# Vysoká škola ekonomická v Praze

## Fakulta informatiky a statistiky

## Katedra informačních technologií


Studijní program: Aplikovaná informatika

Obor: Informatika

# Testování webových aplikací pomocí nástroje Robot Framework

# BAKALÁŘSKÁ PRÁCE


Student: Renat Kulalov

Vedoucí: doc. Ing. Alena Buchalcevová, Ph.D.

Oponent: Ing. Marcel Veselka


**2016**

**Prohlášení:**

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze které jsem čerpal.

V Praze dne 5. 05. 2016 . . . . . .

Renat Kulalov

## Poděkování

Rad bych podekoval paní doc. Ing. Aleně Buchalcevové, Ph.D. za metodické vedení mé práce, připomínky, ochotu a vstřícnost. Dále bych rad podekoval panu Ing. Marcelu Veselkovi za přínosné informace a rady které mi pomohly ke zpracování tohoto tématu.

## Abstrakt

Tato bakalářská práce se zabývá automatizovaným testováním webových aplikaci. Cílem této práce je představeni testovacího nástroje Robot Framework, který umožňuje automatizaci testovacích případů. Teoretická část obsahuje úvodní informaci o tom co je automatizované testování softwaru a jaké má automatizace testovacích případů přínosy. Dále teoretická část obsahuje popis několika vybraných nástrojů pro automatizované testování. V praktické částí nejdříve představen nastroj Robot Framework, návod na instalaci a konfiguraci, psáni automatizovaných testovacích případů a propojeni nástroje Robot Framework s nástrojem BrowserStack. Po seznámení s práci je potenciální uživatel schopen automatizovat testovací případy v nastrojí Robot Framework a spouštět je na cloudovem nástroje BrowserStack. Vlastním příspěvkem k tématu je vytvoření příručky v českém jazyce, která je využitelná při výuce testování softwaru na Vysoké škole ekonomické v Praze.

## Klíčová slova

Robot Framework, BrowserStack, automatizované testy, webové aplikace

## Abstract

This bachelor's thesis deals with automated testing of web applications. The goal of this thesis is to introduce Robot Framework, which is a tool for test case automation. Theory section introduces and describes test automation in general and what benefits it brings to software testing. It also introduces a few chosen tools for automated software testing. Practical part deals with installation and configuration of Robot Framework, writing automated test cases and integration of Robot Framework with BrowserStack. After reading the thesis a potential user will be able to write automated test cases in Robot Framework and run them on BrowserStack. The contribution of this thesis is the emergence of guide in Czech language, which is useful teaching software testing at the University of Economics in Prague.

## Keywords

Robot Framework, BrowserStack, automated tests, web applications

# Obsah

# 1. Úvod

V dnešním světě webové aplikace jsou nedílnou části každodenního života, protože ovlivňují a zasahuji důležité oblasti naších životů. Proto je velmi důležité zajistit kvalitu webových aplikací. V současnosti je obecný trend testové případy automatizovat. Automatizace nese s sebou řadu velkých výhod, jako snížení nákladů a času potřebného na testování softwaru. [4] Existuje vícero nástrojů pro automatizací testovacích případů, v této práci se popisuje práce s jedním z takových nástrojů, který se jmenuje Robot Framework. Vlastní text práce je rozdělen do teoretické a praktické části. Teoretická část obsahuje úvodní informaci o tom, co je automatizované testování softwaru a jaké má automatizace testovacích případů přínosy. Dále teoretická část obsahuje popis nástrojů Robot Framework, BrowserStack a Testlink. V praktické části je představen návod na instalaci a konfiguraci nástroje Robot Framework, psaní automatizovaných testovacích případů a propojení nástroje Robot Framework s nástrojem BrowserStack. Webová aplikace, která je v této bakalářské práci testovaná, je Testlink.

## 1.1. Vymezení tématu práce a důvod výběru tématu

Tématem práce je testování webových aplikaci pomoci nástroje Robot Framework. Práce bude osahovat teoretickou část, která obsahuje základní pojmy a nutnou teorii k pochopení praktické části. Praktická část obsahuje v počátku instalace Robot Frameworku a jeho knihoven, propojení s testovacím nástrojem BrowserStack. Dále práce obsahuje návod na psaní automatizovaných testů, jak spouštět tyto testy pomocí nástroje BrowserStack a tím zajistit pokrytí velkého počtů konfigurací.

S nástrojem Robot Framework přicházím do styku v práci a to je hlavním důvodem, proč jsem toto téma vybral. Dalším důvodem je absence příručky k nástroji Robot Framework v českém jazyce.

## 1.2. Cíl práce

Hlavním cílem bakalářské práce je představení nástroje Robot Framework a vytvoření příručky pro jeho použití, která umožní začít psát automatizované testy a spouštět je na cloudovém testovacím nástroji BrowserStack.

## 1.3. Způsob dosažení cíle

Pro úspěšné dosazení cíle jsem nainstaloval nástroj Robot Framework a vytvořil účty ve webových aplikacích Browserstack a Testlink. Tyto nástroje jsem použil při tvorbě této bakalářské práce. Dále jsem použil materiály z odborné literatury, které se věnují automatizovanému testování softwaru.

## 1.4. Předpoklady a omezení práce

Tato práce nepředpokládá žádné předchozí zkušeností z automatizovaným testováním. Práce je vhodná jak pro testery, kteří jsou zkušení v oblasti automatizace testu, tak i pro úplné nováčky kteří chtějí začít automatizovat testovací případy. Předpokládá se ale zkušenost s manuálním testováním.

## 1.5. Struktura práce

Práce je sestavena z několika kapitol. V práci jsou vymezeny dvě hlavni části, teoretická a praktická. Teoretická část je zahrnuta v kapitolách 2, 3,4, 5, které obsahuji základní pojmy a nutnou teorii k pochopení praktické části. Praktická část je reprezentovaná kapitolou 6 a obsahuje na začátku instalaci Robot Frameworku a jeho knihoven, propojení s testovacím nástrojem BrowserStack. Dále praktická část obsahuje návod na psaní automatizovaných testu, jak spouštět

tyto testy pomocí nástroje BrowserStack a tím zajistit pokrytí velkého počtů konfigurací. Závěr je sedmou a poslední kapitolou této práce, obsahuje vyhodnocení cílů této bakalářské práce.

## 1.6. Výstup a očekávaný přínos

Výstupem bakalářské práce je vznik příručky k nástroji Robot Framework, podle které čtenář bude schopen zprovoznit a používat Robot Framework pro psaní automatizovaných testů. Očekávaným přínosem je vytvoření příručky, která bude použitelná při výuce automatizovaného testování softwaru na Vysoké Škole Ekonomické. Po přečtení práce uživatel bude schopen používat Robot Framework pro psaní automatizovaných testů.

## 1.7. Rešerše zdrojů

V této podkapitole jsem uvedl akademické práce a literaturu, která se zabývá automatizovaným testováním, nástrojem Robot Framework, a jsou relevantními zdroji pro tuto bakalářskou prací.

Hlavním knižním zdrojem je kniha Elfriede Dustina, Thoma Garreta a Bernie Gaufa *Implementing automated software testing: how to save time and lower costs while raising quality.* [2]. Kniha popisuje metodiky vyuzivane při automatizovanem testovani, vyhody a prinosy automatizace a nejlepší praxe při automatizaci testovacich pripadu.

Dalšími zdroji jsou Diplomová práce Pekki Klärcka *ata-Driven and Keyword-Driven Test Automation Frameworks.* Tato diplomová práce stala základem pro Robot Framework. Dále Jani Luostarinen se ve své bakalářské práci *Web Application Test Automation with Robot Framework* se zabývá testováním webových aplikací pomocí nástroje Robot Framework a Klára Smatanová ve své diplomové práci práci *Automatizované testování webových aplikací* popisuje automatizované testování celkově a představuje testovací nástroje.

Hlavnim weboym zdrojem je stranka *robotframewrk.org* , ve které lze najit veskere navody na instalaci a pouzivani nastroje Robot Framework a informace ke knihovnam

# 2. Automatizované testování softwaru

V této kapitole je popsáno co je Automatizované testovaní, jaké má automatizace cíle a jaké jsou hlavní přínosy automatizace testovacích případů.  Dále v této kapitole je popsáno testování řízené klíčovými slovy.

## 2.1. Automatizované testování softwaru

Elfriede Dustin, Thom Garret a Bernie Gauf [2]ve své knize definuje automatizované testování softwaru jako aplikaci a implementaci softwarové technologie přes celý životní cyklus testování softwaru s cílem vylepšení účinnosti a efektivity životního cyklu testování softwaru.

Hlavními rozdíly mezi manuálním a automatizovaným testováním podle Dustina, Garreta a Gaufa [2]je:

- Rozšiřuje rozsah zaměření testování, který by byl nedosažitelný manuálním testováním
- Vývoj softwaru
- Nenahrazuje analytické dovedností manuálního testera, testovací strategii, know how a porozumění testovacím technikám. Tyto odborné znalosti testera slouží jako blueprint pro automatizaci.
- Nemůže být jednoznačně odděleno od manuálního testování. Manuální a automatizované testování je propojeno a vzájemně se doplňuje.

Pokud pro daný software vytvořeny stabilní manuální testovací případy, které se opakovaně vykonávají, je nasnadě otázka, zda by nebylo možné tyto testovací případy vykonávat automaticky.[3] Automatizace testu při správném vytvoření a správné implementaci může podstatně snížit náklady, čas a prostředky potřebované pro testování softwaru.[2]

## 2.2. Přínosy automatizovaného testování

Automatizované testování softwaru má několik výhod oproti manuálnímu testování. Provádění opakovaných testů může byt nudným pro manuálního testera, což může přivést k menší efektivitě testování. Při automatizování opakovaných testů taková možnost odpadá, každé provádění opakovaného testu má stejně vysokou efektivitu. [1]

Automatizované testování je mnohem rychlejší než manuální a dalším očividným přínosem je snížení potřebné manuální práce v testovací fázi. Při porovnání s manuálním testováním, počet potřebovaných pracovních hodin může byt extrémně malý. [4]

Ve své knize Dustin, Garret a Gauf je položena základní otázka „Proč automatizovat?“

Odpovědi je:

- Redukce času a nákladů na testování softwaru
- Zlepšení kvality softwaru
- Vylepšení manuálního testování přes rozšíření testovacího pokrytí a nahrazení pracovně náročných úkolů
- Automatizace dokáže dosáhnout těžko dosažitelné manuální testovací cíle, jako je například testování výkonu

## 2.3. Testování řízené klíčovými slovy

Testování řízené klíčovými slovy je metoda testování softwaru, která odděluje většinu programovací části automatizace testovacích případů od samotného designu. To povoluje dřívější vyvíjení testovacích případů a jejich lehčí údržbu. Hlavními koncepty v testování řízeném klíčovými slovy jsou: [14]

- Klíčová slova, které jsou většinou základní úrovní a popisují operace s uživatelským rozhraním jako „click“, „select“, „enter“

- Byznys šablony, které jsou většinou pokročilou úrovní, jako „login“, „enter tansaction“
- Akční slova nebo akce, které mohou být základní úrovní nebo pokročilou úrovní a ve svém obecném tvaru povolují předem definovaným klíčovým slovům být používanými pro slova na pokročilé úrovni.

# 3. Robot Framework

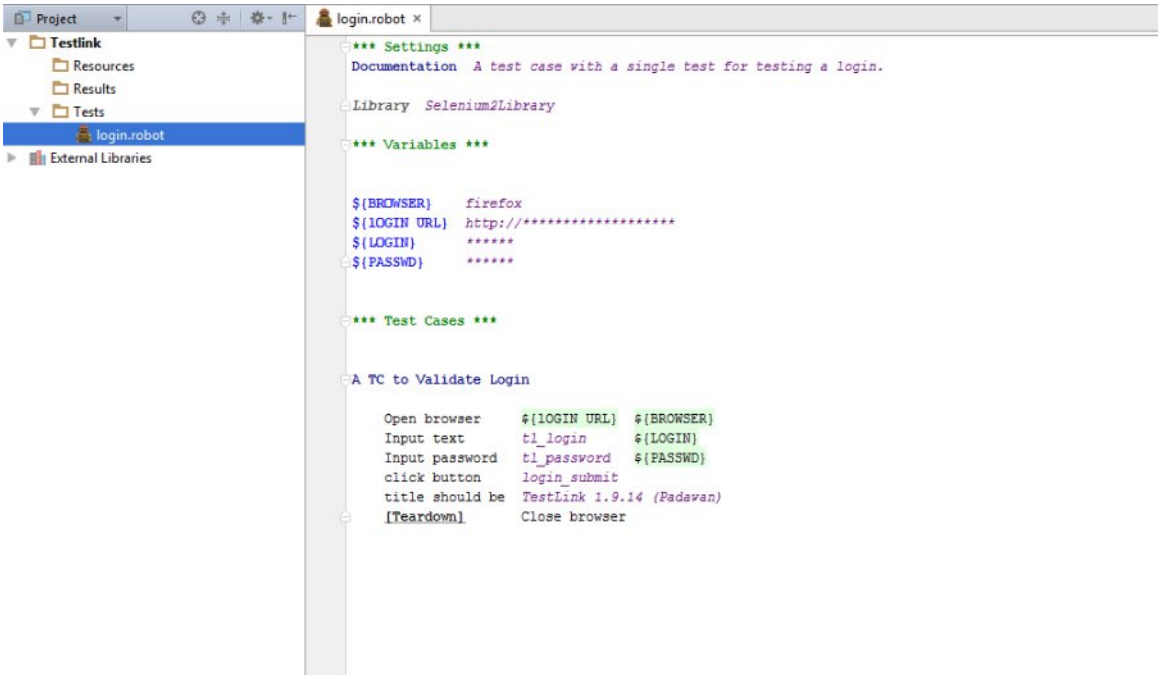
Robot Framework je open source framework pro automatizované testování používaný při acceptance testingu a acceptance test-driven developmentu (ATDD)[24]. Robot Framework má tabulkovou syntaxi testovacích dat a používá testování řízené klíčovými slovy. Testovací možnosti Robot Frameworku mohou byt rozšířeny prostřednictvím knihoven napsaných v jazycích Python nebo Java. [5]

## 3.1. Historie

Základní myšlenky pro Robot Framework byla navrhnuty v diplomové prací Pekki Klacka [1] v roce 2005. Poprvé byl Robot Framework stahován 24. června roku 2008.[5] Robot Framework je napsán v jazyce Python a spuštěn pod licencí Apache 2.0 [6]

## 3.2. Klíčová slova a Syntaxe

Framework začíná běh pasováním testovacích data souborů. Následně pro interakci s testovaným systémem framework používá klíčové slova, které jsou definována v testovacích data souborech. Knihovny komunikují přímo s testovaným systémem.[5]




Obrázek 1. Příklad jednoduchého test data souboru pro validaci správného přihlášeni. Vlastní zpracování.

Test data soubor ma 4 sekce: ***Settings***, ***Variables***, ***Test Cases*** a ***Keywords***. Dokumentace, deklarace knihoven nebo cesta k resource souboru jsou definované v sekci ***Settings***. Globální proměnné jsou psané v složených závorkách ({}) a před závorkami je psán znak dolaru ($) a jsou uvedeny v sekci ***Variables***. Testovací případy jsou psané v sekci ***Test Cases*** a příkazy (keywords) jsou psané do sekce ***Keywords***. [Obrázek 1]

Každé slovo se odděluje jednou mezerou. Dvě nebo vice mezer musí byt mezi klíčovým slovem a definici či proměnnou pro to, aby Robot Framework mohl odlišit proměnné nebo definice od klíčových slov.[7]

### 3.3. Selenium2Library

Selenium2Library je webová testovací knihovna pro Robot Framework. Knihovna používá Selenium 2 (WebDriver) knihovny pro ovládání webových prohlížečů. Knihovna Selenium2Library spouští testové případy v reálné instanci prohlížeče. Selenium2Library funguje

z většinou moderních prohlížečů, přičemž prohlížeč Mozilla Firefox je podporován bez potřeby instalaci webdriverů. Pro použití knihovny v testovacím případě je potřeba knihovnu importovat do sekce Settings.[8] Některé klíčová slova v knihovně Selenium2Library potřebují lokátor pro nalezení elementů na stránce. Lokátorem může být id elementu, xpath, css, jméno atd.

[Obrázek 2]

| Strategy | Example | Description |
|---|---|---|
| identifier | Click Element \| identifier=my_element | Matches by @id or @name attribute |
| id | Click Element \| id=my_element | Matches by @id attribute |
| name | Click Element \| name=my_element | Matches by @name attribute |
| xpath | Click Element \| xpath=//div[@id='my_element'] | Matches with arbitrary XPath expression |
| dom | Click Element \| dom=document.images[56] | Matches with arbitrary DOM express |
| link | Click Element \| link=My Link | Matches anchor elements by their link text |
| partial link | Click Element \| partial link=y Lin | Matches anchor elements by their partial link text |
| css | Click Element \| css=div.my_class | Matches by CSS selector |
| jquery | Click Element \| jquery=div.my_class | Matches by jQuery/sizzle selector |
| sizzle | Click Element \| sizzle=div.my_class | Matches by jQuery/sizzle selector |
| tag | Click Element \| tag=div | Matches by HTML tag name |
| default* | Click Link \| default=page?a=b | Matches key attributes with value after first '=' |

Obrázek 2. Moznosti identifikování prvku v knihovně Selenium2Library. [8]

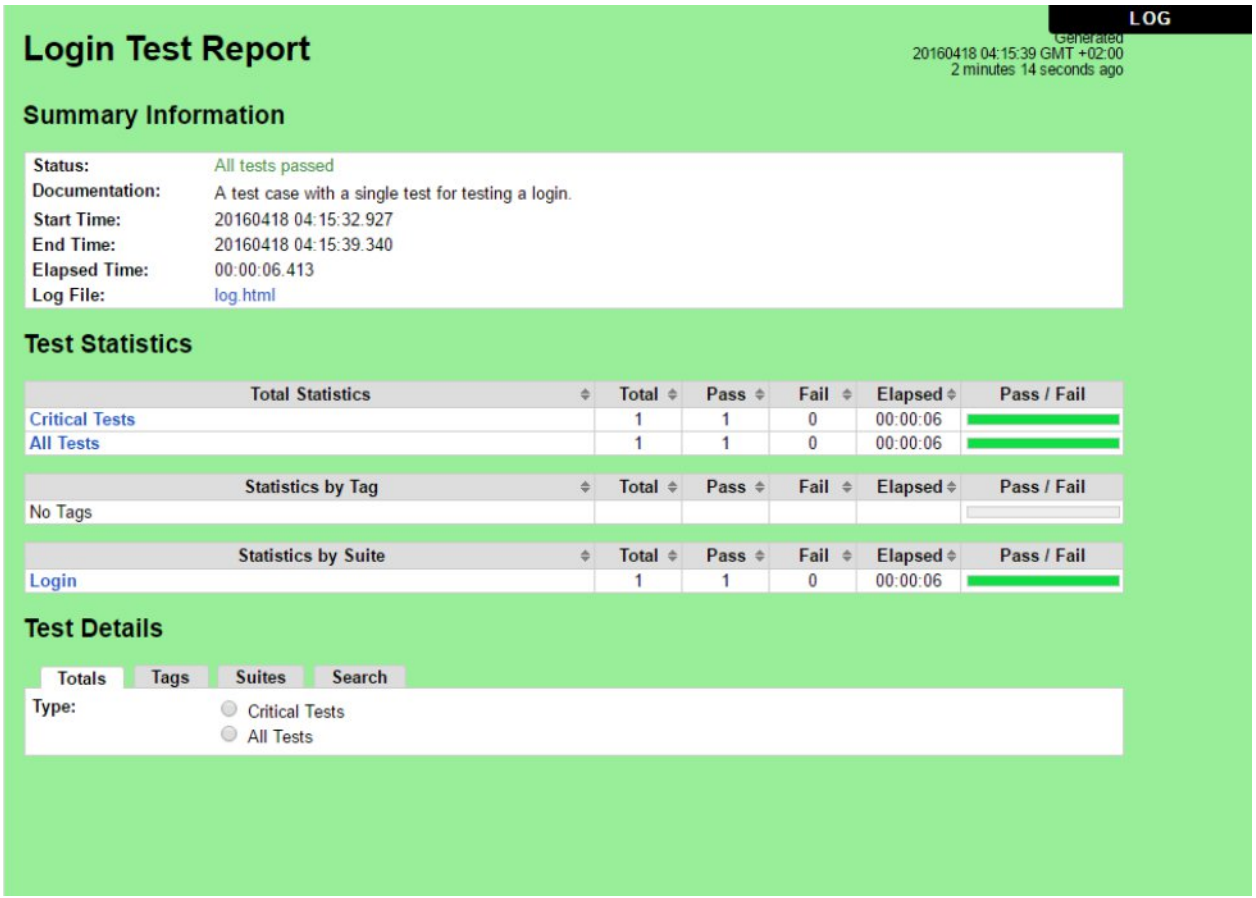
### 3.4 Selenium 2 (WeDdriver)

Selenium 2 (WeDdriver) je nástroj pro automatizaci testovacích případů, který obsahuje WebDriver API. Selenium 2 byl vyvinut pro lepší podporu dynamických webových stránek, na kterých elementy se mohou změnit bez obnovení webové stránky.[9]

### 3.5 Reporty a Logy

Po ukončení běhu testovacího případu Robot Framework ukládá výsledky do reportu a logu. Reporty a logy jsou uloženy ve formátu HTML. Report obsahuje lehce přehledné statistiky, včetně podílu PASS/FAIL a časů trvání testovacího případu. Log umožňuje přehled výsledků

běhu všech klíčových slov. [Obrázek 3] Pomocí logu lze přesně dozvědět na kterém kroku se objevila chyba.[10] [Obrázek 4]



LOG

# Login Test Report

Generated
20160418 04:15:39 GMT +02:00
2 minutes 14 seconds ago

## Summary Information

| | |
|---|---|
| Status: | All tests passed |
| Documentation: | A test case with a single test for testing a login. |
| Start Time: | 20160418 04:15:32.927 |
| End Time: | 20160418 04:15:39.340 |
| Elapsed Time: | 00:00:06.413 |
| Log File: | log.html |

## Test Statistics

| Total Statistics | Total | Pass | Fail | Elapsed | Pass / Fail |
|---|---|---|---|---|---|
| Critical Tests | 1 | 1 | 0 | 00:00:06 | |
| All Tests | 1 | 1 | 0 | 00:00:06 | |

| Statistics by Tag | Total | Pass | Fail | Elapsed | Pass / Fail |
|---|---|---|---|---|---|
| No Tags | | | | | |

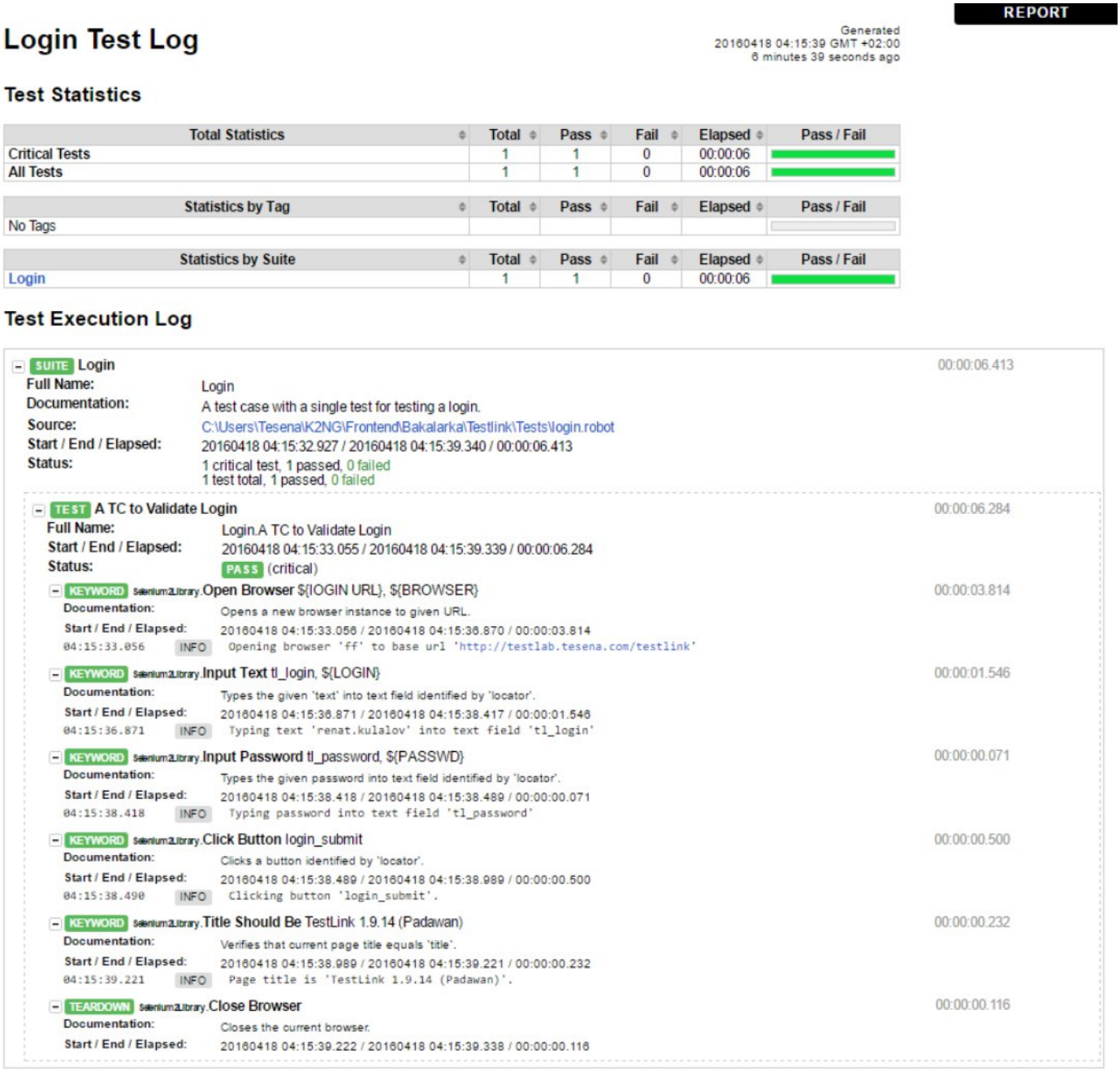
| Statistics by Suite | Total | Pass | Fail | Elapsed | Pass / Fail |
|---|---|---|---|---|---|
| Login | 1 | 1 | 0 | 00:00:06 | |

## Test Details

Totals | Tags | Suites | Search

Type: Critical Tests / All Tests

Obrázek 3. Report testovacího případu z obrázku 1. Vlastní zpracování.



REPORT

# Login Test Log

Generated
20160418 04:15:39 GMT +02:00
6 minutes 39 seconds ago

## Test Statistics

| Total Statistics | Total | Pass | Fail | Elapsed | Pass / Fail |
|---|---|---|---|---|---|
| Critical Tests | 1 | 1 | 0 | 00:00:06 | |
| All Tests | 1 | 1 | 0 | 00:00:06 | |

| Statistics by Tag | Total | Pass | Fail | Elapsed | Pass / Fail |
|---|---|---|---|---|---|
| No Tags | | | | | |

| Statistics by Suite | Total | Pass | Fail | Elapsed | Pass / Fail |
|---|---|---|---|---|---|
| Login | 1 | 1 | 0 | 00:00:06 | |

## Test Execution Log

SUITE Login — 00:00:06.413
Full Name: Login
Documentation: A test case with a single test for testing a login.
Source: C:\Users\Tesena\K2NG\Frontend\Bakalarka\Testlink\Tests\login.robot
Start / End / Elapsed: 20160418 04:15:32.927 / 20160418 04:15:39.340 / 00:00:06.413
Status: 1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed

TEST A TC to Validate Login — 00:00:06.284
Full Name: Login.A TC to Validate Login
Start / End / Elapsed: 20160418 04:15:33.055 / 20160418 04:15:39.339 / 00:00:06.284
Status: PASS (critical)

KEYWORD Selenium2Library.Open Browser ${IOGIN URL}, ${BROWSER} — 00:00:03.814
Documentation: Opens a new browser instance to given URL.
Start / End / Elapsed: 20160418 04:15:33.056 / 20160418 04:15:36.870 / 00:00:03.814
04:15:33.056 INFO Opening browser 'ff' to base url 'http://testlab.tesena.com/testlink'

KEYWORD Selenium2Library.Input Text tl_login, ${LOGIN} — 00:00:01.546
Documentation: Types the given 'text' into text field identified by 'locator'.
Start / End / Elapsed: 20160418 04:15:36.871 / 20160418 04:15:38.417 / 00:00:01.546
04:15:36.871 INFO Typing text 'renat.kulalov' into text field 'tl_login'

KEYWORD Selenium2Library.Input Password tl_password, ${PASSWD} — 00:00:00.071
Documentation: Types the given password into text field identified by 'locator'.
Start / End / Elapsed: 20160418 04:15:38.418 / 20160418 04:15:38.489 / 00:00:00.071
04:15:38.418 INFO Typing password into text field 'tl_password'

KEYWORD Selenium2Library.Click Button login_submit — 00:00:00.500
Documentation: Clicks a button identified by 'locator'.
Start / End / Elapsed: 20160418 04:15:38.489 / 20160418 04:15:38.989 / 00:00:00.500
04:15:38.490 INFO Clicking button 'login_submit'.

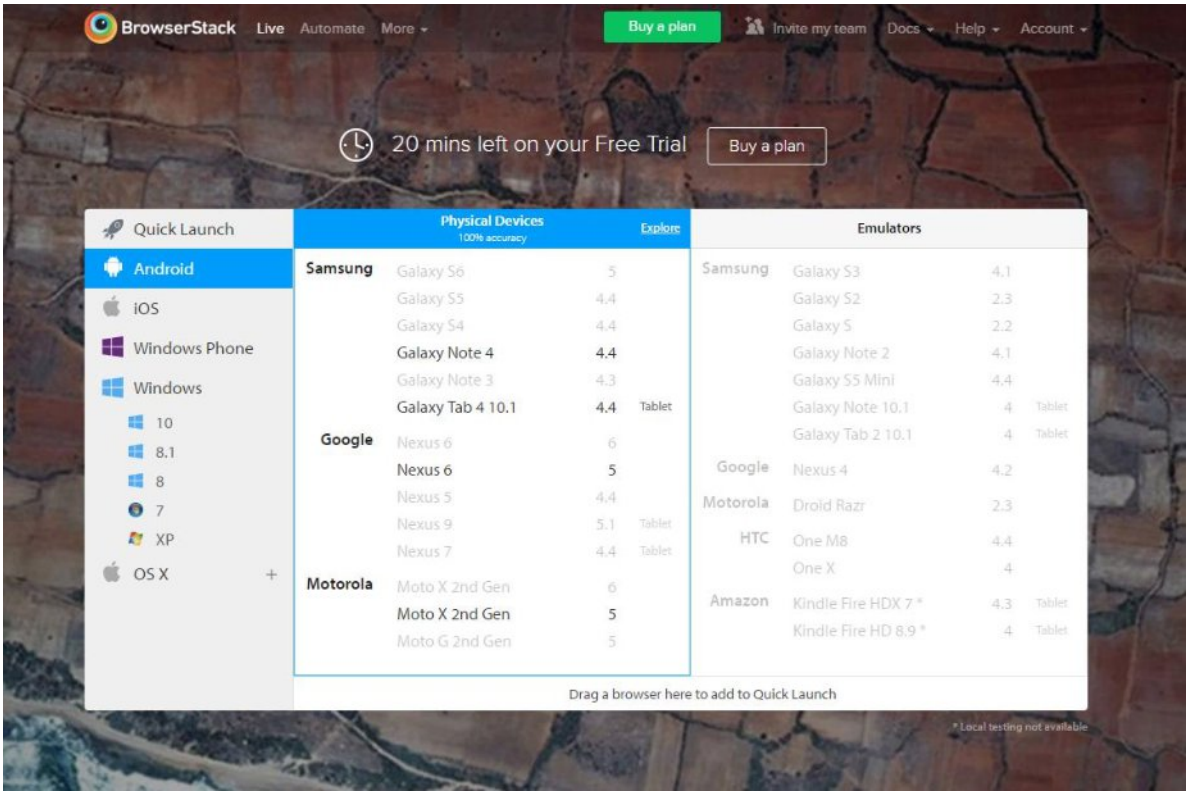
KEYWORD Selenium2Library.Title Should Be TestLink 1.9.14 (Padawan) — 00:00:00.232
Documentation: Verifies that current page title equals 'title'.
Start / End / Elapsed: 20160418 04:15:38.989 / 20160418 04:15:39.221 / 00:00:00.232
04:15:39.221 INFO Page title is 'TestLink 1.9.14 (Padawan)'.

TEARDOWN Selenium2Library.Close Browser — 00:00:00.116
Documentation: Closes the current browser.
Start / End / Elapsed: 20160418 04:15:39.222 / 20160418 04:15:39.338 / 00:00:00.116

Obrázek 4. Log testovacího případu z obrázku 1. Vlastní zpracování.

# 4. BrowserStack

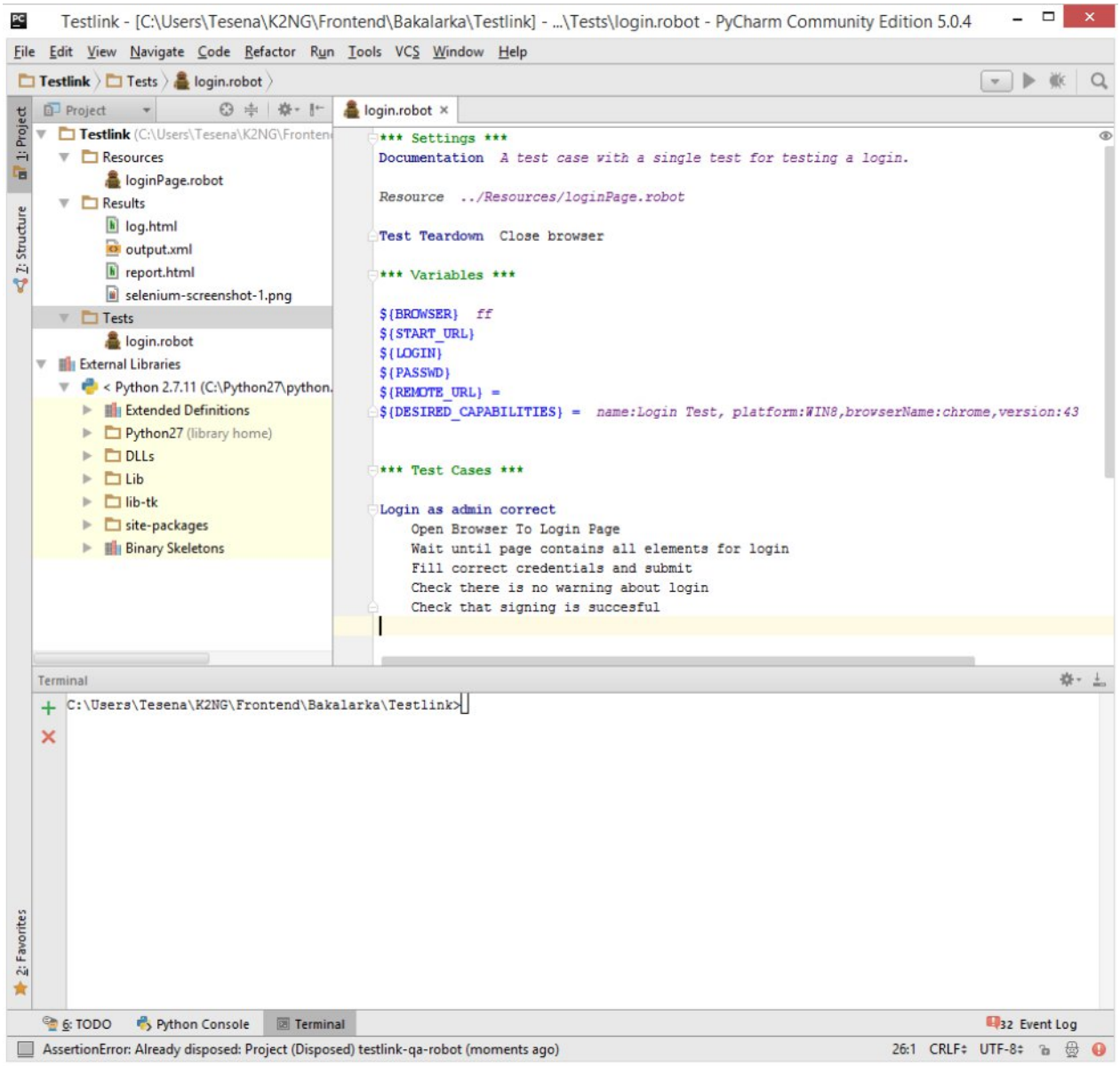
Při testování webových aplikaci je velmi důležité testovat na různých prohlížecích a operačních systémech. Kupovat počítače a mobilní zařízení s různými operačními systémy pro účely testování je velmi drahé. Proto jsou často využívané cloudove nástroje, které nabízejí velký počet různých konfiguraci pro testování. Jedním z takových nástrojů je BrowserStack. BrowserStack je online cloudovy cross-browser testovací nástroj založený v roce 2011, který umožňuje testování webových stránek a aplikací na různých operačních systémech a mobilních zařízeních. [11] [Obrázek 5]




Obrázek 5. Stránka pro vyber konfiguraci na webu BrowserStack.com. Vlastní zpracování.

# 5. Pycharm IDE

Pycharm IDE je vývojové prostředí od české firmy JetBrains, které se používá pro programování v jazyce Python. Pycharm podporuje veršování, debuggování kódu, Je to cross-platformovy nastroj, který podporuje práci v operačních systémech Windows, OS X a Linux. [Obrázek 6]




Obrázek 6. Pycharm Community Edition. Vlastní zpracování.

Pro účely této bakalářské práce je použita verze Community Edition, která je zcela zdarma a spuštěna pod licenci Apache2. Source kód je dostupný na webových stránkách GitHub.

# 6. Testlink

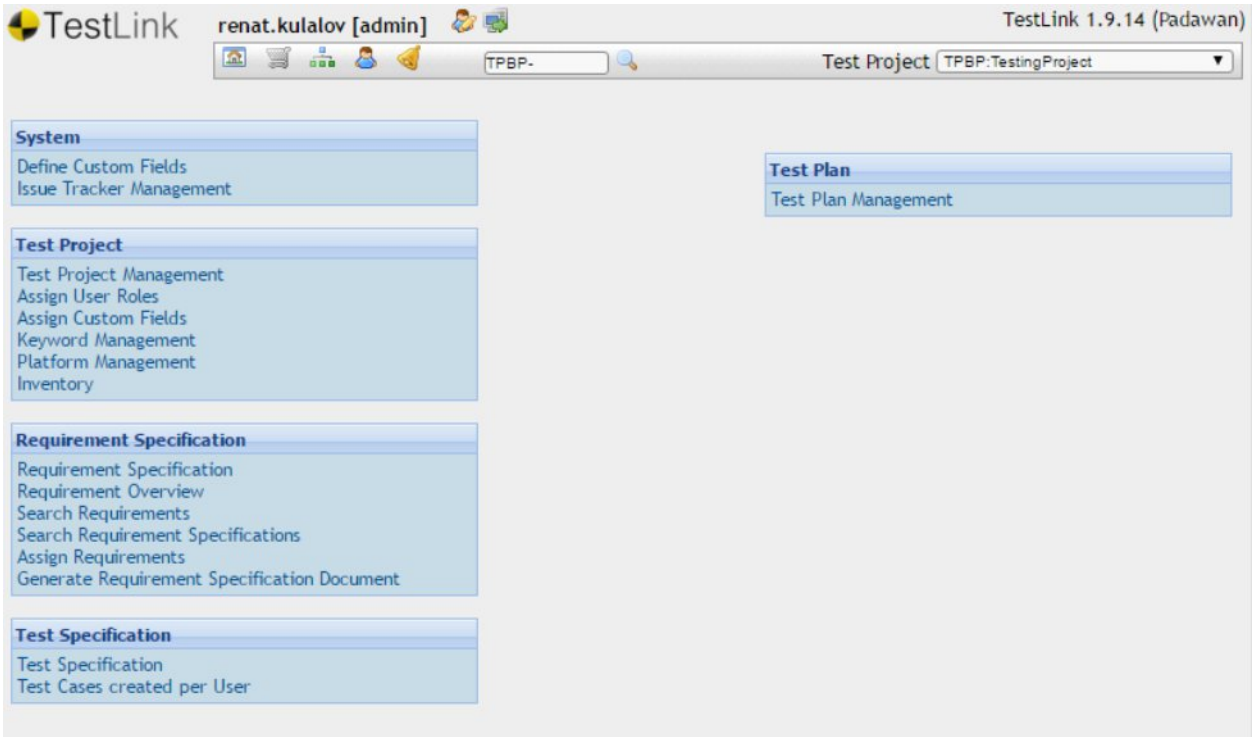
Testlink je zdarma a veřejně dostupný open source nástroj, který slouží pro správy a organizací testu. Uživatelé nepotřebují instalovat Testlink na své počítače, přistupuje se k nástroji přes webové prohlížeče.[12] [Obrázek 7]

Testlink umožňuje definování rolí pro uživatele, které definuji, jaké funkcionality budou uživatelům přístupné. Testlinik obsahuje komponenty které umožnují:[13]

- Vytváření plánu procesu testování
- Psaní testovacích scénářů
- Organizaci testovacích scénářů do hierarchické struktury
- Zaznamenávání softwarových požadavků
- Zapisování průběhů vykonávání testovacích scénářů
- Zápis výsledků proběhlého testování a jejich reportovaní

K dalším funkcionalitám nástroje Testlink patří možnost exportovat testovací případy z XML souboru, přiřazení testovacích případů určitým uživatelům, prioritizace testovacích případů, možnost exportovat reporty do formátu HTML, MS Word a MS Excel [13]

Testlink umožňuje spolupráci s programy pro správu chyb, takovými jako Bugzilla, JIRA a TrackPlus. [13]



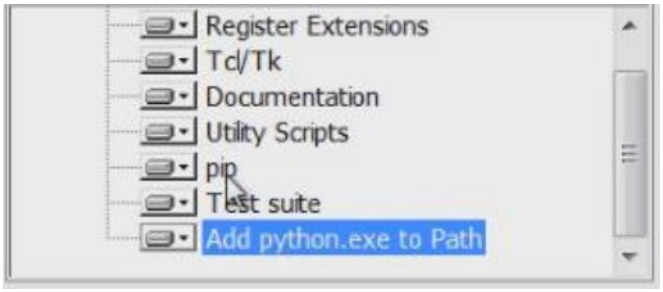

Obrázek 7. Úvodní obrazovka aplikace Testlink. Vlastní zpracování.

# 7. Testovací prostředí

V této kapitole je popsaná instalace nástroje Robot Framework, psaní a spouštění testu. Dále v této kapitole je ukázáno propojení nástroje Robot Framework s nástrojem BrowserStack a následně spouštění testovacích případů v BrowserStack.

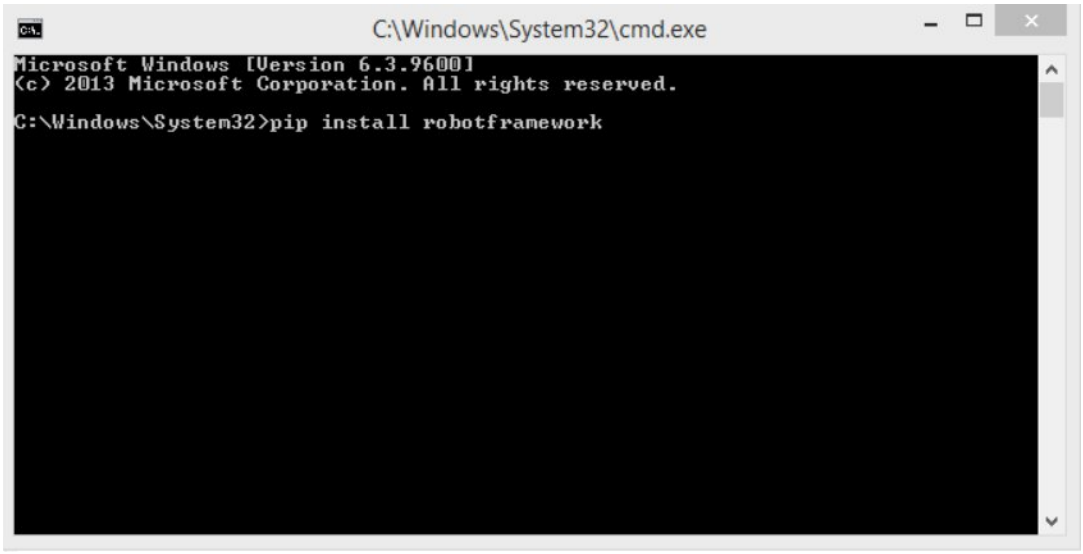
## 7.1. Instalace

Pro instalaci Robot Frameworku je třeba mít nainstalovaný Python. Instalace Pythonu je velice jednoduchá, příručka pro začátečníky je k dispozici zde [15] Při instalaci zvolíme moznost přidat python.exe to Path [Obrázek 8]




Obrázek 8. Zvoleni moznosti přidání python.exe do Path. Vlastní zpracování

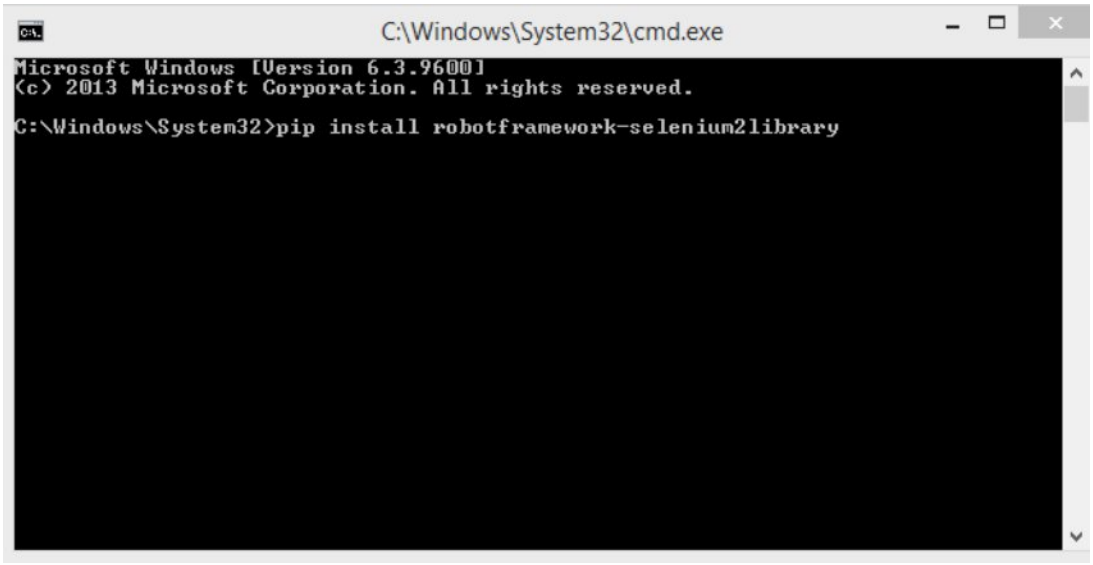
V předchozím bylo přidáno python.exe do Path, to nám umožňuje snadnější instalaci Robot Frameworku tím, že není třeba měnit složku v příkazovém řádku. Otevře se cmd.exe a napíše se do příkazového řádku: „pip install robotframework“. [Obrázek 9] Pak se zmáčkne tlačítko Enter a Robot Framework se nainstaloval. [15]




Obrázek 9. Instalace Robot Frameworku. Vlastní zpracování

Dalším krokem je instalace knihovny Selenium2Library. Podobně jako při instalaci Robot Frameworku se otevře příkazový řádek a do něho se napíše

„pip install robotframework-selenium2library“. [Obrázek 10] [15]



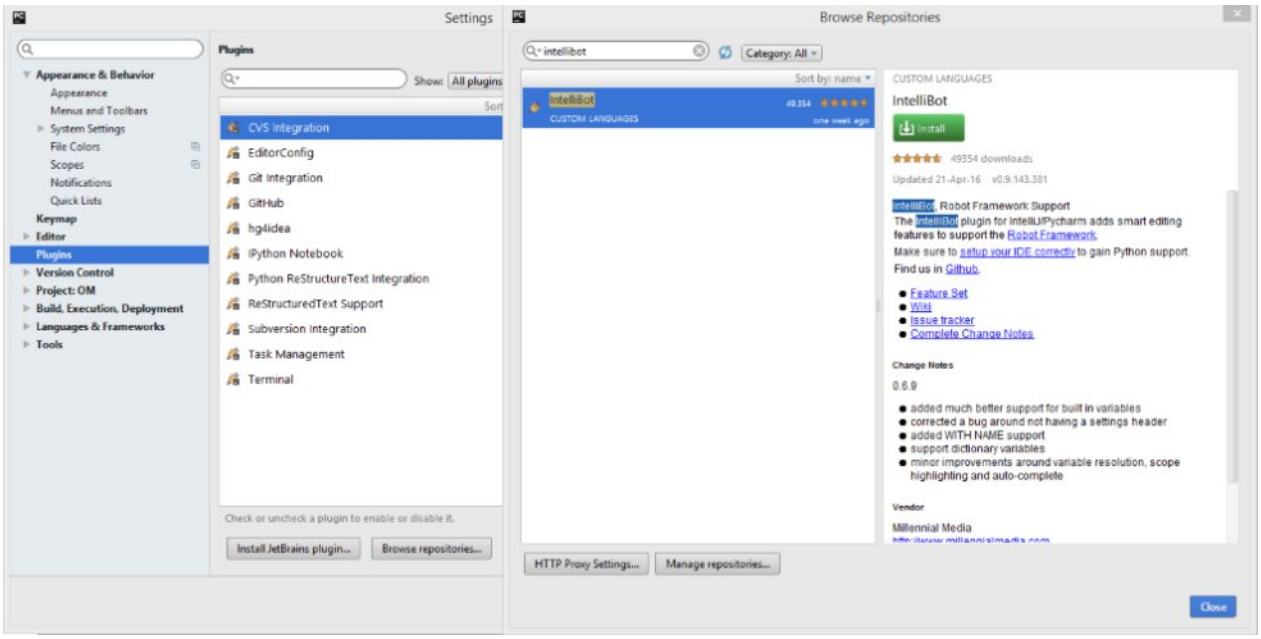

Obrázek 10. Instalace knihovny Selenium2Library. Vlastní zpracování

Jako vývojové prostředí pro Robot Framework je zvolen Pycharm IDE. Instalace je jednoduchá, návod je na [16] Po instalaci vývojového prostředí Pycharm IDE je nainstalován Intellibot plugin, účelem kterého je přidat chytré funkce editovaní v Robot Frameworku. [17]

Vy vývojovém prostředí Pycharm IDE kliknutím na

**File -> Settings->Plugins->Browse repositories**

otevřelo se okno vyhledávání pluginu. Do vyhledávacího políčka zadáno „intellibot“. Vyhledaný plugin je nainstalován potlačením tlačítka Install. [Obrázek 11]



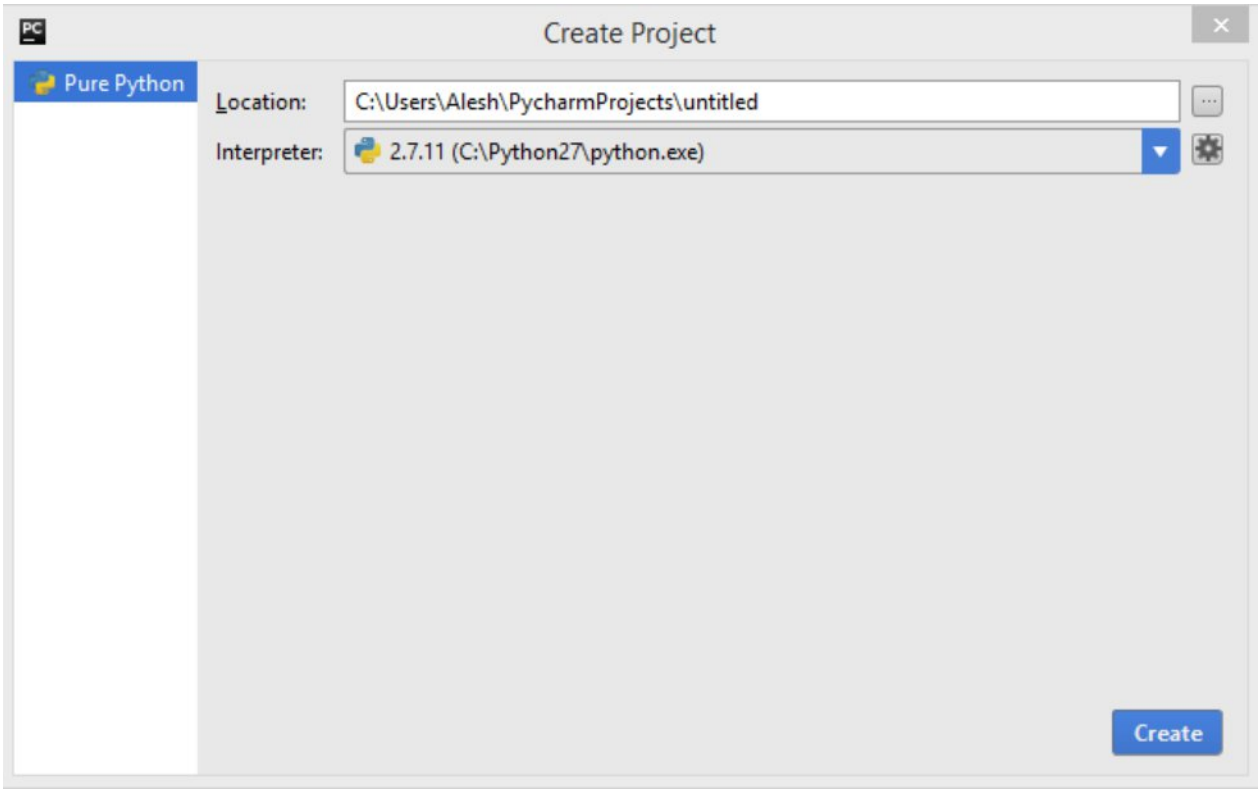

Obrázek 11. Instalace pluginu Intellibot. Vlastní zpracování.

Po všech předchozích krocích je Robot Framework plně nainstalován a připraven k použití.

## 7.2. Psaní testů

Na Obrázku 1 je ukázán příklad jednoduchého testovacího případu, v praxi ale je potřeba používat metodu Page Object Design Pattern[18], která ulehčuje údržbu testovacích případů tím, že při nutnosti úpravy testovacího případu, se upraví jenom metoda v Page Object souboru a tím se opraví všechny testovací případy, které se vztahuji s tuto metodou.

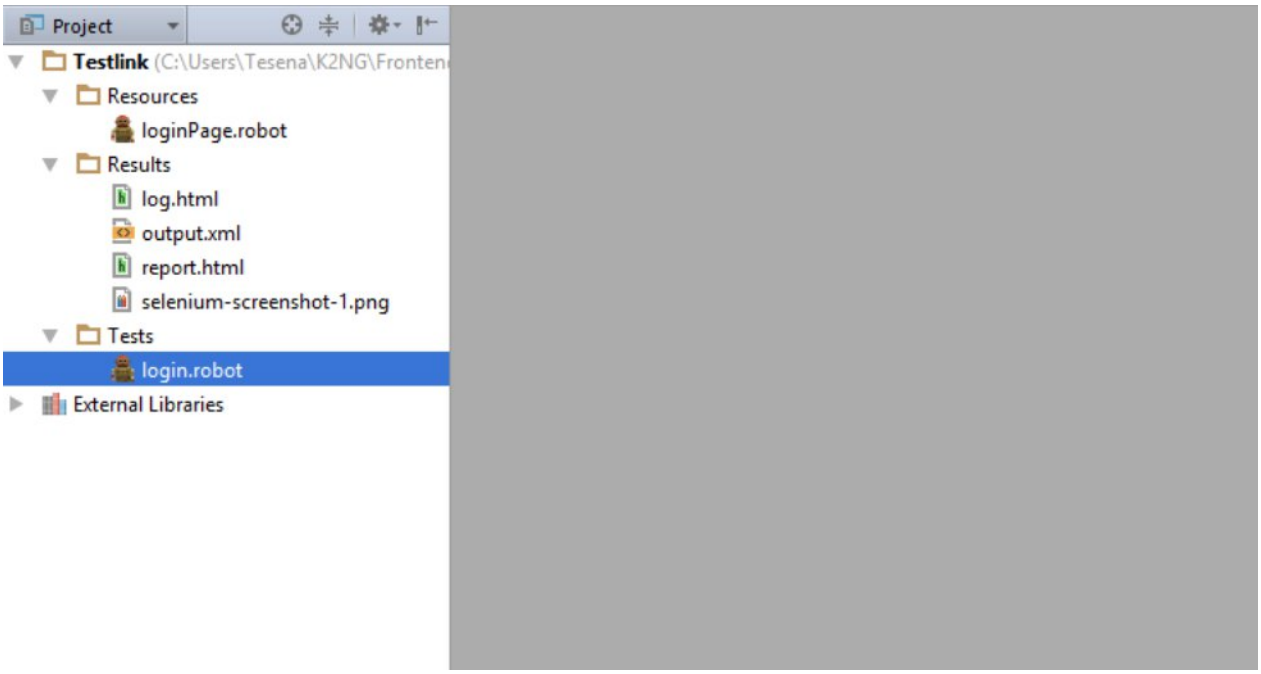
Jako první krok při testování pomoci nástroje Robot Framework je založení nového projektu. Novy projekt založíme kliknutím na **File->New Project…**. Otevřelo se nám okno Create Project [20] [Obrázek 12]




Obrázek 12. Okno vytváření nového projektu. Vlastní zpracování.

Do políčka Location je zadáváno místo uložení nového projektu. Kliknutím do tlačítka „Create“ projekt vytvoříme. [20]

Dále se v projektu vytvoří tři složky: Resources, Results a Tests. Složka Resources obsahuje Page Object soubory, složka Results – Logy a Reporty a složka Tests obsahuje testovací případy. [Obrázek 13]

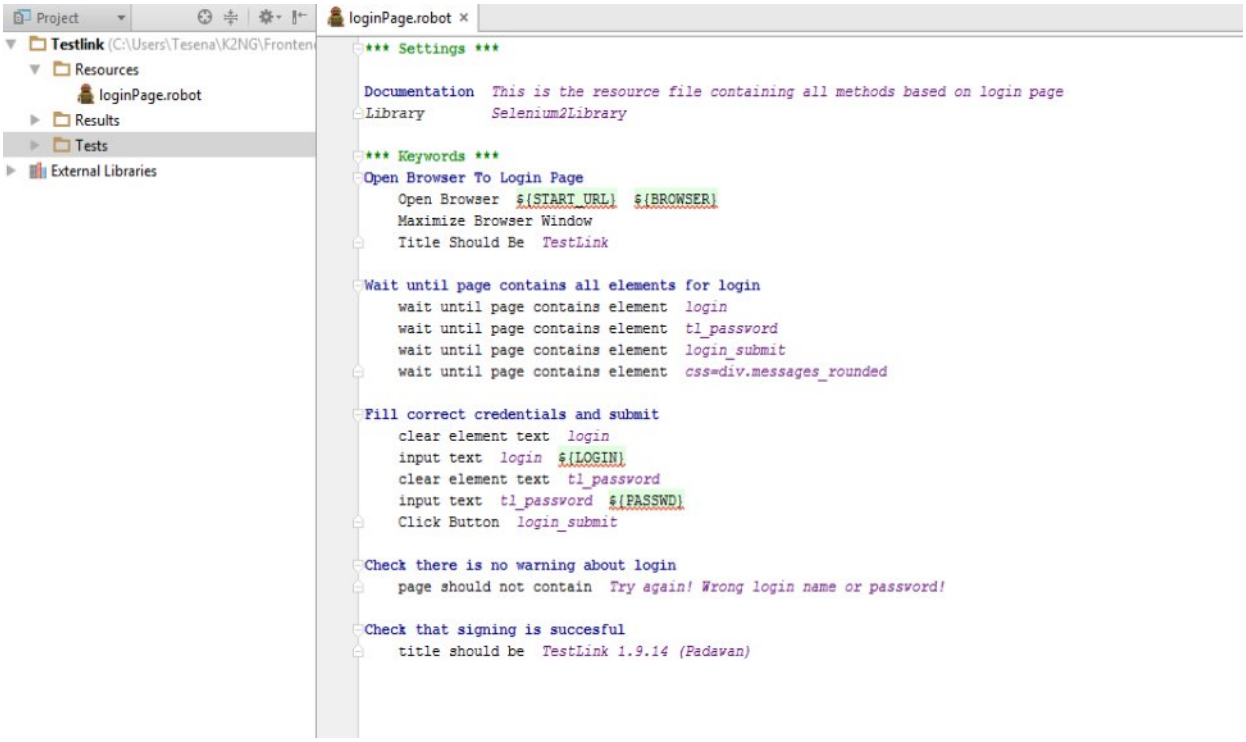



Obrázek 13. Rozmístěni souboru. Vlastní zpracování.

Podle ATDD před tím, než se začne testovací případ psát, je třeba zadokumentovat případ užití. Případ užití pro přihlašování do Testlinku:[24]

1. Otevřít přihlašovací stránku aplikace Testlink
2. Zkontrolovat, že všechny elementy na stránce jsou přítomny
3. Zadat přihlašovací údaje a kliknout na tlačítko login
4. Zkontrolovat, jestli nevyskočila chybová hláška při přihlašování
5. Zkontrolovat, ze přihlášení proběhlo úspěšné

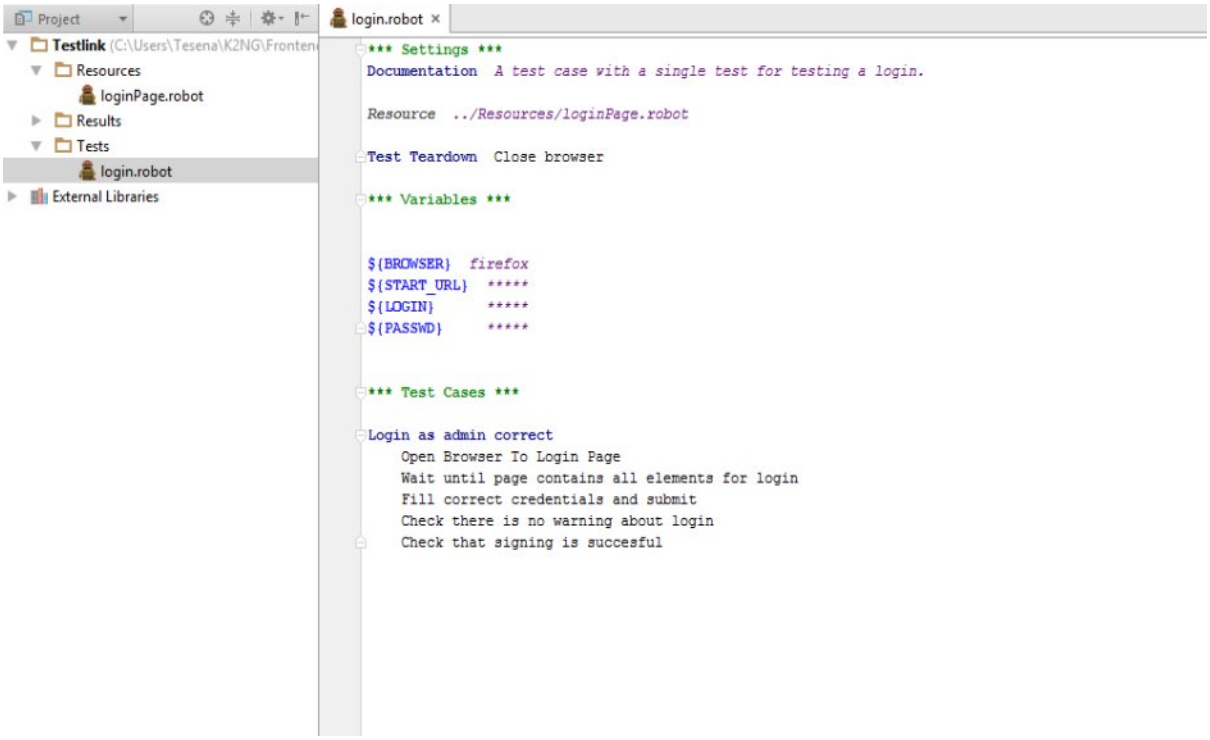
Pro každý krok případu užití v Page Object souboru loginPage.robot se vytvoří odpovídající metoda [15] [Obrázek 14]




Obrázek 14. Metody v Page Object souboru loginPage.robot. Vlastní zpracování

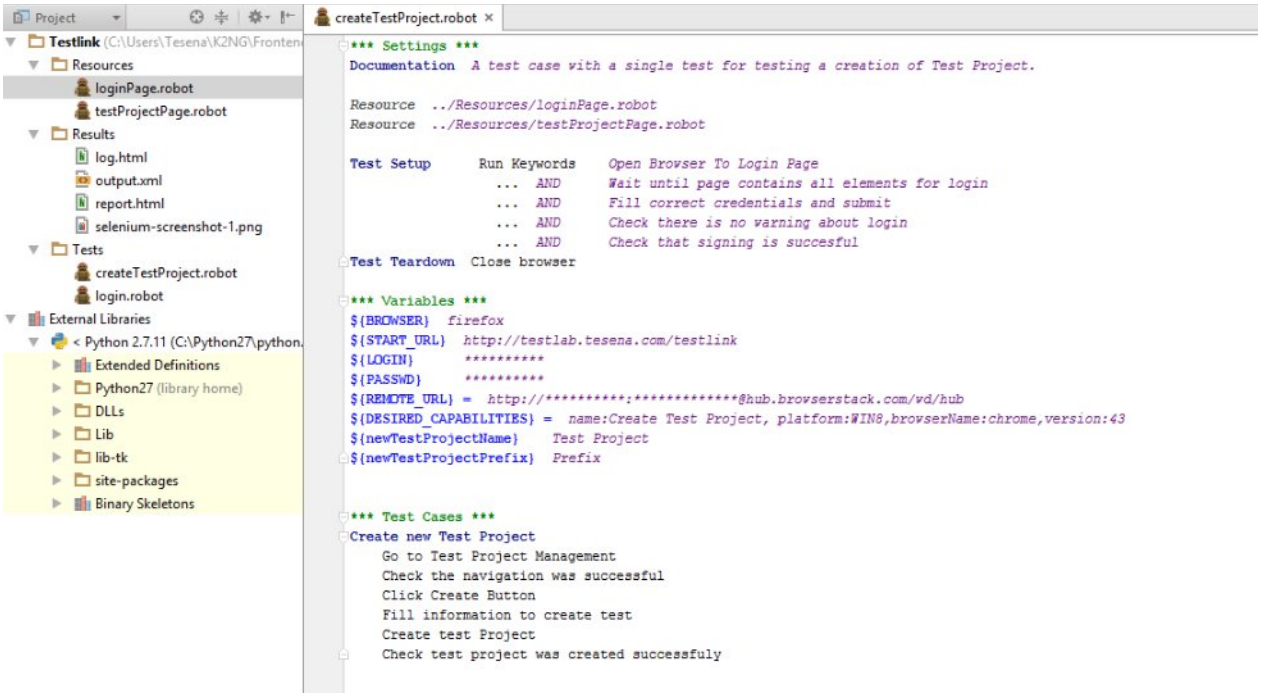
V sekci ***Settings*** zadaná dokumentace k Page Object souboru a zavolaná knihovna Selenium2Library. Sekce ***Keywords*** obsahuje metody odpovídající případu užití. Metody byly napsané použitím klíčových slov, identifikace elementu provedena použitím lokátorů. Přihlašovací údaje a adresa webové aplikace Testlink jsou vyměněny proměnnými, které jsou definované v souboru testovacího případu. [15]

Psaní samotného testovacího případu se skládá z vyvolání metod ze souboru loginPage.robot. [Obrázek 15]




Obrázek 15. Testovací případ Login. Vlastní zpracování

Jako minule v sekci *** Settings*** zadaná dokumentace testovacího případu ale není volaná knihovna Selenium2Library. Místo knihovny je zavolán soubor loginPage.robot, ve kterém jsou napsané metody. V sekci *** Variables*** jsou definované všechny proměnné, které byly použity při psaní metod v souboru loginPage. Sekce *** Test Cases*** obsahuje název testovacího případu a pod nim metody volané ve správném pořadí. [15]




Obrázek 16. Testovací případ Create Test Project. Vlastní zpracování

Na obrázku 16 je testovací případ pro testování vytvoření testovacího projektu v aplikaci Testlink. Na rozdíl od testovacího případu pro přihlašování [Obrázek 15] máme v sekci ***Settings*** nastaven Test Setup, ve kterém vyvoláváme metody ze souboru loginPage.robot.

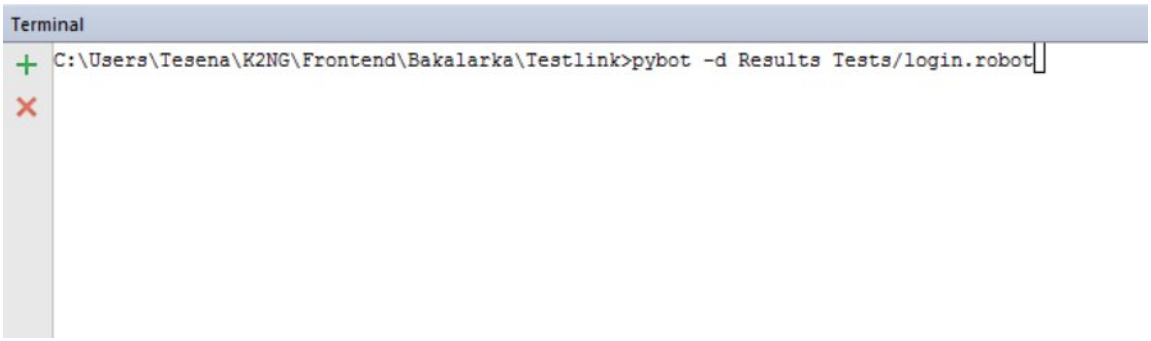
Test Setup vždycky běží před spouštění testovacího případu a Test Teardown vždycky po ukončení běhu testu. [Obrázek 16]

## 7.3. Spouštění testu

Napsané testovací případy mohou byt spouštěny z příkazového řádku vývojového prostředí Pycharm IDE. Pro otevření příkazového řádku je třeba kliknout na

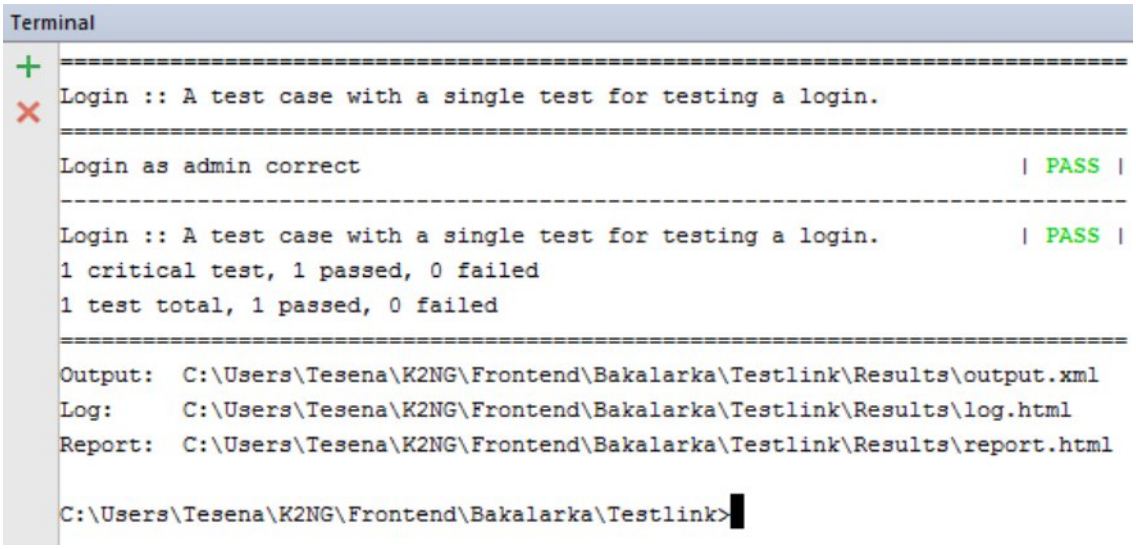
**View -> Tool Windows-> Terminal**

Testovací případy jsou spuštěny příkazem „pybot“, dále lze konfigurovat, kam se ukládají Reporty a Logy a to je přidáním „-d nazevSlozky”. [15] [Obrázek 17]

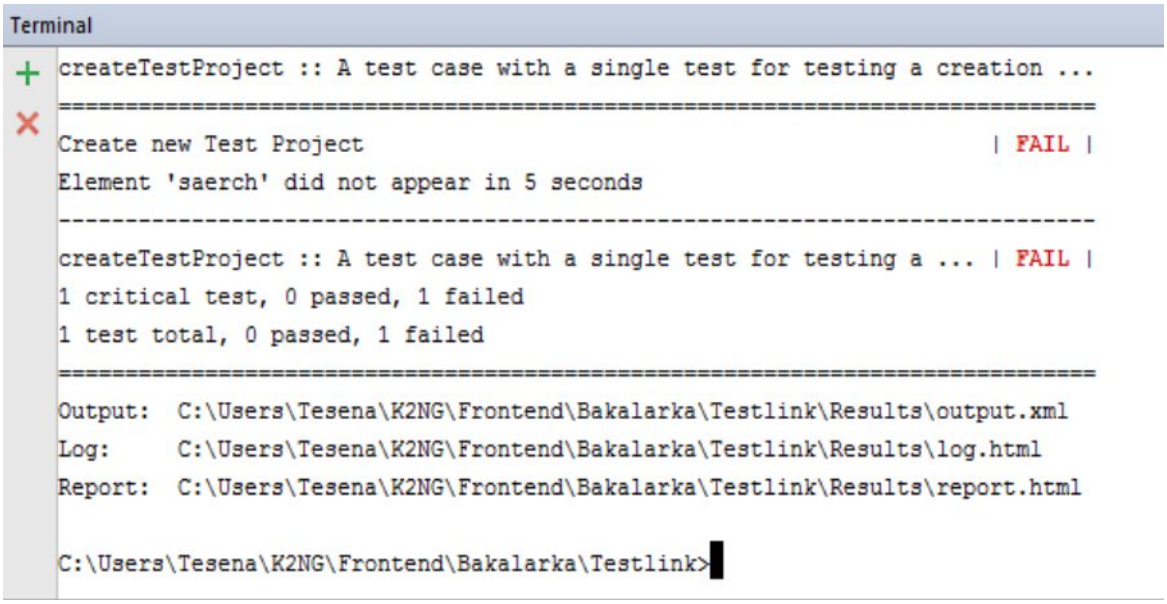



Obrázek 17. Spouštěni testovacích případů. Vlastní zpracování

Test je spouštěn tlačítkem Enter. Po ukončení běhu testovacího případu v příkazovém řádku je zobrazen výsledek běhu testovacího případu, PASS nebo FAIL. [15] [Obrázek 18] [Obrázek 19]



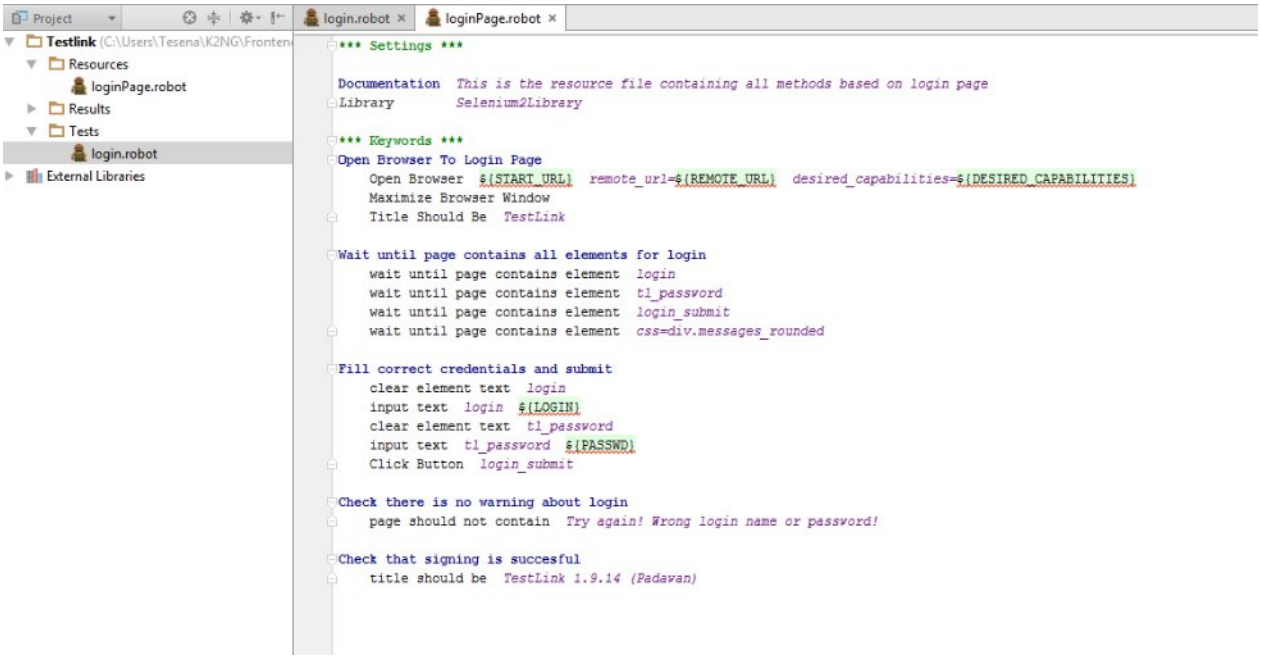

Obrázek 18. Výsledek běhu testovacího případu [PASS]. Vlastní zpracování.




Obrázek 19. Výsledek běhu testovacího případu [FAIL]. Vlastní zpracování.

## 7.4. Propojení s BrowserStack

Pro propojení nástroje BrawserStack a Robot Frameworku je třeba se zaregistrovat na webové stránce Browserstack. Použitím unikátního AccessKey a jména uživatele se propojí Browserstack s nástrojem Robot Framework. V souboru loginPage.robot se upraví metoda „Open Browser To Login Page“. Pro klíčové slovo „Open Browser“ se zadají parametry „remote_url» a „desired_ccapabilities“. [19] [Obrázek 20]



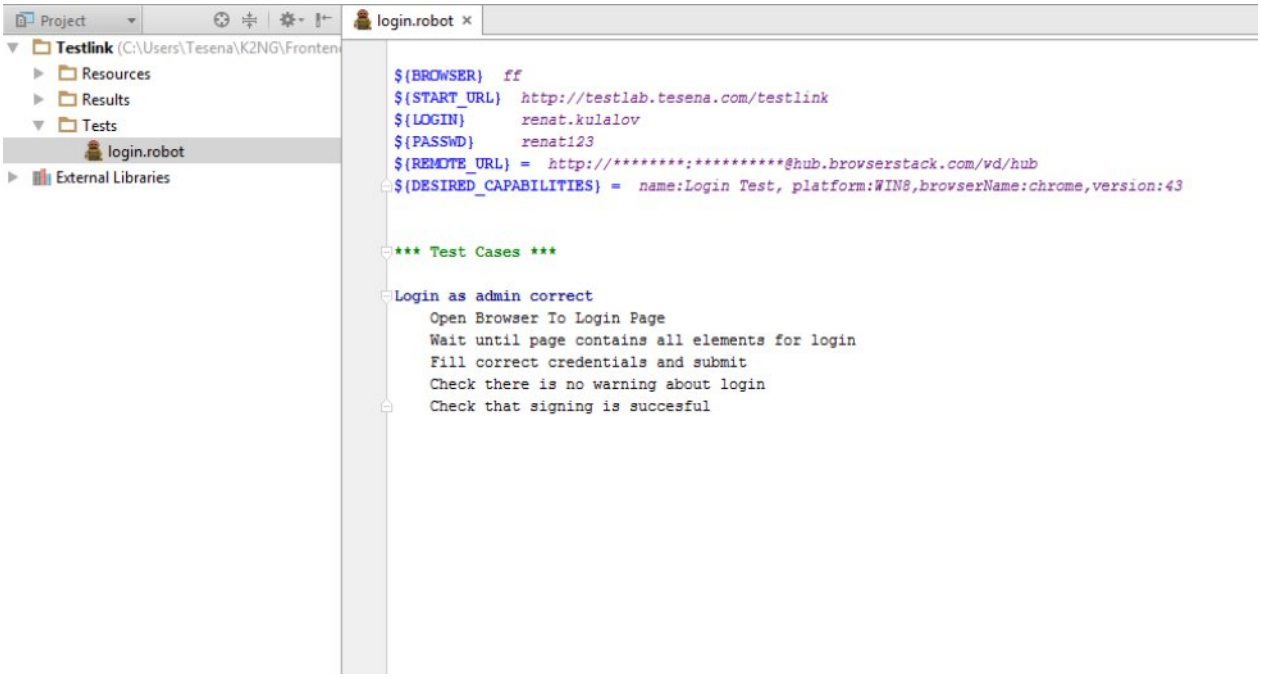

Obrázek 20. Úprava parametru klíčového slova „Open Browser“. Vlastní zpracování

Parametr „remote_url“ se skládá z unikátního AccessKey a jména uživatele ve tvaru: **http://jmenouzivatele:AccessKey@hub.browserstack.com/wd/hub**

Parametr „desired_ccapabilities“ je zadán ve tvaru:

**name:JmenoTestovacihoPripadu, platform:NazevOS,browserName:chrome,version:43**

[19] [Obrázek 21]

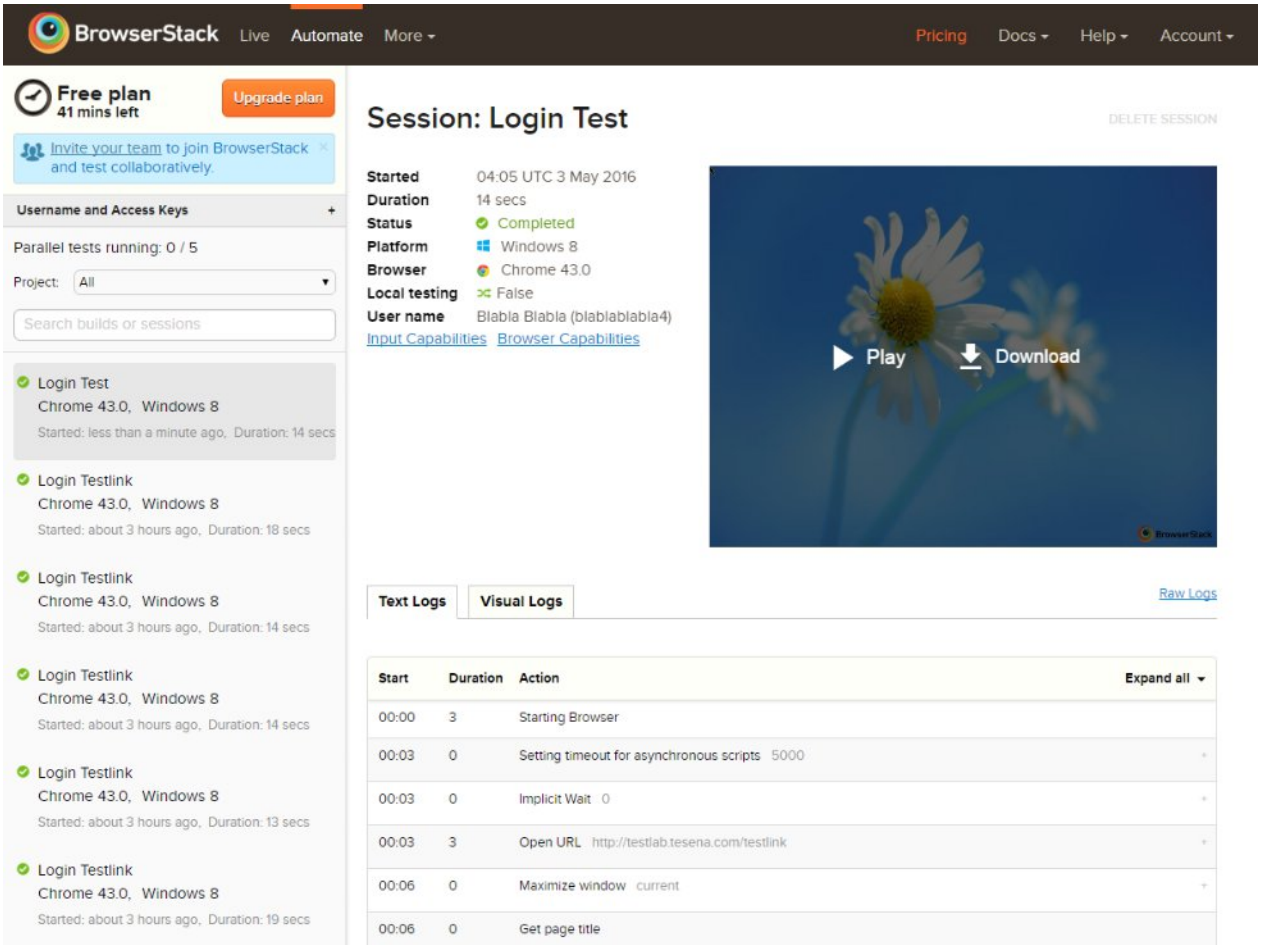



Obrázek 21. Definování proměnných pro propojení s BrowserStack. Vlastní zpracování

## 7.5. Spouštění automatizovaných testu v BrowserStack

Spouštěni testovacích případu v Browserstack probíhá stejně jako spouštění popsané v kapitole 6.3. Do příkazového řádku je zadán příkaz „pybot“ s parametrem pro uložení Reportu a Logu a ukáže se, který testovací případ je spouštěn.

Po spouštění testovacího případu se otevře záložka „Automate“ na webové stránce nástroje BrowserStack. [Obrázek 22]




Obrázek 22. Záložka „Automate“ na webové stránce nástroje BrowserStack. Vlastní zpracování

Z levé strany jsou zobrazeny všechny testovací případy, které byly spuštěny. Po kliknutí na testovací případ vpravo se zobrazuje výsledek běhu testovacího případu. Velkou výhodou je to, ze BrowserStack nahrává video po celou dobu běhu testovacího případu.

# 8. Závěr

Tato bakalářská práce se zabývá testováním webových aplikaci pomoci nástroje Robot Framework. Teoretická část obsahuje úvodní informaci o automatizovaném testování softwaru, o přínosech automatizace testovacích případů a dále pojednává o vybraných nástrojích pro automatizované testování. V praktické části byl představen nástroj Robot Framework, jeho instalace a psaní automatizovaných testu. Po té bylo popsáno spouštění testovacích případů pomoci nástroje BrowserStack.

Hlavním cílem bakalářské práce je představeni nástroje Robot Framework a vytvoření příručky v českém jazyce, která umožnuje začít automatizovat testovací případy a je využitelná při výuce testování softwaru na Vysoké Škole Ekonomické v Praze.

K dosáhnutí hlavního cíle bylo nainstalováno testovací prostředí, které obsahuje nástroje Pycharm IDE a Robot Framework, testovanou webovou aplikaci je Testlink a také je použit nastroj BrowserStack, umožňující pokrytí velkého počtu konfiguraci. Použitím testovacího prostředí bylo demonstrováno psaní automatizovaných testovacích případů. Tím byla vytvořena příručka, která umožni začít automatizovat testovací případy a bylo tak dosaženo cílů bakalářské práce.

# Terminologický slovník

| Termín | Zkratka | Význam [zdroj] |
|---|---|---|
| Software | | Vybavení počítače (nehmotné) [Vlastní definice] |
| Path | | Proměnná, která obsahuje cesty ke složkám se spustitelnými soubory [Vlastní definice] |
| Webova aplikace | | Je jakákoliv aplikace, která používá webový prohlížeč jako klient[21] |
| Page Object Design Pattern | PODP | PODP je návrhový vzor, který může být implementován jako nejlepší praxe při psaní testovacích případů. [18] |
| Open Source | | Open source (někdy také open-source) je označení programů, jejichž zdrojový kód byl poskytnut dalším vývojářům, kteří jej mohou studovat a většinou i upravovat a dále vylepšovat.[22] |
| Metoda | | Metoda je blok kódu obsahující sled příkazů [23] |
| Acceptance Test-Driven Development | ATDD | ATDD je metodika vývoje, která je založena na komunikaci mezi zákazníky, vývojáři a testery.[24] |

| | | |
|---|---|---|
| Framework | | Framework je knihovna, která ulehčuje práci při programování. [25] |

# Výchozí literatura a další informační zdroje:


**[1] Laukkanen, Pekka.** *Data-Driven and Keyword-Driven Test Automation Frameworks.* Master's thesis. HELSINKI UNIVERSITY OF TECHNOLOGY, 2006.

**[2] Dustin, Elfriede, Garrett, Thom a Gauf, Bernie.** *Implementing automated software testing: how to save time and lower costs while raising quality.* Upper Saddle River, NJ: Addison-Wesley, c2009. ISBN 9780321580511.

**[3] Automatizovane testovani.** Testování softwaru. [Online] [Citace: 15. duben 2016.] Dostupne z: *http://testovanisoftwaru.cz/automatizovane-testovani/*

**[4] Luostarinen, Jani.** *Web Application Test Automation with Robot Framework.* Bachelor's thesis. HELSINKI METROPOLIA UNIVERSITY OF APPLIED SCIENCES, 2015.

**[5] Robot Framework.** Indtroduction**.** [Online] [Citace: 20. duben 2016.] Dostupne z: *http://robotframework.org/#introduction*

**[6] Google Code.** Robot Framework. [Online] [Citace: 22. duben 2016.] Dostupne z: *https://code.google.com/archive/p/robotframework/downloads*

**[7]User Guide.** Robot Framework. [Online] [Citace: 22. duben 2016.] Dostupne z: *http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#test-data-syntax*

**[8] Robot Framework.** Selenium2Library. [Online] [Citace: 23. duben 2016.] Dostupne z: *http://robotframework.org/Selenium2Library/doc/Selenium2Library.html*

**[9] Selenium.** Webdriver. [Online] [Citace: 23. duben 2016.] Dostupne z: *http://docs.seleniumhq.org/docs/03_webdriver.jsp*

**[10] Robot Framework. Examples.** [Online] [Citace: 24. duben 2016.] Dostupne z: *http://robotframework.org/#examples*

**[11]BrowserStack.** Features. [Online] [Citace: 24. duben 2016.] Dostupne z: *https://www.BrowserStack.com/features*

**[12] Spoken Tutorial. Testlink.** [Online] [Citace: 24. duben 2016.] Dostupne z: *http://script.spoken-tutorial.org/index.php/Testlink*

**[13] Smatanova, Klara.** Diplomová práce. *Automatizované testování webových aplikací.* UNIVERZITA HRADEC KRALOVE, 2015.

**[14] LogiGEAR MAGAZINE.** Key Success Factors for Keyword Driven Testing. [Online] [Citace: 25. duben 2016.] Dostupne z:

*http://www.logigear.com/magazine/test-process-improvement/key-success-factors-for-keyword-driven-testing/*

**[15]Robot Framework.** User Guide. [Online] [Citace: 25. duben 2016.] Dostupne z: *http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html*

**[16] JetBrains.** Pycharm IDE. [Online] [Citace: 25. duben 2016.] Dostupne z: *https://www.jetbrains.com/help/pycharm/2016.1/installing-and-launching.html?origin=old_help*

**[17] GitHub.** Intellibot. [Online] [Citace: 26. duben 2016.] Dostupne z: *https://github.com/millennialmedia/intellibot*

**[18] Selenium.** Page Object Design Pattern. [Online] [Citace: 26. duben 2016.] Dostupne z: *http://www.seleniumhq.org/docs/06_test_design_considerations.jsp#page-object-design-pattern*

**[19] SW Test Academy.** BrowserStack and RobotFramework Integration. [Online] [Citace: 27. duben 2016.] Dostupne z:

*http://www.swtestacademy.com/browserstack-robotframework-integration/*

**[20] JetBrains.** Pycharm [Online] [Citace: 30. duben 2016.] Dostupne z: *https://www.jetbrains.com/help/pycharm/2016.1/meet-pycharm.html*

[21] **Web trends.** Web application. [Online] [Citace: 30. duben 2016.] Dostupne z: *http://webtrends.about.com/od/webapplications/a/web_application.htm*

[22] **Adaptic.** Open Source. [Online] [Citace: 30. duben 2016.] Dostupne z: http*://www.adaptic.cz/znalosti/slovnicek/open-source/*

[23] **Microsoft.** Microsoft developer network. *Metody (Průvodce programováním v C#).* [Online] [Citace: 30. duben 2016.] *https://msdn.microsoft.com/cs-cz/library/ms173114.aspx.*

[24] **Pugh, Ken**. *Lean-Agile Acceptance Test-Driven Development: Better Software Through Collaboration.* Addison-Wesley, 2011. ISBN 978-0321714084.

[25] **Zdrojak.** Framework. [Online] [Citace: 30. duben 2016.] Dostupne z: *https://www.zdrojak.cz/clanky/nette-framework-zvyste-svoji-produktivitu/*

# Seznam obrázků