MASARYKOVA UNIVERZITA
INSTITUT BIOSTATISTIKY A ANALÝZ

# Metodiky pro automatické testování webové aplikace

Ondřej Melkes, Martin Komenda

# Obsah

# Nesprávné testování sw

- Neznalost testovacího procesu a jednotlivých typů testů = testy vlastně nejsou důležité
  - Znát kompletní testovací agendu
- Nedostatek času pro úplné testování
  - Používat dostupné nástroje pro automatizaci testů

# Proč testovat sw

- Nedílná součást životního cyklu sw
- Cílem je odhalení možných chyb
  - pozdější nalezení chyby znamená vyšší náklady na její opravení
- Testovat vůči
  - specifikaci sw
  - tomu, kdo a jak bude sw používat
    - ve specifikaci mohou být chyby

# Testování je složitý proces

- Specifikace nemusí být jednoznačná a není snadné odhalit, na co se v ní zapomnělo.
- Opakováním stejných testů se tyto testy mohou po čase stát neefektivními, proto je potřeba je aktualizovat a přizpůsobovat měnícím se okolnostem tak, aby pokrývaly co nejvíce nově vznikajících chyb.
- Výsledky testování mohou být ovlivněny špatnou komunikací v týmu nebo se zákazníkem.
- Není možné otestovat stejným způsobem různý software.
- Různé chyby se navenek mohou projevovat stejně, nebo naopak jedna chyba se navenek může projevovat různě.

# Způsoby testování

- Black box testing = funkční testování
  - Nejsou k dispozici zdrojové kódy ani dokumentace
  - Testy podle scénářů, které definují, jak přesně se má při testu postupovat
  - Jasně definované vstupy a výstupy

✓ Rychlost
✓ Jednoduchost
✓ Bez znalosti programovacích jazyků

- Riziko nekvalitního kódu
- Plošné nepokrytí všech chyb

# Způsoby testování

- White box testing = strukturální testování
  - Znalost datových a programových struktur
  - Použití debuggerů pro analyzování kódu programu za jeho běhu

✓ Odhalení nežádoucího kódu

✓ Vyšší kvalita kódu

– Náročnější a nákladnější

# Fáze testování

- Kontrola kódu během napsání
- Jednotkové testování (Unit testing)
  - Podrobné testování malých částí kódu
  - Funkce, procedura, objekt a jeho metody
- Integrační testování (Integration testing)
  - Test nově přidané funkcionality s celkem
- Systémové testování (System testing)
  - Ověření správné funkcionality systému jako celku (multi-iterační proces)
  - Testování výkonnosti (zátěžový test, test objemu dat)
  - Testování použitelnosti (jak se systémem pracují jeho uživatelé- manuálně)
  - Testování přístupnosti (různá zobrazovacích zařízení, hendicapovaní uživatelé)
  - Testování bezpečnosti
- Akceptační testování (Acceptance testing)

# Výhody automatizované testování

- Rychlost – automatické testovací nástroje mohou testy provádět mnohonásobně rychleji, než by je dělal člověk.
- Efektivita – během automaticky vykonávaného testu se může věnovat plánování dalších testových případů a manuálnímu testování případů, které automatizovat nejdou.
- Přesnost – na rozdíl od člověka provádí testovací nástroj práci vždy se stejnou přesností.
- Neúnavnost – automatické testy mohou běžet neustále.

# Unit testy

- White box testování, nízkoúrovňové
- Kontrola správné funkčnosti jedné třídy a jejich metod
- Idea: Ujistit se, že jednotlivé části "skládanky" fungují, jak mají, a tak snížit počet chyb v programu

# Unit testy

- Nutnost správného návrhu aplikace a dodržování standardů
- Nutnost testy udržovat
- Ne pro každý kód lze navrhnout unit test
- Rychlé testy, snadná identifikace kde a kdy vznikla chyba

# Unit testy

- Výhody
  - Ověřují dlouhodobou funkčnost třídy
  - Podporuje změny
  - Zjednodušuje integraci
  - Testy slouží jako dokumentace
  - Pomáhá oddělit rozhraní od implementace

- Nevýhody
  - Vyžaduje disciplínu
  - Zpomaluje psaní a úpravy kódu

# Unit testy

- SRP – single responsibility principe
  - Třída zodpovídá za jednu odpovědnost
  - Porušení tohoto principu zvýší složitost kódu a znesnadní/znemožní testování
  - Důsledek: netestují se private metody, pouze vstupy a výstupy public metod

# Unit testy

- OCP – open closed principe
  - Třída je otevřená pro rozšiřování, uzavřená pro změny
  - Přidávání funkčnosti bez zásahu do hotového kódu
  - Důsledek: používaní rozhraní a dedičnosti

# Unit testy

- Nepoužívat statické a globální metody
  - Globální a statické metody zavádějí do Unit testu závislosti, nelze tak testovat pouze jednu třídu
  - V testech se lokální závislosti nahradí mocky, např. filesystém,
  - Důsledek: třída a metoda mají jasně dané veřejné závislosti a přehledné API

# Unit testy

- KISS – keep it simple stupid
  - Přidávat funkcionalitu nebo vyhodit vyjímku?
- DRY – dont repeaty yourself
  - Používat otestovaný kód
- Používat hlavu a ne dogmatický přístup
  - Testování databáze

# Unit testy - javascript, ajax

- Plánováno
  - výběr testovací frameworku
  - Konfigurace dummy server https://github.com/bak1an/silly-server
    - Řeší jak testovat ajaxové požadavky (chyba může být pouze na straně JS, ne na straně serverové aplikace)

# Integrační testy

- Black box testování
- Testování komunikace komponent/tříd
- Idea: Ujistit se, že jednotlivé části "skládanky" komunikují, jak mají a na zadaný vstup dávají správný výstup

# Integrační testy

- Nutnost definovat vstupy a výstupy jednotlivých operací
- Nutnost testy udržovat
- Pomalé testy, složitější určení místa vzniku chyby, ověřuje zda je aplikace po zaslání požadavku v očekávaném stavu

# Integrační testy

- Výhody
  - Ověřují základní funkčnost aplikace
  - Automatizované se pravidelně se opakují, ověření zda nevznikla chyba z jiných zdrojů
- Nevýhody
  - Netestují živou aplikaci, testování typicky začíná na definovaném stavu aplikace, který se při každém testu obnoví

# Integrační testy

- Plánováno
  - Výběr jednoduchého nástroje pro správu automatizovaných testů (GitLab CI, TestLink, Jenkins)
- Využívat
  - PHP_Unit: testy databáze, sestavení controleru
  - Selenium: kompletní http požadavek a odpověď

# Nástroje pro automatické testování UI

- Funkční testování simulující aktivitu uživatele
- Nutno vždy rozhodnout, zda danou funkcionalitu nepřenechat manuálnímu testu
- Desktopové vs. Webově orientované aplikace
  - běží většinou na webovém serveru a k jejich rozhraní se přistupuje pomocí webového prohlížeče

# Přehled software pro automatizované testování

- Selenium
- Tellurium IDE
- WebAii Framework
- WatiN

# Přehled sofware: Tellurium IDE

- Doplněk do Firefox
- Nahrávání „naklikaných“ testovacích scénářů, jejich editace a spouštění s následným vyhodnocením
- Odlišný způsob identifikace jednotlivých elementů oproti Seleniu
  - Seskupuje UI elementy jako UI objekty

# Přehled sofware: WebAii Framework

- K použití zdarma, ale není open-source
- Testy lze psát v C# nebo Visual Basic .NET
- Probíhá na prohlížečích
  - Internet Explorer, Firefox, Google Chrome a Safari
- Pracuje v kombinaci s některým z frameworků na jednotkové testy
- Podobný přístup ke tvorbě a vykonávání testů jako u Selenia WebDrive
- Dále umožňuje
  - Vyvolávání a odchytávání Javascriptových událostí
  - Správa cookies prohlížeče

# Přehled sofware: WatiN

- Open-source framework
- Psaní testovacích skriptů v jazyce C#
- Pracuje v kombinaci s některým z frameworků na jednotkové testy
- Podporované webové prohlížeče
  - Internet Explorer a Firefox (chyby ve verzi 12)
- Mnoho problematických oblastí
  - minimální možnosti konfigurace testů

# Přehled software: Selenium

- Open-source sada nástrojů
- Psaní testů v mnoha jazycích
  - C#, Java, Groovy, Perl, PHP, Python a Ruby
- Spouštění testů ve webových prohlížečích
  - Doplněk do Firefox
- Nejrozšířenější řešení pro testování webových rozhraní
- Selenium IDE - vývojové prostředí pro Firefox
- Selenium Remote Control - Selenium Server a klientská knihovna
- Selenium Webdriver – nástupce RC
- Selenium Grid

# Selenium IDE: ukázka v praxi

SeleniumHQ
Browser Automation

edit this page search selenium: Go

Projects Download Documentation Support About

## What is Selenium?

*Selenium automates browsers*. That's it. What you do with that power is entirely up to you. Primarily it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should!) also be automated as well.

Selenium has the support of some of the largest browser vendors who have taken (or are taking) steps to make Selenium a native part of their browser. It is also the core technology in countless other browser automation tools, APIs and frameworks.

# Záznamy o testování

- On-line prostřednictvím nástroje GITlab
- Off-line prostřednictvím dokumentu
  - R33 Záznam o testování

# R33 – Záznam o testováni

- Úprava R33 dokumentu
  - Návrh na změnu struktury

| | | |
|---|---|---|
| **Oblast testování** | **Vyberte oblast testování** | |
| **Typ záznamu o testování:** | ☐ **Chyba** | ☐ **Požadavek** |
| **Název chyby/požadavku** | Klikněte a zadejte název požadavku | |
| **Podrobný popis**<br><br>**Chyba:**<br>• Postup pro vyvolání chyby<br>• Jaký je očekávaný výsledek<br>• Jaký je skutečný výsledek<br>• Identifikace výskytu chyby<br><br>**Požadavek:**<br>• Specifikace požadované změny nebo nové funkcionality | Klikněte se a zadejte podrobný popis | |
| **Stanovisko vývojáře** | ☐ **Vyřešeno** | ☐ **Odloženo** |

# Zdroje


- http://www.zdrojak.cz/clanky/testovani-a-tvorba-testovatelneho-kodu-v-php/
- http://www.zdrojak.cz/clanky/navrhove-principy-solid/
- http://www.ibm.com/developerworks/web/library/wa-aj-testing/
- http://testovanisoftwaru.cz/
- http://www.littlehart.net/atthekeyboard/2012/08/16/5-minute-tdd/
- http://www.exubero.com/junit/antipatterns.html/
- http://www.ibm.com/developerworks/opensource/library/os-junit/?ca=dgr-lnxw07JUnite
- http://www.swtestovani.cz/
- http://docs.seleniumhq.org/
- http://is.muni.cz/th/172724/fi_m/_DP_Pietrik.pdf