selenium-ide-tricks> i pobierz plik *user-extensions.js*. To rozszerzenie zawiera polecenia IDE: `while`, `endWhile`, `gotoIf`, `gotoLabel` i `push`. Zwiększ wartość do 10 zależnie od pionowej długości strony, zgodnie z poniższym zrzutem ekranowym.

Command	Target	Value
open	https://accounts.google.com/	
store	20	i
store	0	looptimes
while	storedVars.looptimes <= 10	
storeEval	selenium.browserbot.getCurrentWindow().scrollTo(0,\$(i))	
store	javascript{storedVars.looptimes++;}	
storeEval	\$(i)+20	i
endWhile		

Parametryzacja przy wykorzystaniu tablic

Polecenie Selenium IDE `storeEval` jest wykorzystywane do przechowywania wartości w zmiennej podczas uruchamiania skryptów, natomiast `storedVars` to tablica asocjacyjna JavaScript zawierająca wartości indeksowane ciągami znaków. W poniższym przykładzie `storeEval` zapisuje w tablicy listę rzek, a `storeEval` inicjalizuje i inkrementuje wartości. Niektóre z poleceń wykorzystanych w tej sekcji, na przykład `while` i `endWhile`, są zdefiniowane przez użytkownika. Polecenie `endWhile` jest wykorzystywane do zakończenia pętli, gdy wartość w tablicy osiągnie maksimum. Poniższy zrzut ekranowy przedstawia omawiane polecenia.

Command	Target	Value
storeEval	new Array("Amazonka", "Dunaj", "Eufrat", "Rodan", "Orinoko")	rivers
getEval	i=0;	
while	i < storedVars['rivers'].length	
storeEval	i	data
echo	javascript{storedVars['rivers'][storedVars['data']]}	
getEval	i++;	x
endWhile		

Przyjrzyjmy się innemu przykładowi zaawansowanej parametryzacji przy wykorzystaniu Selenium IDE. Wejdź na stronę <https://sites.google.com/site/seleniumworks/selenium-ide-tricks> i pobierz plik *user-extensions.js*. Poniższy zrzut ekranowy przedstawia esencję naszego omówienia.

Command	Target	Value
setSpeed	400	
getEval	delete storedVars['Array']	
open	http://www.google.com	
push	selenium essentials	Array
push	prashanth sams	Array
push	seleniumworks	Array
storeEval	storedVars['Array'].length	length
store	0	Var
while	storedVars['Var'] < storedVars['length']	
storeEval	storedVars.Array[\${Var}]	selenium
echo	\${selenium}	
type	id=gbqfq	\${selenium}
click	id=gbqfba	
pause	2000	
store	javascript{storedVars.Var++;}	
endWhile		

W tym przykładzie wartości są wstawiane do tablicy ręcznie, a skrypt wykonuje ich wyszukiwanie w Google.

Selenium Builder

Selenium Builder to narzędzie do nagrywania i odtwarzania testów podobne do Selenium IDE i stanowiące rozszerzenie dla przeglądarki Firefox. Posiada pewne unikalne funkcje, których nie posiada Selenium IDE. Na przykład integrację z GitHub pozwalającą eksportować zestawy testów do repozytorium GitHub i je zatwierdzać oraz integrację z TestingBot pozwalającą wykonywać testy w chmurze. Wpiera również więcej języków niż Selenium IDE, włączając w to języki takie jak JSON, Java/TestNG, NodeJS WD, NodeJS Mocha i NodeJS Protractor. Dzięki swoim zaawansowanym funkcjom Selenium Builder jest uważany za przyszłość Selenium IDE. Poniżej znajduje się zrzut ekranowy Selenium Builder.

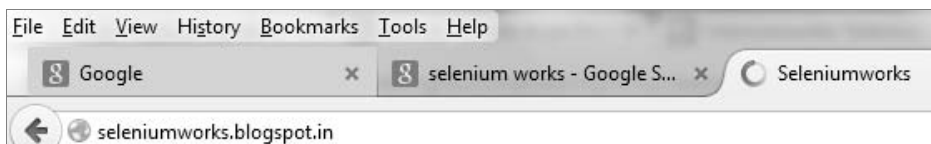


Nagrywanie i odtwarzanie

Po zainstalowaniu rozszerzenia Selenium Builder w przeglądarce Firefox otwórz testowaną stronę internetową (na przykład *www.google.pl*). Jest kilka sposobów na otwarcie Selenium Builder:

- Kliknij prawym przyciskiem na stronie i wybierz *Launch Selenium Builder*.
- Wybierz *Selenium Builder* z menu *Tools*, czyli *Tools/Web Developer/Launch Selenium Builder*.
- Możesz również skorzystać ze skrótu klawiszowego *Ctrl+Alt+B*.

Selenium Builder powinien mieć opcję *Selenium 2* pozwalającą nagrywać skrypty WebDriver. Użytkownik może łatwo stwierdzić, że strona jest nagrywana, ponieważ zakładka jest zaznaczona kolorem zielonym, zgodnie z poniższym zrzutem ekranowym.



Selenium Builder pozwala kontrolować nagrywanie testów aplikacji sieciowej. Pozwala użytkownikowi weryfikować testy przy wykorzystaniu przycisku *Record a verification*. Kliknięcie przycisku podświetla tekst do sprawdzenia po najechaniu na niego kursorem myszy. Funkcja *mouseover* jest świetną funkcją rozszerzenia Selenium Builder pozwalającą wykonać operację najechania na element kursorem myszy. Zaznacz opcję *Record mouseovers*, aby nagrywać akcje najechania na element kursorem myszy. Poniższy zrzut ekranowy przedstawia omawianą funkcję.

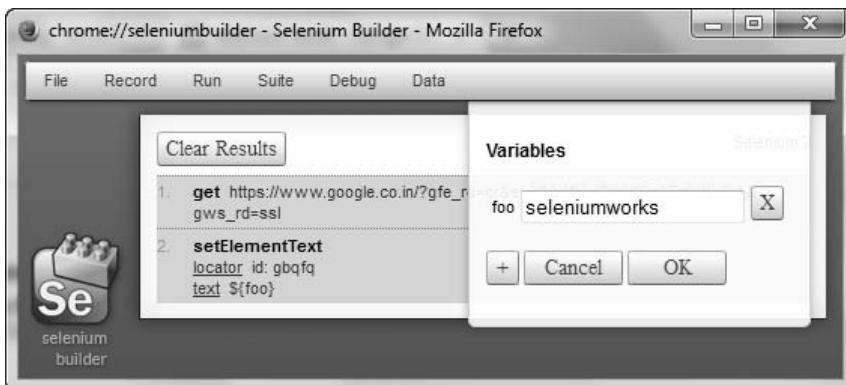


Zatrzymaj nagrywanie skryptu i wyeksportuj go (*File/Export*) w preferowanym formacie.

Testy sterowane danymi

Selenium Builder potrafi importować testy utworzone przy wykorzystaniu Selenium IDE, ponieważ formatem natywnym obu narzędzi jest Selenese. Eksportowanie skryptów Selenium w formacie **Java/TestNG/WebDriver** pozwoli uniknąć dodatkowego wysiłku związanego z pracą z TestNG. TestNG to jeden z najbardziej popularnych frameworków dla testów jednostkowych, jest podobny do JUnit dla programów Java.

Pomimo że Selenium Builder jest narzędziem do nagrywania i odtwarzania, pozwala też użytkownikowi wykonywać podstawowe testy sterowane danymi z wykorzystaniem źródła danych. Wartości mogą być również tymczasowo przechowywane w Selenium Builder przy wykorzystaniu opcji *Manual Entry (Data/Manual Entry)*. Aby to zrobić, utwórz zmienną i przypisz wartość zgodnie z poniższym zrzutem ekranowym.



Selenium Builder zawiera wsparcie dla testów sterowanych danymi i pozwala robić testy przy użyciu plików w formacie JSON, XML i CSV. Format CSV może być wykorzystywany do pracy z plikami zawierającymi duże ilości danych. Przyjrzyjmy się formatom, jakie możesz wykorzystać do tworzenia testów.

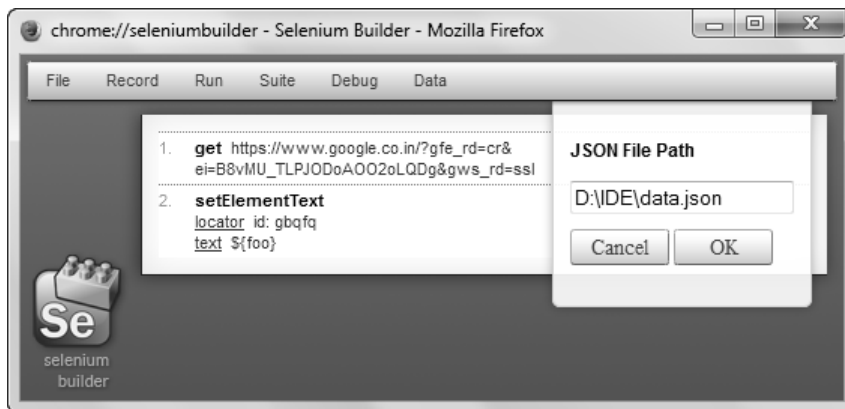
Testowanie przy wykorzystaniu plików JSON

Oto składnia JSON dla źródła danych:

```
[
  { "varname": "wartość", "varname": "wartość", ... },
  { "varname": "wartość", "varname": "wartość", ... },
  ...
]
```

Na przykład utwórz plik JSON z rozszerzeniem *.json* (*data.xml*). W poniższym przykładzie *foo* to nazwa zmiennej, natomiast wartości to *prashanth* i *sams*.

```
[ {"foo": "prashanth"}, {"foo": "sams"} ]
```



Aby odwołać się do zmiennych przechowywanych w pliku JSON, najlepiej odwołać się do zmiennych, na przykład ``${foo}`, we wcześniej nagranych krokach, zgodnie ze zrzutem ekranowym powyżej.

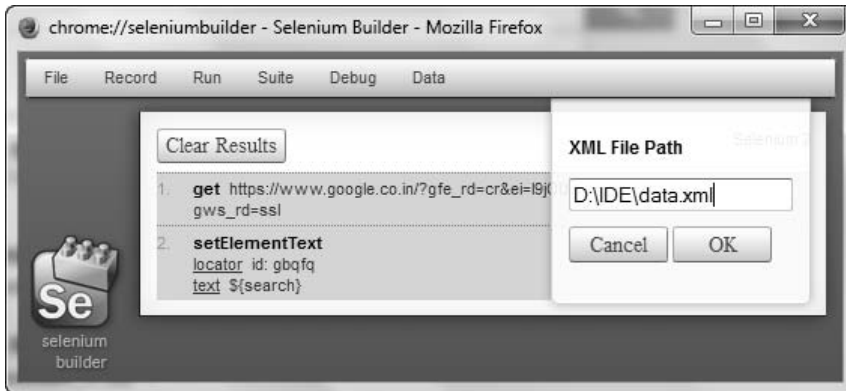
Testowanie przy wykorzystaniu plików XML

Oto składnia XML dla źródła danych:

```
<testdata>
  <testvarname="wartość" />
  <testvarname="wartość" />
  <testvarname="wartość" />
</testdata>
```

Na przykład utwórz plik XML z rozszerzeniem *.xml* (*data.xml*). W poniższym przykładzie search jest nazwą zmiennej, a wartość odnosi się do słowa kluczowego w znaczniku `<test>`.

```
<testdata>
  <test search="selenium automatyczne testowanie aplikacji" />
  <test search="prashanthsams" />
  <test search="seleniumworks" />
</testdata>
```

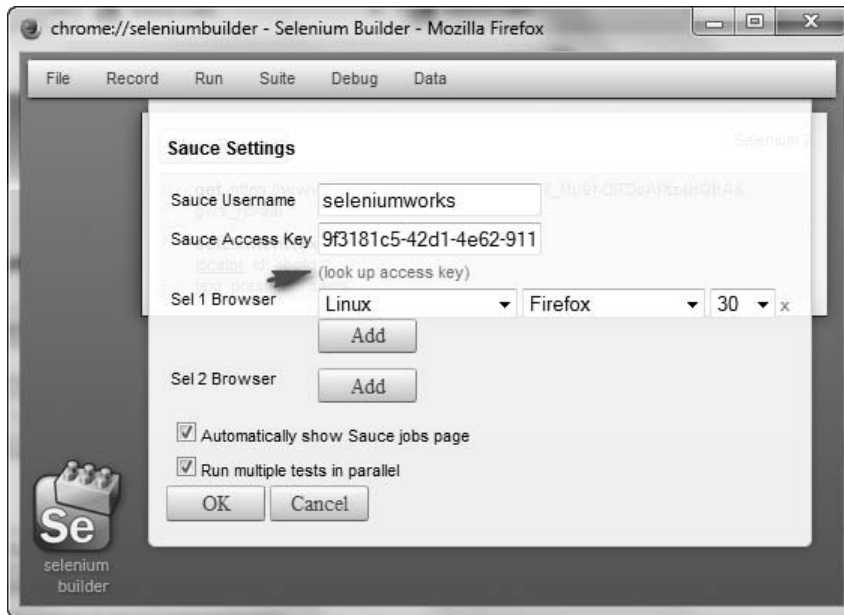


Selenium Builder w chmurze

Selenium Builder pozwala użytkownikom uruchamiać testy na różnych przeglądarkach w chmurze bezpośrednio za pośrednictwem interfejsu. Aby zintegrować Selenium Builder z Sauce i włączyć możliwość eksportowania i odtwarzania skryptów w Sauce OnDemand, konieczna jest instalacja wtyczki Selenium Builder Sauce. Uruchom Selenium Builder, kliknij przycisk *Manage plugins* i zainstaluj wtyczkę Sauce for Selenium Builder. Przed uruchomieniem testów użytkownicy powinni również założyć konto Sauce. Więcej informacji na temat uruchamiania testów w chmurze znajdziesz pod adresem <https://saucelabs.com/>. Poniższy zrzut ekranowy przedstawia stronę Plugins (wtyczek).



Pozyskaj klucz dostępowy po zalogowaniu się do swojego konta Sauce lub klikając *look up access key* w Selenium Builder. Klucz dostępowy jest inny dla każdego użytkownika.



Wybierz *Run on Sauce OnDemand* z menu *Run*. Przed uruchomieniem testów upewnij się, że ustawienia *Sauce Settings* zostały zdefiniowane. Musisz tylko wpisać *Sauce Username* (nazwa użytkownika Sauce), *Sauce Access Key* (klucz dostępowy Sauce) oraz wersje systemów operacyjnych i przeglądarek. Następnie zaloguj się do Sauce i zapoznaj się z wynikami testów. Testy są zapisywane w postaci filmów i zdjęć, które użytkownik może przejrzeć.

Session	Environment	Tags	Build	Results	End
<input type="checkbox"/> Selenium Builder firefox 30 Linux Untitled	A* 30		Pass	Jul 21 2014 12:54:57	31s

Showing page 1 of 1 | Load more jobs...

Podsumowanie

W tym rozdziale poznałeś funkcje Selenium IDE i Selenium Builder oraz dowiedziałeś się, jak dzięki Selenium IDE automatyzować proste testy.

W kolejnym rozdziale omówimy zaawansowane testy kompatybilności z wykorzystaniem Selenium WebDriver. WebDriver pozwala wykonywać testy na różnych przeglądarkach.

Skorowidz

A

- absolutny XPath, 56
- adnotacja
 - @dataProvider, 134, 146, 151
 - @FindBy, 92
 - @FindBy, 93
- Ajax, 82
- akcje myszy i klawiatury, 74
- Apache POI, 136
- Apple Safari, 37
- arkusz SpreadsheetML, 141
- arkusze binarne, 138, 143
- ATDD, Acceptance-Driven Development, 118
- autentykacja, 74
- automatyzacja testów, 40

B

- bezinterfejsowy WebKit, 45
- biblioteka
 - Java.awt.Robot, 105
 - JExcel, 131
 - JSErrorCollector, 114
- błędy JavaScriptu, 114
- BrowserStack, 42

C

- chmura, 29, 40
- Chrome, 49
- ciasteczka, 64
- CSV, Comma Separated Values, 153
- Cucumber JVM, 119

E

- eksport Selenium, 16
- Excel, 127

F

- Firefox, 49
- framework, 117
 - BDD Cucumber, 118
 - BDD JBehave, 123
 - hybrydowy, 157
 - sterowany danymi, 136, 144, 148, 153
 - sterowany danymi JXL API, 127
 - sterowany słowami kluczowymi, 155
- funkcja
 - accept(), 73
 - clear(), 55
 - click(), 54
 - close(), 54
 - dismiss(), 73
 - dragAndDrop(), 75
 - dragAndDropBy(), 76
 - findElement(), 56
 - findElements(), 57
 - get(), 63
 - getAttribute(), 60
 - getCurrentUrl(), 54
 - getLocation(), 62
 - getPageSource(), 54
 - getSize(), 62
 - getTagName(), 61
 - getText(), 60, 73
 - getTitle(), 54

funkcja

- isDisplayed(), 61
- isEnabled(), 61
- isSelected(), 61
- navigate().back(), 63
- navigate().forward(), 63
- navigate().to(), 63
- quit(), 54
- sendKeys(), 55
- submit(), 55

funkcje

- elementów WebElements, 60
- JavaScript, 22
- lokalizujące elementy, 55
- okna, 66
- WebDriver, 54
- wybierające, 70

G

Google Chrome, 36

H

HSSF, Horrible Spreadsheet Format, 138
HTMLUnitDriver, 47

I

IDE, Integrated Development Environment, 9
interfejs użytkownika, 48
Internet Explorer, 36

J

Java Robot, 105
JExcel, 131

K

klasa

- EventFiringWebDriver, 93
- FileWriter, 153
- JavascriptExecutor, 110
- PageFactory, 90

klucz klienta, 43

kolektor błędów JavaScript, 114

L

limity czasu, 87
lokalizowanie elementów, 55

M

metoda

- addCookie(), 65
- afterChangeValueOf(), 97
- afterClickOn(), 96
- afterFindBy(), 96
- afterNavigateBack(), 94
- afterNavigateForward(), 94
- afterNavigateTo(), 95
- afterScript(), 97
- authenticateUsing(), 74
- beforeChangeValueOf(), 98
- beforeClickOn(), 96
- beforeFindBy(), 97
- beforeNavigateBack(), 95
- beforeNavigateForward(), 95
- beforeNavigateTo(), 95
- beforeScript(), 97
- build(), 74
- click(), 74
- clickAndHold(), 75
- contextClick(), 75
- deleteAllCookies(), 65
- deleteCookie(), 65
- deleteCookieNamed(), 66
- deselectAll(), 72
- deselectByIndex(index), 72
- deselectByValue(value), 72
- deselectByVisibleText(text), 72
- doubleClick(), 75
- FluentWait, 86
- getAllSelectedOptions(), 71
- getContents(), 128
- getCookieNamed(), 65
- getCookies(), 64
- getFirstSelectedOption(), 70
- getOptions(), 71
- getPosition(), 67
- getSize(), 67
- getWindowHandle(), 68
- getWindowHandles(), 68
- implicitlyWait(), 87

isElementPresent(), 82
 isMultiple(), 71
 keyDown(), 76
 keyUp(), 76
 maximize(), 66
 moveByOffset(), 76
 moveToElement(), 77
 onException(), 98
 pageLoadTimeout(), 87
 perform(), 77
 refresh(), 64
 release(), 77
 selectByIndex(index), 70
 selectByValue(value), 70
 selectByVisibleText(text), 70
 sendKeys(), 74, 77
 setPosition(), 68
 setScriptTimeout(), 87
 setSize(), 68
 switchTo.window(), 68
 write(), 128
 metody własne JavaScript, 19
 model

- HSSF, 138
- SS, 143
- XSSF, 141

 Mozilla Firefox, 36

N

nagrywanie, 26
 najechanie myszą, 79
 najlepsze praktyki, 81
 nawigacja, 63

O

obsługa

- alertów, 72
- oczekiwania jawnego, 84, 85
- okien wyskakujących, 72, 105
- ramek iframe, 103
- stron wykorzystujących Ajax, 82
- testów, 69

 oczekiwanie, 83

- jawne, 84

 odczyt arkusza Excela, 127
 oddalenie strony, 78

odtwarzanie, 26

- WebDriver, 13

 okna wyskakujące przeglądarki, 106
 Opera, 37
 otwieranie nowego okna, 113

P

parametryzacja, 24
 PhantomJS, 45
 PhantomJSDriver, 46
 plik

- testng.xml, 38
- testNG.xml, 141

 pliki

- .properties, 148
- JSON, 28
- tekstowe, 144
- właściwości, 148
- XML, 28

 pobieranie plików, 109
 podświetlanie elementów, 112
 POI, Poor Obfuscation Implementation, 136
 polecenia

- akcji, 11
- oczekiwania, 83

 priorytety lokatorów, 15
 profil Firefox, 109
 programowanie

- sterowane testami akceptacyjnymi, 118
- sterowane zachowaniem, 118

 przechwytywanie zrzutów ekranowych, 107
 przeglądarka, 36

- Chrome, 36, 49
- Firefox, 36, 49
- Internet Explorer, 36
- Opera, 37
- Safari, 37

 przeglądarki bezinterfejsowe, 45, 47
 przewijanie kółkiem myszy, 24
 przybliżenie

- 100%, 78
- strony, 78

R

ramki iframe, 103
 relatywny XPath, 56

S

SauceLabs, 40
 schowek Selenium IDE, 18
 sekret klienta, 43
 Selenium Builder, 9, 25
 w chmurze, 29
 Selenium IDE, 9
 Selenium WebDriver, 9, 35, 36
 skrolowanie strony, 111
 słowa kluczowe, 155
 sprawdzenia, 11
 sterownik uruchamiający zdarzenia, 99

T

tablice, 24
 TDD, Test-Driven Development, 118
 TestingBot, 43
 TestNG, 37, 134, 146, 151
 testowanie sterowane danymi, 131, 134
 testy
 kompatybilności, 36
 na konkretnych wersjach przeglądarki Firefox,
 50
 na wielu przeglądarkach, 35
 Selenium w chmurze, 40
 sterowane danymi, 19, 27, 129, 146, 151
 w przeglądarce bezinterfejsowej, 45
 z niestandardowego profilu
 Chrome, 52
 Firefox, 51
 tryb
 assert, 11
 verify, 11
 waitFor, 11

U

unikanie eksportu Selenium, 16
 usypianie wątku, 86

W

warunki odtwarzania WebDriver, 13
 WebDriver, 13
 wykonywanie testów w chmurze, 42
 wykorzystanie TestNG, 134, 146, 151
 wywołanie JavaScript, 23
 wzorzec
 Page Object, 88
 regex, 122

X

XSSF, XML SpreadSheet Format, 141

Z

zapis arkusza Excela, 127
 zdarzenia, 99
 zmienianie interfejsów użytkownika, 48
 zrzut ekranowy, 107

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

Selenium Automatyczne testowanie aplikacji

Selenium WebDriver jest udostępnianym na zasadach *open source* narzędziem, które służy do automatyzacji aplikacji sieciowych. Oprogramowanie Selenium jest niezależne od systemu operacyjnego i może być obsługiwane w dowolnym z popularnych języków programowania. Historia tego narzędzia rozpoczyna się w 2004 roku. Dziś Selenium WebDriver jest najczęściej wykorzystywanym narzędziem do automatyzacji testów, cenionym za wszechstronność, elastyczność i łatwość użytkowania.

Ten przewodnik jest przeznaczony dla osób posiadających już pewną wiedzę o automatyzacji testów aplikacji. Zawarto tu niezbędne informacje o środowisku Selenium IDE oraz o sposobach jego wykorzystania do automatyzacji testów i debugowania rozbudowanych aplikacji. Przedstawiono narzędzie Selenium WebDriver, dzięki któremu można wykonywać testy automatyczne na wielu przeglądarkach. Opisano bardziej złożone mechanizmy uruchamiania takich testów w chmurze i omówiono funkcje udostępniane przez Selenium WebDriver API, które mogą zostać wykorzystane do tworzenia efektywnych testów automatycznych. Zaprezentowano również techniki tworzenia testów sterowanych danymi i zachowaniem aplikacji.

Testowanie aplikacji? Z Selenium wykonasz je sprawnie i efektywnie!



W książce znajdziesz:

- zasady korzystania z oprogramowania Selenium IDE i Selenium Builder
- sposoby wykonywania efektywnych testów kompatybilności
- funkcje Selenium WebDriver wraz z ich omówieniem i przykładami
- różne techniki zarządzania zadaniami automatycznymi Selenium
- frameworki, sposoby ich dostosowania i tworzenia nowych na podstawie Selenium WebDriver

Prashanth Sams jest inżynierem automatyzacji testów działającym w branży IT od 2011 roku. Zdobył bogate doświadczenie, realizując wiele projektów przy użyciu różnych narzędzi do automatyzacji. Uwielbia nowe technologie, a przy tym chętnie dzieli się wiedzą, aktywnie wspierając społeczności programistów i testerów, szczególnie oprogramowania Selenium.

PACKT open source
PUBLISHING community experience distilled

Helion

księgarnia internetowa



<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

Informatyka w najlepszym wydaniu

Helion SA
ul. Kościuszki 1c, 44-100 Gilwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
<http://helion.pl>

Sprawdź najnowsze promocje:
● <http://helion.pl/promocje>
Książki najchętniej czytane:
● <http://helion.pl/bestsellery>
Zamów informacje o nowościach:
● <http://helion.pl/nowosci>

sięgnij po WIĘCEJ



KOD KORZYSCI

ISBN 978-83-283-3039-9



cena: 39,90 zł