SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

TESTOVANIE UŽIVATEĽSKÉHO ROZHRANIA POMOCU FRAMEWORKU SELENIUM

SEMINÁRNA PRÁCA

2016 Martin Kiesel

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

TESTOVANIE UŽIVATEĽSKÉHO ROZHRANIA POMOCU FRAMEWORKU SELENIUM

SEMINÁRNA PRÁCA

Študijný program:	Aplikovaná informatika
Predmet:	I–ASOS – Architektúra softvérových systémov
Prednášajúci:	RNDr. Igor Kossaczký, CSc.
Cvičiaci:	Ing. Eugen Antal

Bratislava 2016 **Martin Kiesel**

Obsah

Zoznam obrázkov a tabuliek

Zoznam skratiek

API Application Programming Interface

IDE Integrated Development Environment

DOM Document object model

Úvod

Automatizované testovanie uživateľského prostredia je dôležitá súčasť procesu vývoja kvalitných internetových aplikácii. Cieľom práce je oboznámenie sa s problematikou automatizovaného testovania a demonštrovanie práce s nástrojom na testovanie (automatizovanie) internetových prehliadačov – Selenium.

1 Automatizované testovanie

Manuálne testovanie je dôležitý no zdĺhavý proces, nakoľko sa jedná o neustále sa opakujúcu sa činnosť. Pre vykonanie manuálneho testu je zakaždým potrebné vynaložiť rovnaké úsilie a nároky na zdroje teda rastú lineárne s počtom vykonávaných testov. Automatické testy sú naproti tomu znovupoužiteľné, je možné ich opakované spúšťanie. Zavádzanie automatických testov sa oproti manuálnemu testovaniu vyznačuje vyššími vstupnými nákladmi spojenými s vytváraním testov, pričom náklady na otestovanie zostavenia aplikácie postupom času klesajú, keďže testovanie sa deje automaticky, typicky po zostavení novej verzie aplikácie [1, 2]. Hlavným cieľom automatizácie testov softvéru je časová úspora pri ich spúšťaní. Všeobecne sa dá povedať, že automatizácia testov softvéru má za účel zjednodušiť a zefektívniť proces testovania softvéru. Pokiaľ sú testy spúštané úplne automaticky – bez možnosti zásahu ľudského faktoru, výrazne sa znižuje možnosť chybného prevedenia postupu testovania softvéru.

1.1 Typy testovania softvéru

Testovanie aplikácie prebieha na viacerých úrovniach, od jednotkových testov jednotlivých metód, cez testy užívateľského rozhrania až po rôzne testy bezpečnosti a výkonu. Každý z týchto testov má v procese vývoja a následného testovania aplikácie svoje miesto a opodstatnenie, každá z úrovní sa zameriava na činnosti z inej fázy tvorby softvéru [3]:

jednotkové testovanie individuálnych kúskov kódu (jednotlivé metódy triedy)

integračné testovanie množstva jednotiek dokopy

regresné testovanie už raz otestovaných jednotiek po každej zmene kódu

akceptačné testovanie aplikácie podľa očakávaní zákazníka

1.2 Biela a čierna skrinka

Medzi jeden z možných spôsobov klasifikácie prístupov testovania aplikácie patrí rozdelenie na testy čiernej a bielej skrinky. Vychádza z úrovneznalostí kódu aplikácie a jej vnútorného fungovania [2].

V prípade testovania softvéru ako bielej skrinky sa skúma fungovanie algoritmov, interných mechanizmov a využíva analýzu programového kódu. Tento pohľad umožňuje lepšie odhadnúť, ktoré z vetiev algoritmu budú vykonávané častejšie, než iné a tým lepšie prispôsobiť testovanie aplikácie. Testy založené na tomto princípe dôsledne testujú správanie sa aplikácie, avšak príliš nekorešpondujú s reálnymi spôsobmi využívania aplikácie [4]. Výhody testovania pomocou bielej skrinky sú: efektívne nájdenie chýb v zdrojovom

kóde, pomáha písať kvalitný (testovateľný) kód, vysoké pokrytie testov. Nevýhody testovania pomocou bielej skrinky sú: vyžaduje prístup k zdrojovému kódu, vyžaduje vysokú znalosť vnútorných procesov aplikácie.

V prípade testovania aplikácie ako čiernej skrinky dochádza ku skúmaniu bez akejkoľvek znalosti interných mechanizmov a fungovania testovanej aplikácie. Overuje sa iba výsledok, ako sa zadané vstupné hodnoty zmenili, či zodpovedajú očakávaným výstupom. Tento spôsob na jednej strane simuluje reálny spôsob použitia aplikácie, avšak na druhej strane generuje nadmerné množstvo testovacích prípadov určitých častí programu a zároveň nedostatočnému otestovaniu iných [2, 4]. Výhody testovania pomocou čiernej skrinky sú: efektivita pri velkých projektoch, nepotrebný prístup k zdrojovému kódu, separácia uživateľskej a vývojárskej perspektívy. Nevýhody testovania pomocou čiernej skrinky sú: neefiktivita z dôvodu nevedomosti vnútorných procesoch aplikácie, slabé pokrytie testov.

2 Selenium

Selenium je projekt, ktorý zastrešuje množstvo nástrojov a knižníc slúžiacich na automatizovanie ovládania internetových prehliadačov [5].
Selenium poskytuje:

- rozšírenia na emuláciu interakcie užívateľa s prehliadačom
- distribučný server slúžiaci na zväčšenie alokácie zdrojov (prehliadačov)
- infraštruktúru na implementáciu W3C WebDriver[1] špecifikácie

2.1 Selenium Remote Control

Známy ako Selenium 1, je staršia verzia Selenium API, ktorá už v súčasnosti nie je naďalej vyvíjaná a podpora zostala zachovaná už iba v rámci spätnej kompatibility a nedochádza k ďalšiemu vývoju. Selenium 1 dovoľuje písanie automatických testov vo forme skriptov, ktoré dokážu byť interpretované používanými prehliadačmi podporujúcimi JavaScript. Selenium 1 používa proxy server a injektuje JavaScript do prehliadača s cieľom kontrolovať ho [2, 6].

2.2 Selenium IDE

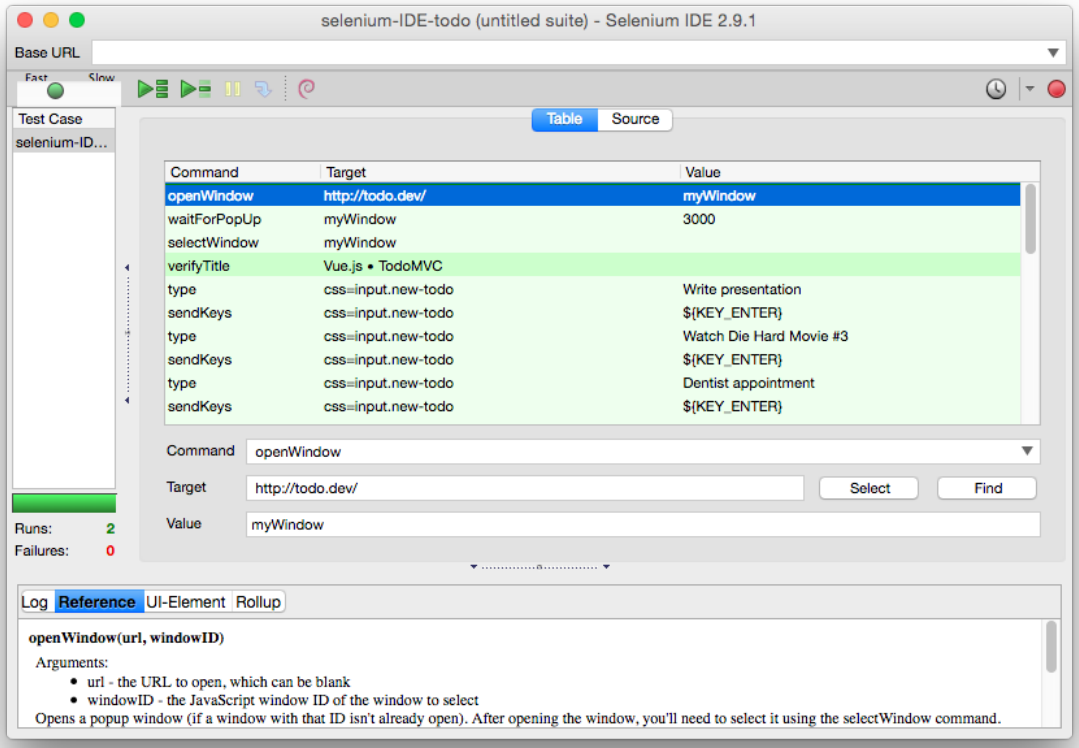
Selenium IDE (obr. 1) je prídavok (addon, plugin) do prehliadača Firefox a je určený na nahrávanie (a spätné spustenie) testovacích krokov pre prehliadač Firefox. Selenium IDE môže byť tiež použité na rýchle generovanie testovacieho kódu a následného konvertovania do programovacích jazykov ako Java, Python či Ruby [7].

2.3 Selenium WebDriver

Známy ako Selenium 2, je nástupcom Selenium Remote Control. Selenium 2 má odlišné API, tým pádom nie je spätne kompatibilné so Selenium RC. Od predchádzajúcej implementácie sa líši v spôsobe komunikácie s prehliadačmi a kladie dôraz na lepšiu podporu dynamických webových stránok, kde môže dochádzať ku zmenám elementov bez potreby opakovaného načítavania stránky (manipulácia s DOM[2] elementmi) [2, 6].

Selenium 2 podporuje automatizáciu väčšinu známych prehliadačov spolu s prehliadačmi pre mobilné zariadenia s operačným systémom Android a iOS [8]. V tabuľke č. 1 je zoznam prehliadačov a ich podporované verzie.

[1] `https://www.w3.org/TR/webdriver/`
[2] `http://stackoverflow.com/questions/1122437/what-is-dom-element`



Obrázok 1: Selenium IDE

WebDriver je API a protokol, ktorý definuje jazykovo neutrálne rozhranie na kontrolu správania sa prehliadačov. Každý prehliadač má špecifickú WebDriver implementáciu, ktorý sa nazýva ovládač (driver). Ovládač je komponenta zodpovedná za delegovanie príkazov smerom k prehliadaču, a teda riadi komunikáciu medzi prehliadačom a frameworkom Selenium. Ovládač dokáže ovládať prehliadač na nízkej úrovni, dôsledkom čoho je simulácia reálneho používateľa prehliadača.

Selenium 2 sa snaží využívať ovládače tretích strán s vysokou mierou. Pre prehliadače, pri ktorých nie je možné použiť natívny ovládač, ovládače poskytuje Selenium Organizáciou [6, 5].

2.4 Selenium Grid

Selenium Grid umožnuje paralelné spúštanie sady automatických testov proti rôznym prehliadačom na rôznych klientských staniciach zároveň. Umožňuje distribuované vykonávanie testov a vytvára pomerne ľahko spravovateľné prostredie pre automatické testy. Výhodou tohto riešenia je jednoduchá konfigurácia testov, kde na začiatku testu dôjde k

Tabuľka 1: Podporované prehliadače

prehliadač	**vývojár (maintainer)**	**podporované verzie**
Chromium	Chromium	všetky
Firefox	Selenium	4 a novšie
Internet Explorer	Selenium	6 a novšie
Opera	Opera Chromium / Presto	10.5 a novšie
Safari	Selenium	5.1 a novšie

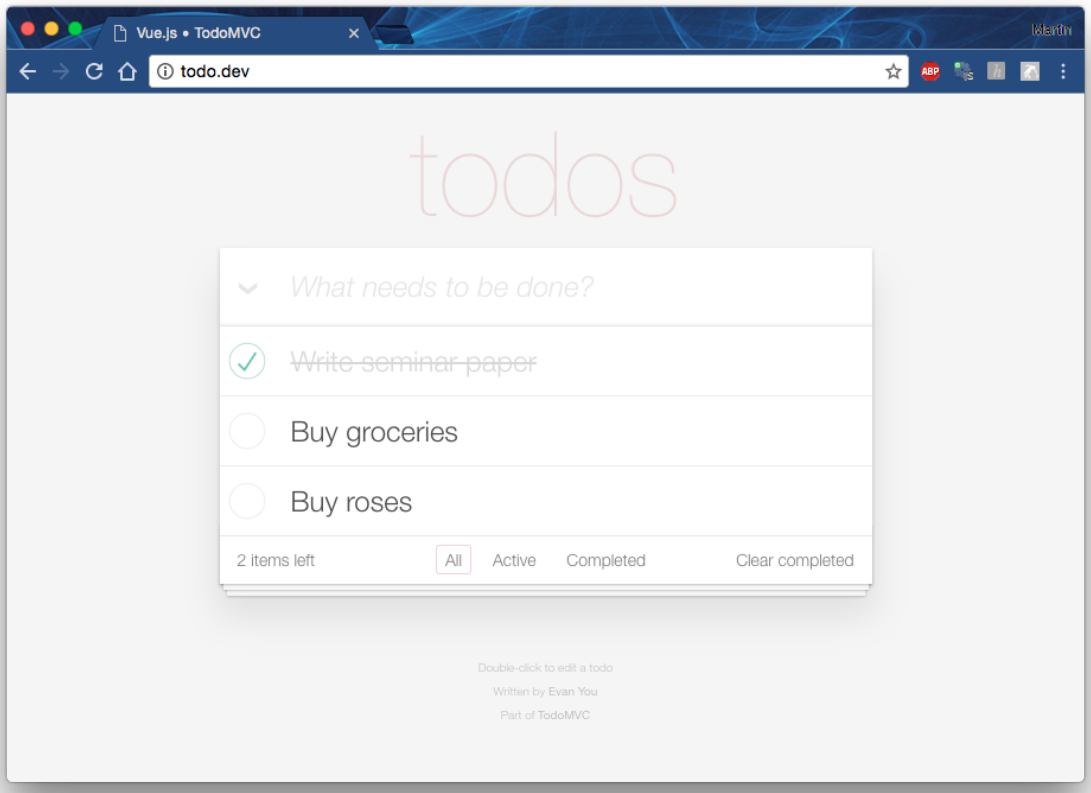
definovaniu požadovaných parametrov – prehliadač, jeho verzia, operačný systém a podobne. Centrálny uzol sa sám postará o priradenie konkrétneho testu príslušnej stanici podľa požadovaných kritérií. Týmto mechanizmom je umožnený paralelný beh mnohých testov ako aj vyrovnávanie záťaže medzi pripojenými uzlami [2, 7].

3 Demonštrácie práce so Selenium

3.1 Testovacia aplikácia

K samotnému demonštrovaniu nástrojov Selenium, je potrebná aplikácia, na ktorej môžeme vykonávať testy. Snažil som sa vybrať takú aplikáciu aby bola napísaná v JavaScripte a nepotrebovala server. Z dlhého zoznamu[3] už naprogramovaných Todo aplikácií som vybral implementáciu v JavaScriptovom frameworku Vue.js[4].

Todos aplikácia je jednoduchá implementácia Vue.js frameworku a dovoľuje uživateľovi vytvoriť, označiť či filtrovať zoznam úloh (CRUD a filtrovanie). Testovacia aplikácia je zobrazená na obrázku č. 2. Kompletný návod ako rozbehať todos aplikáciu je popísaný v prílohe B.2.



Obrázok 2: Todos Application

[3] https://github.com/tastejs/todomvc
[4] http://vuejs.org/

3.2 Selenium IDE

Testovanie uživateľského rozhrania pomocou Selenium IDE je v skutočnosti jednoduchý ale zdĺhavý proces. Najskôr je potrebné nainštalovať si doplnok Selenium IDE[5] pre internetový prehliadač Firefox. Po úspešnom nainštalovaní doplnku treba doplnok zapnúť (alebo nechať zobraziť), a to tak, že v hornej lište prehliadača Firefox prejdeme na zložku Tools a následne vyberieme z menu Selenium IDE, výsledkom čoho je zobrazenie ako na obrázku č. 1.

Proces testovania spočíva v postupnom preklikávaní sa todos aplikáciou, pričom je zapnuté nahrávanie v prostredí Selenium IDE. Výsledkom nahrávania akcií je HTML súbor (tabuľka), ktorý obsahuje jednotlivé akcie ako napríklad: `openWindow`, `verifyTitle`, `click`, `type`, `sendKeys` a iné[6], spolu s parametrami, ktoré Selenium IDE zachytilo pri vyklikávaní (alebo písaní textu, pohybu kurzoru myši a iné).

Selenium IDE vyhodnocuje úspešnosť daného testu (sekvenciu akcií) spôsobom, či sa vôbec daný test dá spustiť to znamená, že ak má Selenium IDE v teste akciu "klikni na tlačítko s id=add", no reálne také tlačítko nie je nájditeľné v DOM–e, celý test je označený ako "neprejdený".

V tabuľke č. 2 je (skrátený) zoznam akcií, ktoré rozhodujú o úspešnosti testu. Samotný test uložený v súbore `application/selenium-ide-TODO`[7].

Tabuľka 2: Zoznam akcií v teste pri použití Selenium IDE

akcia	**parameter**	**parameter–2**
otvor okno	http://todo.dev	–
verifikuj nadpis	Vue.js TodoMVC	–
napíš do	input.new-todo	Write presentation
zmačkni	ENTER	–
napíš do	input.new-todo	Watch Die Hard Movie #3
zmačkni	ENTER	–
klikni na	link=Active	–
klikni na	link=Completed	–
klikni na	css=button.clear-completed	–
zavri okno	–	–

[5] `https://addons.mozilla.org/sk/firefox/addon/selenium-ide/`

[6] `http://seleniummaster.com/sitecontent/index.php/introduction-to-selenium-automation/selenium-ide/114-selenium-ide-complete-list-of-commands`

[7] `https://github.com/Kyslik/asos-selenium/blob/master/application/selenium-IDE-todo`

3.3 Selenium 2

Pre demonštráciu Selenium 2 (WebDriver) som sa rozhodol použiť jazyk PHP, ktorý nepotrebuje kompiláciu a má jednoduchú syntax.

Knižnica php-webdriver je obal (wrapper) pre Selenium WebDriver v jazyku PHP z dielne Facebook-u, ktorá umožnuje kontrolu respektíve ovládanie internetových prehliadačov priamo z PHP [9]. V dokumentácií[8] php-webdriver-u je dôsledne popísaný postup ako s ním pracovať.

PHPUnit je najpopulárnejší[9] framework na vykonávanie automatizovaných testov v jazyku PHP. Pre všetky testy vo frameworku PHPUnit platí, že pred každým testom sa spustí metóda `setUp()` a po ukončení testu metóda `tearDown()`, to znamená, že každý test je "spustený vo vákuu" (jednotkový test). Konkrétnejšie v demonštračnom zdrojovom kóde v metóde `setUp()` sa nastaví: spojenie (so selenium-webdriver-serverom), prehliadač (typ) a jeho vlastnosti (napr. velkosť okna) a v metóde `tearDown()` je kód na zatvorenie okna internetového prehliadača.

Kompletný návod ako nainštalovať a rozbehať testovaciu aplikáciu spolu s PHPUnit a Selenium 2 je popísaný v prílohe B.1.

V tabuľke č. 3 je (úplný) zoznam testov, ktoré testujú todos aplikáciu.

Tabuľka 3: Zoznam testov pri použití Selenium 2

názov testu	**vysvetlenie**
TodoAppHome	Testuj dostupnost aplikácie (http://todos.dev)
AddingTodos	Testuj pridávanie úloh
MarkingAsComplete	Testuj označenie ako úlohu ako "splnenú"
DestroyingTodo	Testuj vymazanie úlohy
EditingTodo	Testuj editovanie úlohy
MakeAnEditToTodo	Testuj vykonanie zmeny úlohy (po editácii)
SeeAllTodos	Testuj filtrovanie úloh (filter "všetky")
SeeActiveTodos	Testuj filtrovanie úloh (filter "aktívne")
SeeCompletedTodos	Testuj filtrovanie úloh (filter "splnené")
ClearCompletedTodos	Testuj vymazanie "splnených" úloh
CheckAllTodosAsCompleted	Testuj označenie všetkých úloh ako "splnené"
CheckAllTodosAsActive	Testuj označenie všetkých úloh ako "aktívne"

[8] `https://github.com/facebook/php-webdriver/wiki`

[9] `http://www.hongkiat.com/blog/automated-php-test/`

3.4 Porovnanie Selenium IDE a Selenium 2

Prácu so Selenium IDE by som zhrnul do nasledovných bodov:

plus jednoduché na používanie

plus nepotrebuje žiaden server (selenium-webdriver-server)

mínus funguje iba na internetovom prehliadači Firefox

mínus veľmi pomalé testovanie (vykonanie testu trvá dlho)

Prácu so Selenium 2 by som zhrnul do nasledovných bodov:

plus vhodné pre programátorov

plus podporuje veľké množstvo internetových prehliadačov

plus podporuje veľké množstvo programovacích jazykov (Java, Python, Ruby)[10]

plus oveľa rýchlejšie vykonenie testov (v porovnaní so Selenium IDE)

mínus potreba študovať dokumentáciu

[10] `http://docs.seleniumhq.org/docs/03_webdriver.jsp`

Záver

V seminárnej práci som zjednodušene popísal, na aké účely slúži automatizované testovanie, aké typy automatizovaného testovania existujú, aký je rozdiel medzi testovaním "bielej" a "čiernej" skrinky. Podrobne som predstavil nástroj Selenium, ktorý slúži na automatizované testovanie uživateľského rozhrania. Demonštroval som prácu so Selenium IDE, ako aj prácu so Selenium 2. Cieľ seminárnej práce sa mi podaril splniť.

Zoznam použitej literatúry

1. ZALLAR, Kerry. *Practical Experience in Automated Testing* [online] [cit. 2016-10-24]. Dostupné z: http://www.methodsandtools.com/archive/archive.php?id=33.
2. SOKOL, Martin. *Automatické testování webových aplikací*. 2013. Diplomová práca. MASARYKOVA UNIVERZITA FAKULTA INFORMATIKY.
3. GETLAURA. *TESTING: UNIT VS INTEGRATION VS REGRESSION VS ACCEPTANCE* [online]. 2014 [cit. 2016-10-24]. Dostupné z: http://www.getlaura.com/testing-unit-vs-integration-vs-regression-vs-acceptance/.
4. FARCIC, Viktor. *Black-box vs White-box Testing* [online]. 2013 [cit. 2016-10-25]. Dostupné z: https://technologyconversations.com/2013/12/11/black-box-vs-white-box-testing/.
5. *Selenium Documentation* [online]. 2016 [cit. 2016-10-24]. Dostupné z: https://seleniumhq.github.io/docs/start.html.
6. *Selenium Documentation* [online]. 2016 [cit. 2016-10-22]. Dostupné z: http://www.seleniumhq.org/docs/.
7. *Selenium (software)* [online]. 2016 [cit. 2016-10-24]. Dostupné z: https://en.wikipedia.org/wiki/Selenium_(software).
8. KRILL, Paul. *Open source Selenium web app test suite to support iPhone and Android* [online] [cit. 2016-10-20]. Dostupné z: http://news.techworld.com/applications/3272444/open-source-selenium-web-app-test-suite-to-support-iphone-and-android/.
9. *PHP Webdriver*. 2013. Dostupné tiež z: https://github.com/facebook/php-webdriver.

Prílohy

A Štruktúra elektronického nosiča

/
/LICENSE
/README.md
/application
/application/composer.json
/application/composer.lock
/application/index.html
/application/LICENSE
/application/package.json
/application/phpunit.xml
/application/README.md
/application/selenium-IDE-todo
/application/vue.js-todo.png
/application/js
/application/js/app.js
/application/js/routes.js
/application/js/store.js
/application/tests
/application/tests/AbstractTestCase.php
/application/tests/TodoApp
/application/tests/TodoApp/TodoAppTest.php
/documentation
/documentation/FEIstyle.cls
/documentation/LICENSE
/documentation/Makefile
/documentation/selenium.sublime-project
/documentation/selenium_kiesel.pdf
/documentation/selenium_kiesel.tex
/documentation/img
/documentation/img/blok.png
/documentation/img/vzhlad.png
/documentation/includes

/documentation/includes/abbreviations.tex
/documentation/includes/attachmentA.tex
/documentation/includes/attachmentB.tex
/documentation/includes/attachmentC.tex
/documentation/includes/bibliography.bib
/documentation/includes/conclusion.tex
/documentation/includes/core.tex
/documentation/includes/declaration.tex
/documentation/includes/introduction.tex
/documentation/includes/resume.tex
/presentation
/presentation/selenium-martin-kiesel.pdf

B Testing Todo application

Demonstration of Selenium using php-webdriver by Facebook on top of PHPUnit.

B.1 Prerequisites

Make sure your server (computer) meets the following requirements:

- PHP >= 5.6.4 (server (`apache2` or `nginx`) is **not required**)
- composer (how to install guide[11])
- npm (how to install guide[12])
- selenium-standalone-server 3.0.1 [∼20MB][13]
- browser specific drivers (whole list can be found here[14] in section *Third Party Browser Drivers*)
  - chromedriver[15] - Google Chrome
  - geckodriver[16] - Mozilla Firefox
  - safaridriver[17] - Safari
  - edgedriver[18] - Microsoft Edge
  - operadriver[19] - Opera

Note: I tested this working example only with **chromedriver**
Note: on Mac OS you can install **selenium-standalone-server** (and some drivers) using homebrew[20] `brew install selenium-server-standalone brew install chromedriver`

[11] https://getcomposer.org/doc/00-intro.md#installation-linux-unix-osx
[12] https://docs.npmjs.com/getting-started/installing-node
[13] http://selenium-release.storage.googleapis.com/index.html?path=3.0/
[14] http://www.seleniumhq.org/download/
[15] https://sites.google.com/a/chromium.org/chromedriver/
[16] https://github.com/mozilla/geckodriver/releases
[17] https://github.com/SeleniumHQ/selenium/wiki/SafariDriver
[18] https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/
[19] https://github.com/operasoftware/operachromiumdriver
[20] http://brew.sh/index.html

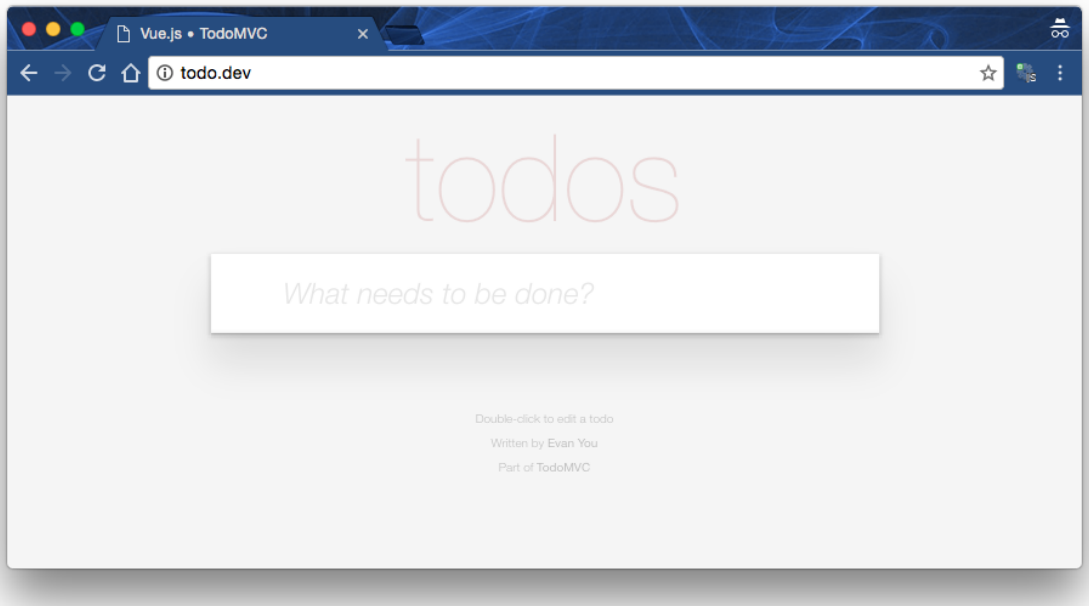
B.2 Installation && setup

- install php dependencies using composer
- install Todo application using npm

  In terminal change working directory to `./application` and run following:

  ```
  composer install
  composer dump-autoload -o
  npm install
  ```

- serve `./application/index.html`; two options are available (remember which one you used)
  - set up server and serve using virtual-host
  - use `file:///<path to application folder>/index.html`
- manually test working Todo application you should see following:



Note: I am using apache2 and vhost to serve static page on local domain *todo.dev*

After package managers install all required software you are ready to use example code.

B.3 Usage /running tests

Change `$url` variable on TodoAppTest.php on line 11[21] to reflect path to Todo application.

Open up two terminal windows (or tabs), one is going to be used for **selenium-standalone-server** (tab A) and second one is going to be used for **PHPUnit** (tab B).

- in tab A start **selenium-standalone-server** on port 4444 (default), to do so run following: `java -jar <path_to>/selenium-server-standalone-<version>.jar` >**Note**: on Mac OS if you used homebrew to install selenium-standalone-server you can run following: `selenium-server -port 4444`
- in tab B change working directory to `./application` and simply run `vendor/bin/phpunit`

B.4 Brief explanation

In tab A you can monitor what exactly is PHPUnit invoking while test is beeing run (kind of selenium log). In tab B you may observe execution of PHPUnit test suite located in `./application/tests/TodoApp`.

On the background PHPUnit invokes selenium-standalone-server with some rules/ patterns and selenium-standalone-server translates it to (in this case) chromedriver API.

[21] https://github.com/Kyslik/asos-selenium/blob/master/application/tests/TodoApp/TodoAppTest.php#L11

C Kontrolné otázky a odpoveďe

1. Čo to je Selenium? Na aké činnosti sa v praxi používa?
2. Napíšte plusy a mínusy pre: automatizované a manuálne testovanie.

C.1 Odpoveď na otázku č. 1

Selenium projekt vyvýja nástroje a knižnice slúžiace na automatizovanie ovládania internetových prehliadačov. V praxi sa využíva na testovanie webových aplikácii. Počas testovaniu Selenium simuluje uživateľské vstupy. Selenium podporuje väčšinu populárnych prehliadačov (Chrome, Firefox, Opera, Safari, Edge), ale aj prehliadače určené pre operačný systém Android či iOS.

C.2 Odpoveď na otázku č. 2

Automatizované testovanie plusy: rýchly a efektývny proces testovania, šetria peniaze pri dlhodobom vývoji softvéru, samo-dokumentácia kódu
Automatizované testovanie mínusy: pomalý vývoj, vyššie náklady pri menších projektoch
Manuálne testovanie plusy: menšie náklady pri menších projektoch, väčšia šanca nájdenia problému v grafickom rozhraní, flexibilita – rýchla adaptácia zmene kódu
Manuálne testovanie mínusy: repetitívne a nudné, niektoré testy sú nerealizovateľné – ťažko vykonateľne (testovanie nízko úrovňového rozhrania, alebo API)