

MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY



Funkční testování aplikace pro výuku zeměpisu

BAKALÁŘSKÁ PRÁCE

Erik Sýkora

Brno, Podzim 2015

Místo tohoto listu vložte kopie oficiálního podepsaného zadání práce a prohlášení autora školního díla.

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Erik Sýkora

Vedoucí práce: Mgr. Jan Papoušek

Poděkování

Rád bych zde poděkoval zejména svému vedoucímu Mgr. Janu Papouškovi za cenné rady a odbornou pomoc při tvorbě této práce. Dále chci také poděkovat své rodině za jejich podporu a pochopení během tvorby této práce.

Shrnutí

This is the abstract of my thesis, which can span multiple paragraphs.

Klíčová slova

funkční testování, webové aplikace, Java, Arquillian Graphene, automatizace testování...

Obsah

1	Úvod	1
2	Testování webové aplikace a použité přístupy	3
2.1	Typy testování	3
2.1.1	Funkční testování	3
2.1.2	Unit testování	3
2.1.3	Statické a dynamické testování	4
2.1.4	Regresní testování	4
2.2	Použité metody a postupy	4
2.2.1	Testování černé, šedé a bílé skřínky	4
2.2.2	Validace a verifikace	5
2.2.3	Automatizace testování	5
3	Aplikace Slepé Mapy a její analýza	7
4	Technologie pro funkční testování webové aplikace	9
4.1	Selenium	9
4.1.1	Selenium WebDriver	10
4.1.2	Selenium IDE	10
4.2	Arquillian	10
4.2.1	Arquillian Graphene	10
4.2.2	Arquillian Drone	11
4.3	Ostatní nástroje	11
4.3.1	JUnit	11
4.3.2	Maven	11
4.3.3	Travis CI	11
5	Tvorba testů	13
6	Systém automatizovaného testování	15
7	Závěr	17

Seznam tabulek

Seznam obrázků

1 Úvod

Vývoj softwaru je komplexní proces sestávající z mnoha kroků. Tento proces se liší dle použitého vývojového modelu, nicméně vždy by měl zahrnovat určitou formu testování. Náklady na opravu chyby totiž strmě rostou při jejím pozdním nalezení, v případě kritického softwaru může nešetřená chyba dokonce způsobit ztráty na životech.

Webové aplikace jsou dnes velice rozšířeným druhem softwaru. Testování webové aplikace zahrnuje spoustu oblastí, jmenovitě například funkční testování, testování kompatibility či validace a verifikace. Tyto způsoby testování se běžně používají pro testování softwaru obecně, nicméně testování webových aplikací má svá specifika. Metodami testování použitými pro testování konkrétní webové aplikace se bude zabývat druhá kapitola.

Představené postupy budou následně aplikovány při testování aplikace pro výuku zeměpisu – Slepé mapy¹. Jelikož tato aplikace postrádá specifikaci softwaru, bude důležitou částí práce analýza aplikace, čemuž bude věnována třetí kapitola. Tato práce je zaměřena na funkční testování, proto zde bude také popsáno chování a funkčnost aplikace, jež bude následně otestována.

Pro tvorbu samotných testů je možno vybrat ze široké škály nástrojů. Pro testování aplikace Slepé mapy byla vybrána testovací platforma Arquillian a rozšiřující plugin Graphene. Tato platforma si zaslouží podrobnější popis, proto se čtvrtá kapitola bude zabývat představením tohoto a dalších použitých nástrojů. Na získaných znalostech o použitých technologiích bude stavět pátá kapitola, ve které bude vysvětlen postup při tvorbě testů. Vysvětlen bude kompletní postup použitý při testování webové aplikace Slepé mapy, a to od počáteční tvorby testového plánu, rozdělení aplikace na testovatelné části, návrh jednotlivých testových případů a následně jejich realizace pomocí výše zmíněných technologií.

Hotové testy budou spouštěny automaticky při aktualizování aplikace. Jelikož by bylo nepraktické po každé aktualizaci testy spouštěn manuálně, bude součástí práce automatizace testové sady. Způsob realizace automatického testování bude popsán v šesté kapitole.

1. <http://slepemapy.cz/>

1. Úvod

Závěrem budou vyhodnoceny výsledky testování. Součástí výstupů této práce bude také zhodnocení současného stavu aplikace na základě pokrytí testovými sadami. Vytvořená dokumentace společně s testovou sadou je dostupná v příloze.

2 Testování webové aplikace a použité přístupy

Testování softwaru je komplexní proces, jehož cílem je v programu či aplikaci nalézt softwarové chyby a ověřit, zda splňuje požadavky na něj kladené. Chybu lze definovat jako neočekávané chování programu, lišící se od specifikace softwaru [4]. Jelikož je téměř nemožné v rozumném čase kompletně otestovat i jednoduchý program, je třeba se při testování zaměřit na určité části aplikace a použít vhodné postupy. V této kapitole budou představena specifika testování webové aplikace a přístupy použité při jejich testování. Všechny zde uvedené postupy a metody byly využity při testování konkrétní aplikace – Slepé mapy.

2.1 Typy testování

2.1.1 Funkční testování

Cílem funkčního testování je otestování správného chování funkcí části aplikace či aplikace jako celku. Chování softwaru může být popsáno ve specifikaci. Pokud není specifikace k dispozici, tak je testováno předpokládané chování aplikace [3]. Během funkčního testování tester nahlíží na aplikaci očima uživatele a jeho úkolem je ověřit, zda aplikace funguje správně. Testovány mohou být například přechody mezi jednotlivými stránkami aplikace, ošetření zápisu neplatných hodnot do formulářů nebo ošetření chybových stavů aplikace.

2.1.2 Unit testování

Tento typ testování využívá malých, testovatelných částí aplikace. Touto takzvanou jednotkou mohou být například funkce, třídy, procedury či rozhraní. Cílem jednotkového testování je izolovat části programu a otestovat, zda splňují svou funkci a je možné je použít v celku, jehož jsou součástí. Výhodou tohoto přístupu je jednodušší objevení případných chyb v aplikaci. Jednotkové testování navíc bývá prováděno v ranných fázích vývoje softwaru, proto je také objevení chyb v programu výrazně levnější [5].

2.1.3 Statické a dynamické testování

Těmito pojmy lze rozlišit dvě základní možnosti, jakými můžeme k testování softwaru přistupovat. Statickým testováním je myšleno testování nespustitelné části programu, jde převážně o zkoumání dokumentace a specifikace softwaru. Tento přístup lze aplikovat ještě dříve, než je vytvořena spustitelná verze softwaru. Dynamické testování naopak vyžaduje spustitelný kód, na základě jehož chování je software testován. [2] Jako typ dynamického testování bychom mohli označit i funkční testování či jednotkové testování.

2.1.4 Regresní testování

Po opravě chyb v softwaru se může stát, že sice byla chyba opravena, ale jako vedlejší efekt se objevila chyba na jiném místě. Proces opakování spouštění testů při změně softwaru se nazývá regresní testování. Jeho cílem je zjistit, zda byly dříve odhalené chyby opraveny a zda jejich oprava nezanesla do kódu nové chyby. Regresní testování může probíhat manuálně, často ovšem bývá automatizováno. Manuální spouštění velkého množství testů při každé změně softwaru je značně nepraktické, a v případě komplexního softwaru nereálné provést v rozumném čase. [6]

2.2 Použité metody a postupy

2.2.1 Testování černé, šedé a bílé skřínky

Testování softwaru lze dělit pomocí pojmů testování černé, bílé a šedé skřínky. Při postupu testování černé skřínky má tester k dispozici pouze popis funkčnosti aplikace, nemůže využít zdrojového kódu softwaru. Lze tedy nesprávnou funkčnost pouze odpozorovat dle odlišného chování od specifikace či předpokládaného chování. Pokud lze k testování využít i zdrojový kód programu, jedná se o testování bílé skřínky. V tomto případě lze zdrojový kód využít pro důslednější tvorbu testů. Třetí možností je testování šedé skřínky, která kombinuje předchozí dva postupy. Nejprve jsou testy navrhovány z pohledu uživatele (testování černé skřínky), následně je ale k návrhu testů využit také zdrojový kód (testování bílé skřínky) [1]. Postup testování šedé

skříňky lze díky dostupnosti zdrojového kódu každé webové stránky využít i při testování webových aplikací.

2.2.2 Validace a verifikace

Oba tyto pojmy lze přeložit jako ověřování či kontrola, nicméně význam těchto dvou výrazů se liší a pro účely testování je tento rozdíl velmi důležitý. Verifikace je proces, jehož cílem je potvrzení, že testovaný software vyhovuje své specifikaci. Účelem procesu validace je kontrola, zda testovaný software vyhovuje požadavkům uživatele. V případě, že byla specifikace softwaru vytvořena nesprávně a nevyhovuje původním požadavkům uživatele, může dojít k situaci, kdy software sice splňuje verifikaci, ale validaci nikoliv [2]. Jelikož testovaná aplikace nemá vlastní specifikaci, tak byla v této práci využívána převážně validace.

2.2.3 Automatizace testování

Často se stává, že napsané testy bude nutné spustět vícekrát, například v případě regresního testování. Nástroje automatizace testovacího procesu jsou vhodným řešením pro tyto situace. Mezi jejich hlavní výhody patří například rychlejší provádění testů, než je možné manuálně. Dále je testování efektivnější, jelikož při manuálním testování není možné provádět jinou činnost. Nespornou výhodou je také správnost a přesnost. Člověk provádějící manuální testování může provést v testu chybu, zatímco testovací nástroj je vždy ve své práci konzistentní [2].

3 Aplikace Slepé Mapy a její analýza

4 Technologie pro funkční testování webové aplikace

V dnešní době existuje pro testování webových aplikací spousta aplikací. Tyto technologie se liší jak v podpoře různých webových prohlížečů, skriptovacích jazyků nebo cenou. Hlavním cílem této kapitoly není představit velké množství těchto nástrojů, ale pouze ty, které byly použity či s použitým řešením nějakým způsobem souvisí. Hlavní technologií, která byla při vytváření testů použita, je testovací platforma Arquillian a její rozšíření. Velmi populární alternativou je testovací nástroj Selenium, jehož komponenta WebDriver byla využita i v této práci. Tvorba testů samozřejmě není to jediné, k čemu je nutné použít specializované nástroje, proto zde budou představeny i další nástroje nutné pro automatizaci testování.

4.1 Selenium

Selenium je jeden z nejpokulárnějších nástrojů sloužících k automatickému testování webových aplikací. Jedná se o open source framework, jehož součástí je několik komponent, momentálně hlavně Selenium IDE a Selenium WebDriver, které budou dále představeny. Původní část projektu Selenium Remote Control je již zastaralá a jejím nástupcem je Selenium WebDriver, který také obsahuje schopnosti původně samostatné části Selenium Grid. Tento celek nabízí velice flexibilní platformu umožňující tvorbu testů pro širokou škálu webových prohlížečů a operačních systémů.

Ve své podstatě Selenium umožňuje ovládat instance webového prohlížeče a emulovat jeho interakci s uživatelem. Touto interakcí mohou být různé běžně prováděné úkony, jako například pohyb myši, vyplnění textového pole nebo kliknutí na tlačítka či odkazy. Jedním z principů projektu Selenium je také jednotné rozhraní pro všechny běžné prohlížeče, což umožňuje jeden test spouštět na různých webových prohlížečích bez úprav [8]. Podstatnou výhodou Selenium frameworku je také jednoduché využití jako součásti v jiných aplikacích, příkladem budiž testovací platforma Arquillian.

4.1.1 Selenium WebDriver

Po dlouhou dobu byl hlavním modulem Selenium projektu Selenium Remote Control (nazývaný také Selenium RC či Selenium 1.0), ten měl ale značné nedostatky. Tím hlavním je samotný princip fungování Selenium RC, a to vkládání funkcí psaných pomocí JavaScriptu do webového prohlížeče po jeho načtení. Tento princip činil práci s dynamickými webovými stránkami náročnou. Další nevýhodou bylo rozdílné chování JavaScriptu v různých webových prohlížečích, což mohlo způsobit neočekávané chování testů.

Přirozenou evolucí projektu Selenium RC se stal Selenium WebDriver (nazývaný také Selenium 2.0). Ten vznikl sloučením projektu WebDriver a původního Selenium 1.0 a oproti starší verzi nabízí mnohá vylepšení. Tím hlavním je využití tzv. WebDriver API¹, které využívá nativní podpory komunikovat přímo s webovým prohlížečem. Každý podporovaný webový prohlížeč má svůj vlastní WebDriver, který zajišťuje specifika práce s tímto prohlížečem. Uživatel pak už jen využívá WebDriver rozhraní, aniž by musel řešit rozdílné chování webových prohlížečů. Podporovaný jsou následující prohlížeče: Internet Explorer, Mozilla Firefox, Google Chrome, Safari, Opera, HtmlUnit a PhantomJS. Selenium WebDriver také podporuje několik programovacích jazyků, mezi ty hlavní patří Java, C#, Ruby, Python a JavaScript.

4.1.2 Selenium IDE

Placeholder.

4.2 Arquillian

Placeholder.

4.2.1 Arquillian Graphene

Placeholder.

1. Application Programming Interface

4.2.2 Arquillian Drone

Placeholder.

4.3 Ostatní nástroje

4.3.1 JUnit

4.3.2 Maven

4.3.3 Travis CI

5 Tvorba testů

6 Systém automatizovaného testování

7 Závěr

Literatura

- [1] 1. PAGE, Alan – JOHNSTON, Ken – ROLLISON, Bj. *Jak testuje software Microsoft*. Vyd. 1. Brno: Computer Press, 2009, 384 s. ISBN 978-80-251-2869-5.
- [2] 2. PATTON, Ron. *Testování softwaru*. Vyd. 1. Praha: Computer Press, 2002, xiv, 313 s. Programování. ISBN 80-7226-636-5.
- [3] 3. *What is Functional testing (Testing of functions) in software?* ISTQB Exam Certification [online]. 2015 [cit. 2015-04-12]. Dostupné z: <http://istqbexamcertification.com/what-is-functional-testing-testing-of-functions-in-software/>
- [4] 4. *What is Software Testing?* ISTQB Exam Certification [online]. 2015 [cit. 2015-04-12]. Dostupné z: <http://istqbexamcertification.com/what-is-a-software-testing/>
- [5] 5. *What is Unit testing?* ISTQB Exam Certification [online]. [cit. 2015-11-06]. Dostupné z: <http://istqbexamcertification.com/what-is-unit-testing/>
- [6] 6. *What is Regression testing in software?* ISTQB Exam Certification [online]. [cit. 2015-11-06]. Dostupné z: <http://istqbexamcertification.com/what-is-regression-testing-in-software/>
- [7] 7. *Selenium documentation*. [online]. 12.11.2015 [cit. 2015-11-12]. Dostupné z: <http://docs.seleniumhq.org/docs/index.jsp>
- [8] 8. *Selenium documentation* [online]. 12.11.2015 [cit. 2015-11-12]. Dostupné z: <https://seleniumhq.github.io/docs/index.html>