Technologia i rozwiązania

Selenium
Automatyczne testowanie aplikacji

Prashanth Sams

Helion

[PACKT] open source
community experience distilled
PUBLISHING

Tytuł oryginału: Selenium Essentials

Tłumaczenie: Jakub Hubisz

ISBN: 978-83-283-3039-9

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.

Spis treści

1

ROZDZIAŁ

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`.

Akcesory 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
open	/?gfe_rd=cr&ei=...	
store	prashanth sams	search
type	css=#gbqfq	${search}

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:

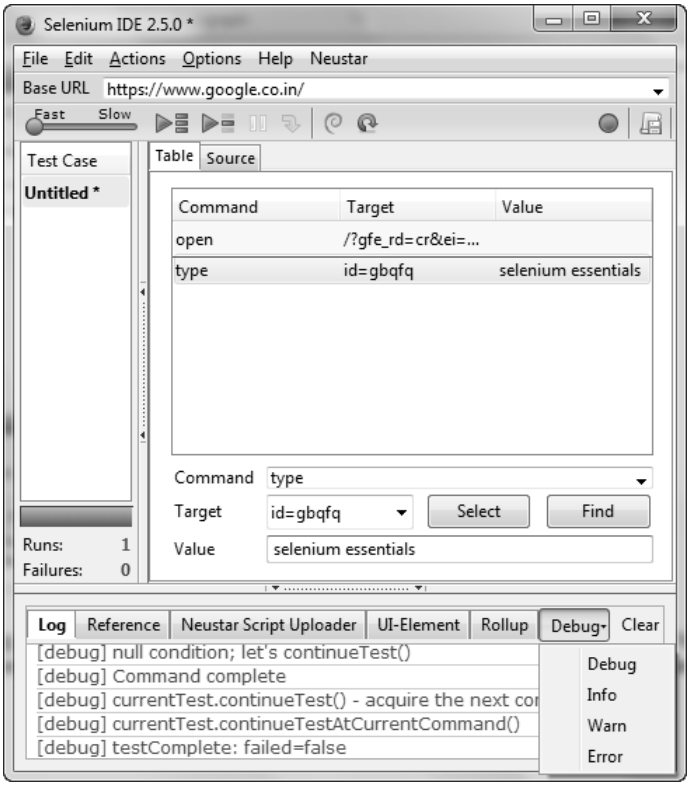
```
locatorType = argument
```

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.

Odtwarzanie WebDriver

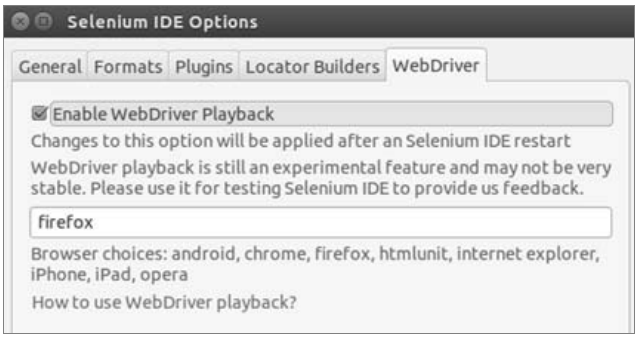
Funkcjonalność odtwarzania WebDriver w Selenium IDE pozwala na uruchamianie testów na dowolnej z najpopularniejszych przeglądarek: Chrome, Firefox, HtmlUnit, Internet Explorer i Opera. Domyślnie funkcjonalność odtwarzania WebDriver jest wyłączona i nieaktywna. Aby uruchamiać skrypty Selenium IDE, musisz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

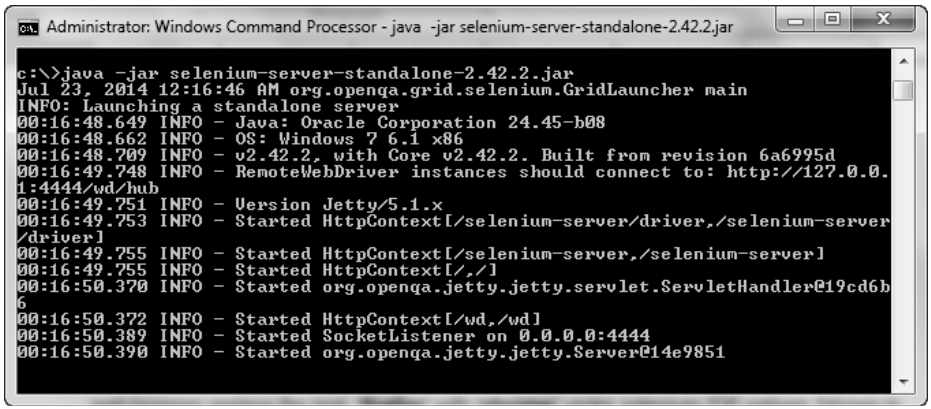


Selenium Server może być uruchomiony ręcznie za pomocą terminala lub wiersza poleceń. Otwórz terminal lub wiersz poleceń, znajdź plik JAR Selenium Server i uruchom polecenie:

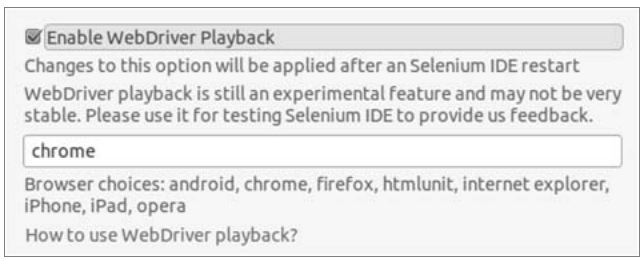
```
java -jar selenium-server-standalone-<version-number>.jar
```

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:

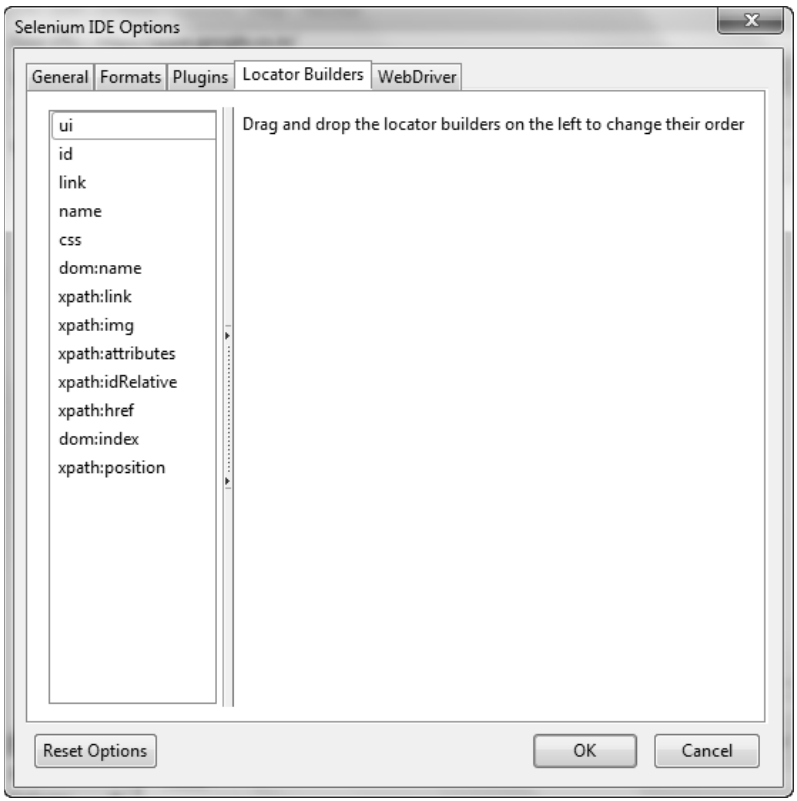
1. Rozpakuj spakowaną paczkę (`chromedriver_mac32.zip`).
2. Przekopiuj sterownik do katalogu */usr/bin*.
3. Zostaniesz poproszony o podanie hasła administratora; wprowadź je, aby ustawić ścieżkę.

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 `csslocator` 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.



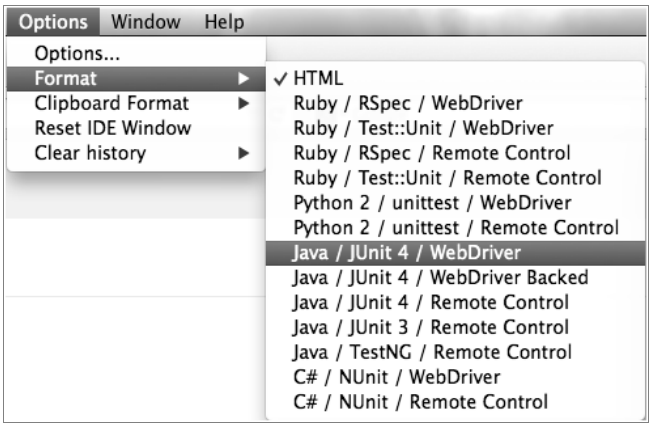
Aby zmienić kolejność konstruktorów lokatorów, możesz je przeciągać. Możesz teraz kliknąć przycisk *OK* i zrestartować Selenium IDE, aby zmiany weszły w życie. Aby wrócić do domyślnych ustawień Selenium IDE, kliknij przycisk *Reset Options* znajdujący się w lewym dolnym rogu okna.

Unikanie eksportu Selenium

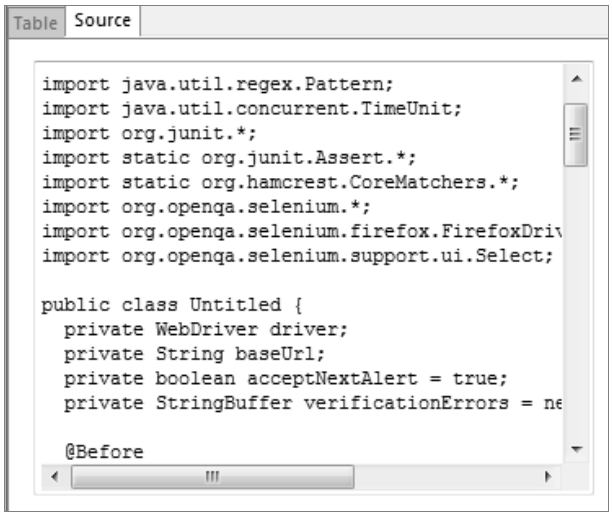
Eksportowanie przypadków testowych za każdym razem może przeszkadzać użytkownikowi. Selenium IDE zawiera świetną funkcję pozwalającą uniknąć problemów z eksportem. Kliknięcie przycisku `Source` pod panelem przypadku testowego wyświetli aktualny przypadek testowy w języku Selenese. Selenium IDE przekształca 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*).

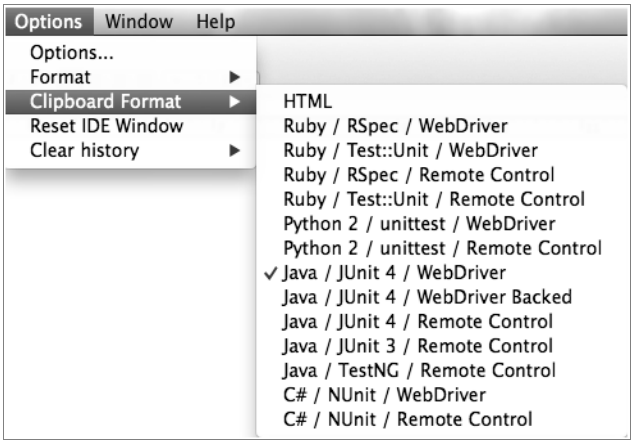


Po wyłączeniu funkcji *Table* zakładka automatycznie zmieni się na widok *Source* (źródła), zgodnie z poniższym zrzutem ekranowym.



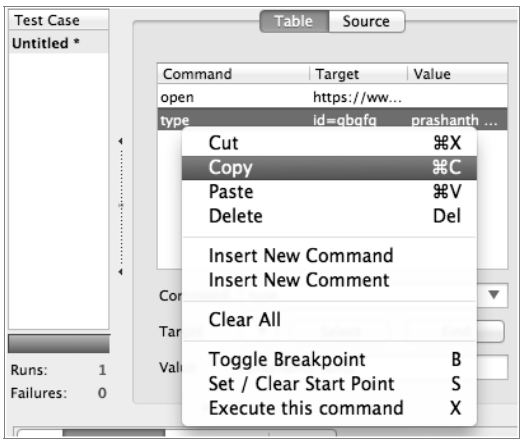
Schowek Selenium IDE

Kopiowanie fragmentów kodu za pośrednictwem ***Clipboard Format*** (schowka) 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"
```

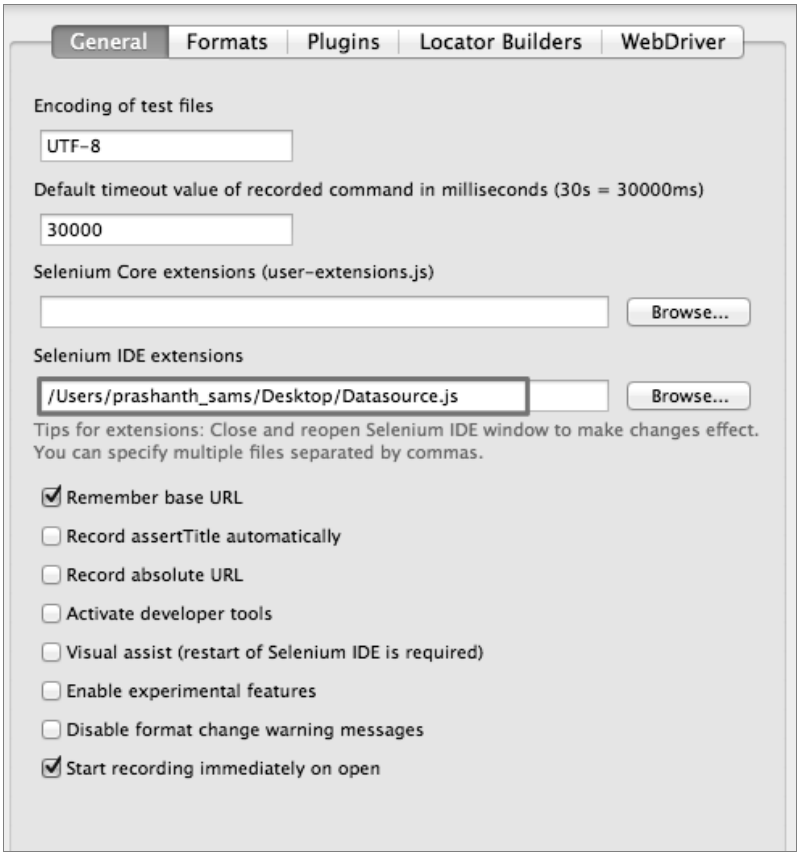
Uruchom Selenium IDE i wybierz *Options…* z menu *Options*, *Option/Options…*. Przejdź teraz do pola *Selenium IDE extensions* (rozszerzenia Selenium IDE) i wybierz plik *.js* utworzony wcześniej (*Datasource.js*), zgodnie z pierwszym zrzutem ekranowym zamieszczonym na następnej stronie.

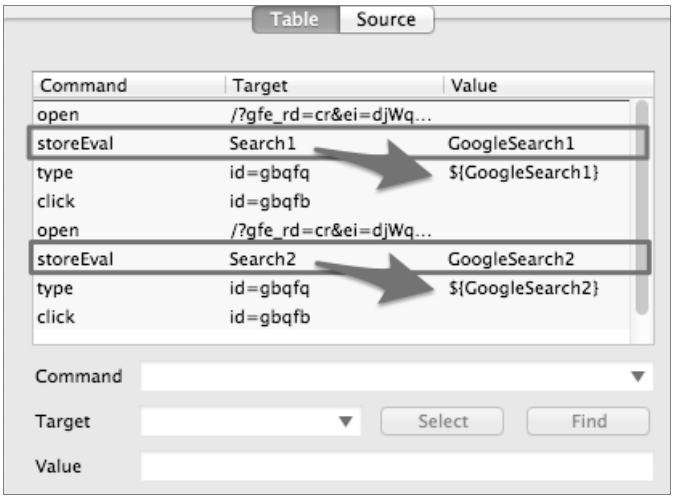
Uruchom teraz powtórnie Selenium IDE, aby zmiany weszły w życie. Inicjalizujesz, przechowujesz i pobierasz wartości z pliku *.js* przy wykorzystaniu polecenia `storeEval`, zgodnie z drugim zrzutem ekranowym zamieszczonym na następnej stronie.

`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

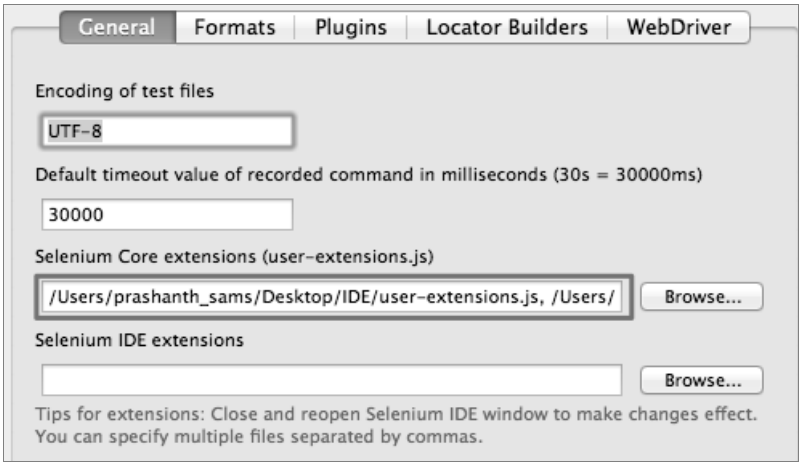
Akcje IDE, akcesory i sprawdzenia mogą być definiowane przez użytkownika i dostosowywane. Aby to osiągnąć, użytkownik powinien dodać metody JavaScript do prototypu obiektu Selenium i prototypu obiektu PageBot. Selenium IDE weryfikuje metody zdefiniowane przez użytkownika przy starcie. Rozszerzenia Selenium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





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.

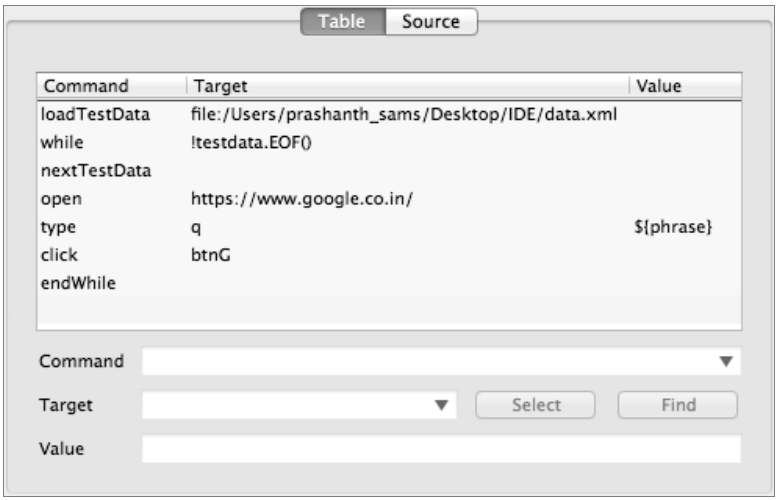
   Oto przykład źródła danych w formacie XML:

```
<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:

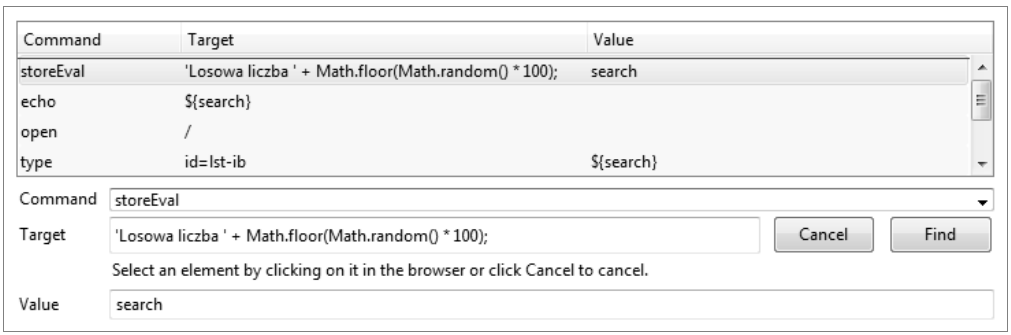
```
<testdata>
  <test phrase=" Selenium Automatyczne testowanie aplikacji " />
  <test phrase="seleniumworks.com" />
  <test phrase="prashanthsams" />
</testdata>
```

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>${search}</td>
  <td></td>
</tr>
<tr>
  <td>open</td>
  <td>/</td>
  <td></td>
</tr>
<tr>
  <td>type</td>
  <td>id=lst-ib</td>
  <td>${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!")}`.

Zobaczmy, 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-exten ↪sions.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		

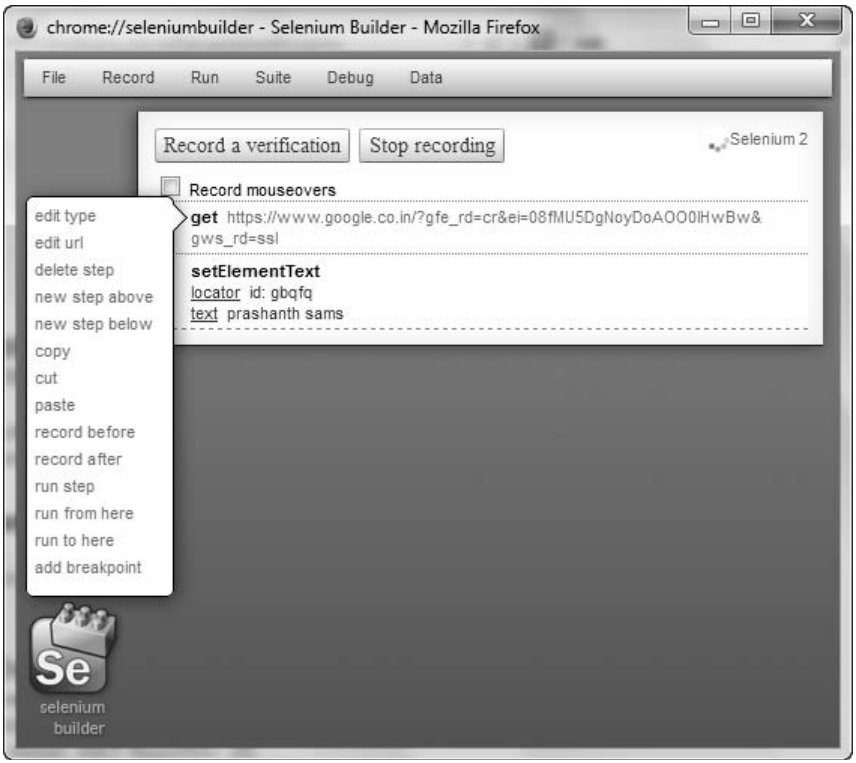
Przyjrzyjmy się innemu przykładowi zaawansowanej parametryzacji przy wykorzystaniu Selenium IDE. Wejdź na stronę *https://sites.google.com/site/seleniumworks/selenium-ide-tricks* i pobierz plik *user-extensions.js*. Poniższy zrzut ekranowy przedstawia esencję naszego omówienia.

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.

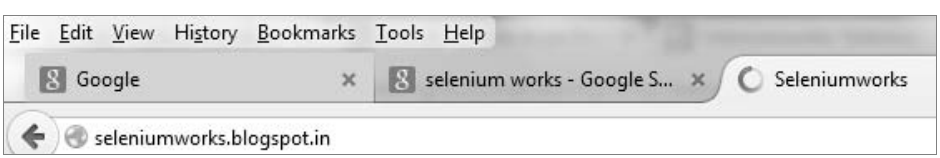


Nagrywanie i odtwarzanie

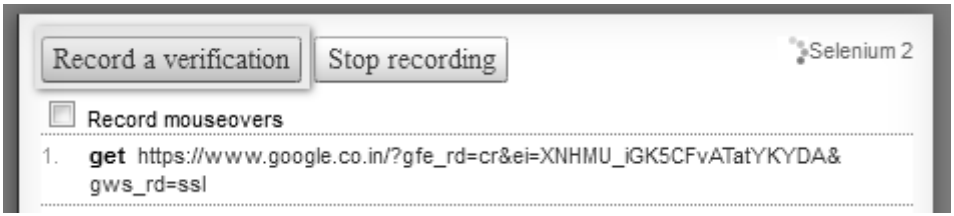
Po zainstalowaniu rozszerzenia Selenium Builder w przeglądarce Firefox otwórz testowaną stronę internetową (na przykład *www.google.pl*). Jest kilka sposobów na otwarcie Selenium Builder:

- 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 użytkownikom 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.

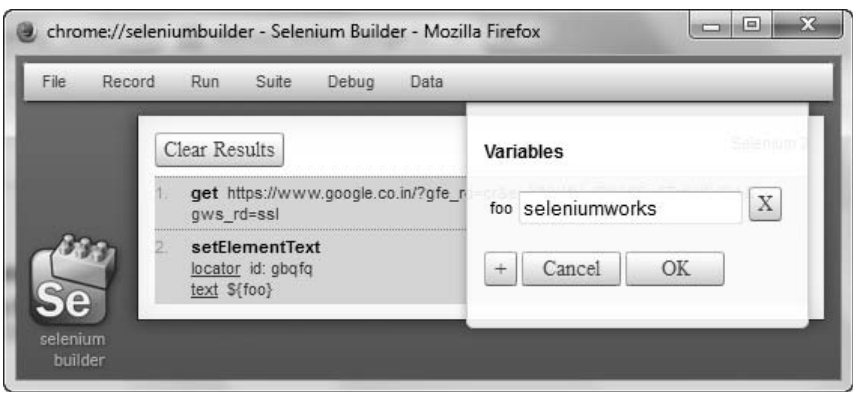


Zatrzymaj nagrywanie skryptu i wyeksportuj go (*File/Export*) w preferowanym formacie.

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 użytkownikom 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.

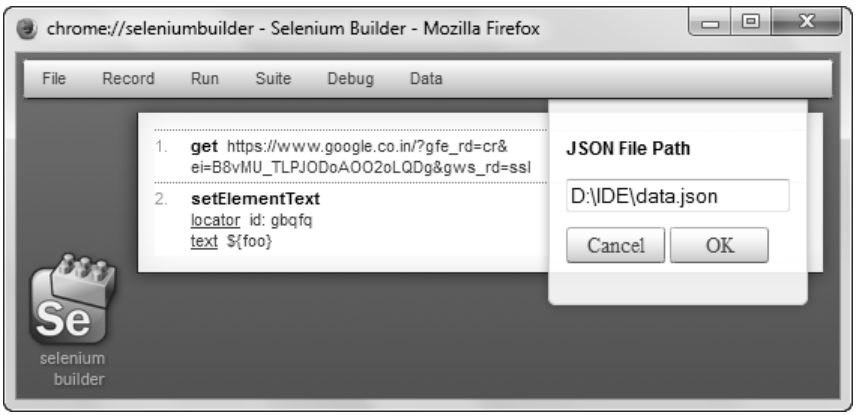
Testowanie przy wykorzystaniu plików JSON

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.

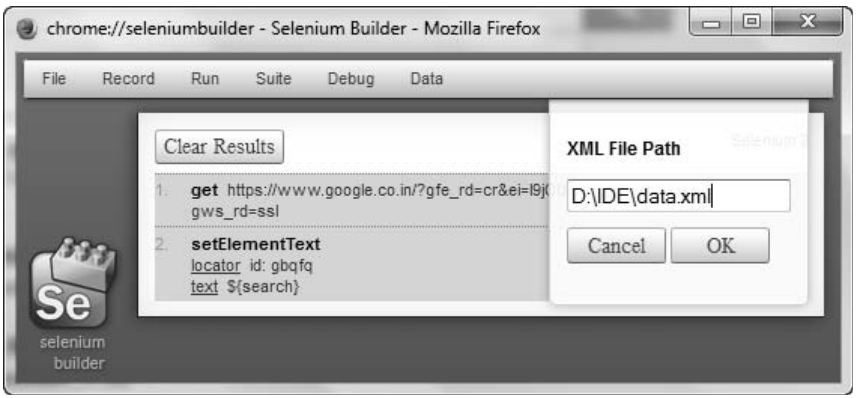
Testowanie przy wykorzystaniu plików XML

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>`.

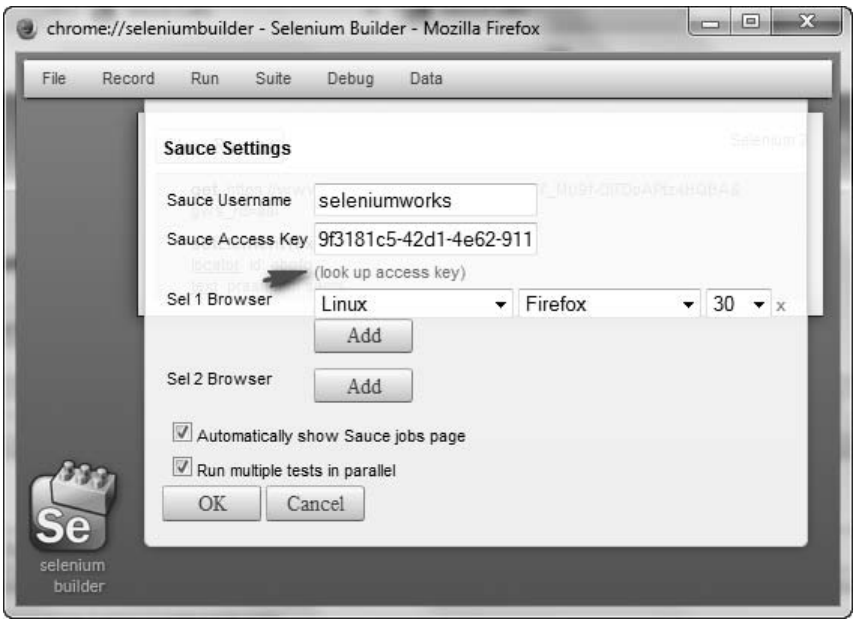
```
<testdata>
  <test search="selenium automatyczne testowanie aplikacji" />
  <test search="prashanthsams" />
  <test search="seleniumworks" />
</testdata>
```



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 meu *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.

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

G

H

I

J

K

L

M

N

O

P

R

S

T

U

W

X

Z

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

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
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

Informatyka w najlepszym wydaniu

sięgnij po WIĘCEJ
KOD KORZYŚCI

ISBN 978-83-283-3039-9

cena: 39,90 zł