Test-Webseite

Zurück

Willkommen

Dies ist eine Webseite für einen automatischen Test. Geben Sie unten ihren Vornamen und Nachnamen ein.
Bedingungen:

- Der Vorname und Nachname darf nur alphabetisch sein.
- Der Vorname und Nachname darf nicht länger als 10 Zeichen sein.

JeanPierre

Name

Tshimanga

Nachname

senden

Test-Webseite

Das Ergebnis!

Hallo, JeanPierre Tshimanga!

Zurück zur Webseite

Vitruvius Hochschule Leipzig - Hochschule für angewandte Wissenschaften

Modul Web-Technologien 1

Projektarbeit Testing

Wintersemester 2017 / 2018

Studiengang Software Engineering und Information Security

Thema:	Webseiten Testing
Eingereicht von:	Jean-Pierre Tshimanga
Eingereicht am:	
Betreuer:	Herr Dr. Hartwig

Inhaltsverzeichnis

Einleitung

Website-Testing oder Web-Applikation's testen, ist das testen einer Webseite oder Web-Applikation auf potentielle Programmierfehler oder Bugs vor Veröffentlichung. Webseiten können z.B. auf Sicherheitslücken, Barrierefreiheit und Anpassung an verschiedene Geräte, Browser und Betriebssyteme überprüft werden.

In der vorliegenden Arbeit werden kurz gängige Einstiegsbegriffe erläutert, wichtige Aufgaben im Testing benannt und zwei einführende Praxisbeispiel mit dem Selenium Tool vorgestellt.

Diese Webseite soll im ganzen einen ersten Einblick in die Thematik verschaffen.

Warum Testing ?

Das Web-Applikationstesten wird eingesetzt, um die Qualität einer Webseite zu messen. Es wird dabei überprüft , ob die Webseite die für ihren Einsatz bestimmten Anforderungen erfüllt. Die Tests können z.B. während der Softwareentwicklung durchgeführt werden.
Einen Nachweis das keine Fehler mehr vorhanden sind kann das Softwaretesten aber leider nicht erbringen.

> “ Program testing can be used to show the presence of bugs,
> but never show their absence! ”
>
> Edsger W. Dijkstra

Für das Softwaretesten gibt es verschiedene Definitionen. Diese sind Teil der allgemeinen Softwareentwicklung und nicht speziell

für Web-Applikationen definiert. Siehe: ANSI/IEEE Std. 610.12-1990

Einstiegs Begriffe

Begriff	Erklärung
Bug:	Der Begriff "Bug" steht als Synonym für Programmfehler. Es bezeichnet das Fehlverhalten eines Programmes.
Bug Report:	Ein "Bug Report" (Fehlerbericht) ist die Beschreibung eines auftretenden fehlers.
Software Feature:	Eine spezielle Funktionalität einer Software.
User Story:	Anforderungen des Benutzers an die Software.
Testfall/Testcase:	Beschreibt einen elementaren funktionalen Softwaretest zur Überprüfung einer Spezifikation.
Nachtest:	Zum belegen, dass der Fehler behoben wurde.
Regressionstest:	Die Wiederholung von Testfällen um bei Änderungen neue Bugs zu vermeiden.

Aufgaben im Testing

Es gibt verschiedenen Softwaretests. Die Tests variieren je nach Produkt, Anforderungen und Service.
Außerdem unterscheiden sich Tests für Web-Anwendungen von denen für Desktop Anwendungen.

Software Testing Arten	Haupt-Fokus	Wer sind die Software Tester?
Installation Testing	Sicherstellung der korrekten Installation	Entwickler
Compatibility Testing	Sicherstellung der Kompatibilität des Systems	Entwickler
Smoke Testing	Basistest der Software Funktion	Entwickler
Regression Testing	Fehler Dokumentation nach Code Aktualisierungen	Entwickler
Acceptance Testing	Test in dem User Anforderungen überprüft werden	Qualitätssicherung / Kunden
Alpha Testing	Interne Form des Acceptance Testing	Qualitätssicherung
Beta Testing	Externe Form des Acceptance Testing	Externe Qualitätssicherung / Kunden
Continuous Testing	Automatisierte Software Tests	(primär) Entwickler
Destructive Testing	Gezielte Produktion von Software Abstürzen	Qualitätssicherung
Usability Testing	Test des User Interfaces	Qualitätssicherung / Kunden
Accessibility Testing	Barrierefreiheit-Tests (W3C Standard)	Interne Qualitätssicherung
Security Testing	Evaluierung inwieweit Daten unautorisiert gelesen verwendet und modifiziert werden können	Interne & externe IT Spezialisten
A/B Testing	2 Varianten Tests	Marketing / Qualitätssicherung

https://usersnap.com/de/blog/software-testing/

- Mit den verschiedenen Softwaretests unterscheiden sich die Aufgaben.
  Allgemein können Aufgaben für das Softwaretesting z.B. das testen von neu implementierten Features oder User Stories sein.
- Oder das testen von neu erstellten Hompages oder Web-Anwendungen. Der Tester sucht dabei nach Bugs in der Anwendung.
  Ist ein Bug gefunden wird dieser gemeldet.
  Die Entwickler beheben den Fehler und es wird ein Nachtest gemacht um zu überprüfen, ob der Fehler noch besteht.
- Der Regressionstest wird vom Tester durchgeführt um bestehende Softwarefeatures auf Fehler zu überprüfen. Es ist bei

Programmänderungen wichtig sicherzustellen, dass alle bisherigen Programmfeatures weiterhin funktionieren.

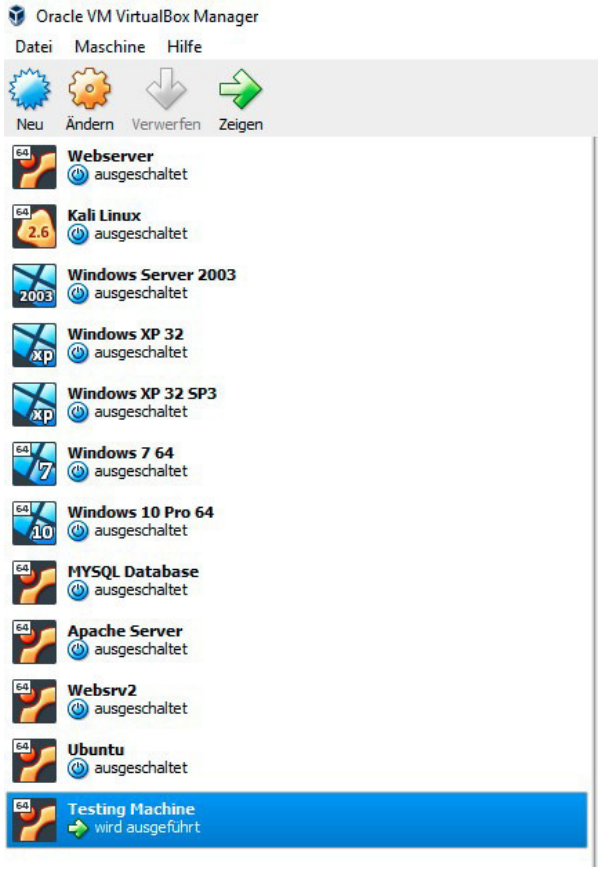
- Explorative Tests werden von Testern durchgeführt ohne ein konkreten Fehler im Blick zu haben. Der Tester testet erfahrungsgemäß um z.B. fehlerhaftes Verhalten der Anwendung oder offensichtlich unlogische Abläufe zu finden.
- Der Tester erstellt auch die Testfälle die geprüft werden sollen.

Testing-Praxis

In der Praxis können einige Hilfsmittel und Tools verwendet werden um das Testen zu erleichtern.
Es gibt das manuelle und automatisierte Testing. Zweiteres werde ich weiter unten mit einem Praxisbeispiel kurz vorführen.
Manuelle Tests werden vom Tester von Hand und ohne Unterstützung von tools oder Skripts ausgeführt. Manuelles Testing ist dann von Vorteil, wenn Testfälle nur ein oder zweimal getestet werden müssen. Automatisierte Tests basieren auf manuellen Tests die Tests müssen also zuerst von Hand geschrieben werden.

Virtuelle-Maschinen

Um die Web-Applikation oder Webseite zeitgleich auf verschiedene Betriebssysteme zu testen, können virtuelle Maschinen verwendet werden.
Auf den Maschinen können mehrere Systeme gleichzeitig laufen. So kann man z.B. die Webseite auf einem Linux Betriebssystem oder einem Windows Betriebssystem prüfen. Die virtualisierungssoftware Oracle VM VirtualBox ermöglicht das erstellen von Maschinen. www.virtualbox.org



VirtualBox Software

Website-Testing

Um ein praktisches Beispiel vom Testen zu haben werde ich hier eine kleine Web-Applikation vorstellen.
Die Web-Applikation wird auf einem LAMP Server installiert. LAMP bedeutet Linux, Apache, MySQL und PHP. Der LAMP Server läft auf einer virtuelle Maschine in der Software VirtualBox.

Die Installierten Komponente:

Betriebssystem	FTP	Webserver	Datenbank	PHP

Ubuntu Server 16.04	vsftpd	Apache2	mysql	php7.1

Die Webseite wird mit Html5 und dem W3.css (W3Schools) framework benutzerfreundlich gestaltet Siehe hier. Es befindet sich im article Container der Webseite eine Form. Diese Form wird verwendet um einen Vornamen und einen Nachnamen aufzunehmen. Die Werte werden dann an die Seite test_index.php gesendet und überprüft. Die PHP Anwendung prüft die Eingaben auf:

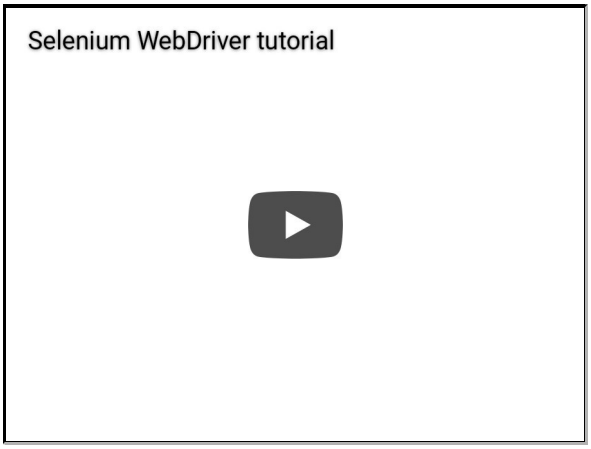
1. Anzahl der Zeichen dürfen maximal 10 Zeichen lang sein.
2. Eingabe darf nur alphabetisch sein.

Selenium

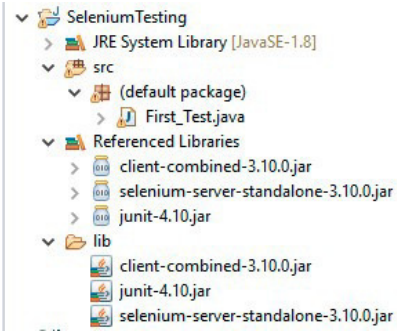
Um die Webseite mit den Selenium Tools zu testen benötigt man:

- Selenium Standalone Server, Selenium Client & WebDriver Language Bindings,
- Java,
- Eclipse IDE
- GeckoDriver
- Firefox
- JUnit

Die Installation der einzelnen Komponente sollte man sich entsprechend im Manual der Hersteller ansehen. Allgemein werden die Java Komponente und Bibliotheken von den Selenium Tools im entsprechenden Eclipse Projekt eingebunden. Der GeckoDriver erlaubt dann den Zugriff über Java Code auf den Firefox Browser. In Java können JUnit Tests geschrieben werden um Testcases zu prüfen.



Selenium automatisiert den Web-Browser und ist kompatibel mit dem Firefox Browser. Es unterstützt die Programmiersprachen Java,C#,Ruby,Python und Javascript. Nach hinzufügen der Komponente sieht die Projekt Struktur im Eclipse IDE so aus:



Die Testfälle

Das testen der Webseite mit den Selenium Tools erfolgt im Eclipse IDE. Es wird ein JUnit Projekt erstellt und anschließend die Tests im Java Code geschrieben. Die Test fälle werden mit @Test beschriftet.

```
@Test   //Erster Test überprüft den Titel
public void title_test() {

    System.out.println("Test 1");
    System.setProperty("webdriver.gecko.driver","C:\\Selenium\\geckodriver.exe");
    browser = new FirefoxDriver();
    browser.get("http://192.168.178.39/testing/test_index.html");

    String title = browser.getTitle();
    assertTrue(title.contains("Testing Website"));

    browser.close();

}

@Test   //zweiter Test überprüft das Ergebnis der Namenseingabe
public void formfield_test() {
    System.out.println("Test 2");
    //Firefox geckodriver pfad
    System.setProperty("webdriver.gecko.driver","C:\\Selenium\\geckodriver.exe");
    browser = new FirefoxDriver();

    browser.get("http://192.168.178.39/testing/test_index.html");

    WebElement article_vorname = browser.findElement(By.name("vorname"));
    WebElement article_nachname = browser.findElement(By.name("nachname"));

    article_vorname.sendKeys("Jean-Pierre");
    article_nachname.sendKeys("Tshimanga");
    article_vorname.submit();
    browser.manage().timeouts().implicitlyWait(1, TimeUnit.SECONDS);
    boolean text = browser.getPageSource().contains("alphabetisch");
    assertTrue(!text);
    browser.close();
}
```

Es wird getestet auf:

- Title im Header
- Eingabe des Vornamens und Nachnamen und ausführen des Submit Button.

Der Titel der Webseite soll lauten "Testing Website". Wird die Bedingung erfüllt gibt die Assert Anweisung ein true zurück und der erste Test war erfolgreich.

Der zweite Test füllt selbstständig das Formular aus und betätigt den Submit Button. Auf der Nachfolgenden Webseite wird dann der Text "alphabetisch" im Quelltext der Seite gesucht.

Findet die Anweisung den Text bedeutet dies das der eingegebene Name ungültig ist. Wir testen also, ob die Fehlerhafte Eingabe auch als falsch erkannt wird. Der Vorname Jean-Pierre enthällt das Zeichen "-" und ist somit nicht alphabetisch. Die Assert Funktion gibt also auch hier ein True zurück.

Fehlerberichte

Zu den Testfällen werden Fehlerberichte generiert. Die Fehlerberichte geben auskunft, ob alle Testfälle erfolgreich verlaufen sind. Man kann das Java Projekt mit JUnit ausführen und bekommt dann im JUnit Fenster einen kleinen Test Status angezeigt.

Die beiden Tests waren in dem Fall erfolgreich.

Testing mit Gradle

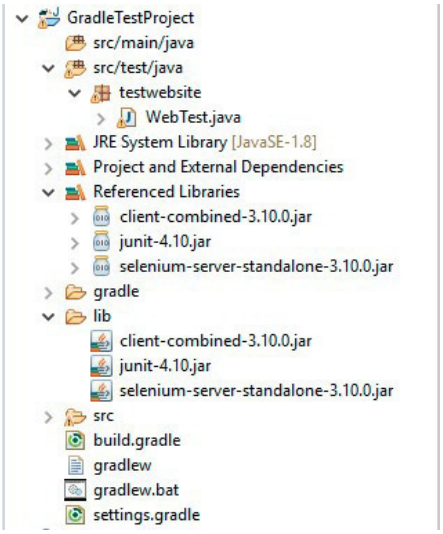
Gradle ist ein Build-Management Tool und basiert auf Java.

Um Selenium und Gradle unter Eclipse zu verwenden benötigt man einige weitere Komponente. Ich werde hier Gradle verwenden um einen HTML Report für die Testfälle auszugeben.

Die Installation des Gradle-4.10 ist gut Dokumentiert sodass keine Schwierigkeiten entstehen. Gradle Installation Guide

Nach der Installation muss darauf geachtet werden das die JAVA_HOME Variable in der Umgebungsvariable richtig gesetzt wird. Für die aktuelle Java Version lautet der Pfad unter Windows "C:\Program Files\Java\jdk-9.0.4". Wenn Gradle Installiert ist kann man dies in der Konsole mit dem Befehl "gradle" überprüfen.

In der Eclipse Umgebung kann man sich das Gradle Plugin Buildship dazu Installieren Github Eclipse Buildship Plugin Erstellt man ein neues Gradle Projekt sollte der Projekt Ordner so aussehen:



Wieder werden die Selenium Bibliotheken wie zuvor hinzugefügt. Man Öffnet die Datei build.gradle und kann dort die nötigen dependencies für Selenium eintragen. Auf der Webseite https://mvnrepository.com wird man für selenium-java fündig.

```
// Apply the java-library plugin to add support for Java Library
apply plugin: 'java-library'
apply plugin: 'java'
// In this section you declare where to find the dependencies of your project
repositories {
    // Use jcenter for resolving your dependencies.
    // You can declare any Maven/Ivy/file repository here.
    jcenter()
}

dependencies {
    // This dependency is exported to consumers, that is to say found on their compile classpath.
    api 'org.apache.commons:commons-math3:3.6.1'

    // This dependency is used internally, and not exposed to consumers on their own compile classpath.
    implementation 'com.google.guava:guava:23.0'

    // Use JUnit test framework
    testImplementation 'junit:junit:4.12'
    testCompile group: 'junit', name: 'junit', version: '4.12'
    // https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java
    compile group: 'org.seleniumhq.selenium', name: 'selenium-java', version: '3.10.0'
    //compile group: 'org.hamcrest', name: 'java-hamcrest', version: '2.0.0.0'

}

test {
    reports {
        junitXml.enabled = false
        html.enabled = true
    }
}
```

Das Build Skript enthält auch Anweisungen im Test Block für das Ausgabeformat des Test Reportes. Die Datei WebTest.java im Projektverzeichnis GradleTestProject liegt unter src/test/java. WebTest.java unterscheidet sich nicht von dem Test unter 3.3. Mit der Kommandozeile navigiert man in das Projektverzeichnis und führt das Kommando "gradle test" aus. Gradle sollte zuerst alle Abhängigkeiten installieren und dann erfolgreich die Tests durchlaufen.

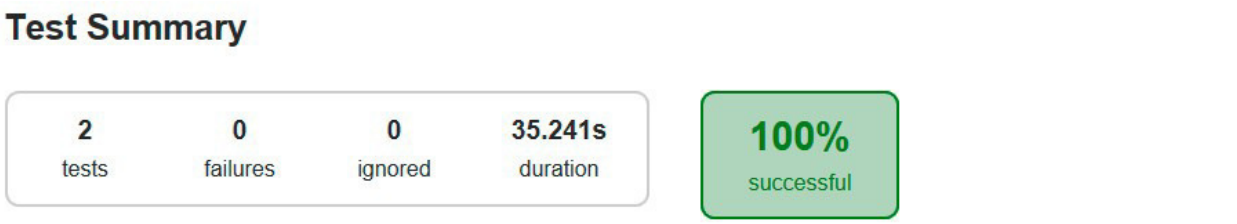
```
Download https://jcenter.bintray.com/com/squareup/okio/okio-parent/1.13.0/okio-parent-1.13.0.pom
Download https://jcenter.bintray.com/com/squareup/okhttp3/parent/3.9.1/parent-3.9.1.pom
Download https://jcenter.bintray.com/org/seleniumhq/selenium/selenium-java/3.10.0/selenium-java-3.10.0.jar
Download https://jcenter.bintray.com/org/seleniumhq/selenium/selenium-edge-driver/3.10.0/selenium-edge-driver-3.10.0.
Download https://jcenter.bintray.com/org/seleniumhq/selenium/selenium-opera-driver/3.10.0/selenium-opera-driver-3.10.
ar
Download https://jcenter.bintray.com/org/seleniumhq/selenium/selenium-ie-driver/3.10.0/selenium-ie-driver-3.10.0.jar
Download https://jcenter.bintray.com/org/seleniumhq/selenium/selenium-chrome-driver/3.10.0/selenium-chrome-driver-3.1
.jar
Download https://jcenter.bintray.com/org/seleniumhq/selenium/selenium-support/3.10.0/selenium-support-3.10.0.jar
Download https://jcenter.bintray.com/org/seleniumhq/selenium/selenium-firefox-driver/3.10.0/selenium-firefox-driver-3
.0.jar
Download https://jcenter.bintray.com/org/seleniumhq/selenium/selenium-api/3.10.0/selenium-api-3.10.0.jar
Download https://jcenter.bintray.com/org/seleniumhq/selenium/selenium-remote-driver/3.10.0/selenium-remote-driver-3.1
.jar
Download https://jcenter.bintray.com/net/bytebuddy/byte-buddy/1.7.9/byte-buddy-1.7.9.jar
Download https://jcenter.bintray.com/commons-codec/commons-codec/1.10/commons-codec-1.10.jar
Download https://jcenter.bintray.com/org/apache/httpcomponents/httpcore/4.4.6/httpcore-4.4.6.jar
Download https://jcenter.bintray.com/com/squareup/okhttp3/okhttp/3.9.1/okhttp-3.9.1.jar
Download https://jcenter.bintray.com/com/squareup/okio/okio/1.13.0/okio-1.13.0.jar
Download https://jcenter.bintray.com/org/seleniumhq/selenium/selenium-safari-driver/3.10.0/selenium-safari-driver-3.1
.jar

BUILD SUCCESSFUL in 40s
2 actionable tasks: 2 executed
C:\Users\xorpad\eclipse-workspace\GradleTestProject>gradle test

BUILD SUCCESSFUL in 38s
2 actionable tasks: 1 executed, 1 up-to-date
C:\Users\xorpad\eclipse-workspace\GradleTestProject>
```

Der Test Report liegt dann im Verzeichnis "\GradleTestProject\build\reports\tests\test".

Abschließend der HTML Report:



Package	Tests	Failures	Ignored	Duration	Success rate
testwebsite	2	0	0	35.241s	100%

Fazit

Testing ist ein sehr umfangreiches Gebiet. Wichtige Begriffe aus Web- und Desktop-Applikation Testing überscheiden sich aber. Die Aufgaben für den Tester können sehr unterschiedlich sein und sind teils abhängig vom Produkt. Manuelle Tests sind aufwändig, dienen aber als Vorlage für automatisierte Tests. Ein hilfreiches Framework ist Selenium. Nutzt man außerdem das richtige Build Tool wie z.B. Gradle stehen einem weitere nützliche Optionen zur Verfügung.

Quellenangaben und Literatur

https://de.wikipedia.org/wiki/Softwaretest
https://en.wikipedia.org/wiki/Software_feature
https://en.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers
https://de.wikipedia.org/wiki/User-Story
https://de.wikipedia.org/wiki/Regressionstest
https://usersnap.com/de/blog/software-testing/
https://www.testbirds.de/leistungen/quality-assurance/exploratives-bug-testing/
https://produkt-manager.net/2011/exploratory-testing-scrum-teams-testen-wie-touristen/

Bücher

Selenium Webdriver: Software Automation Testing Secrets Revealed von Narayanan Palani
Mastering Selenium WebDriver von Mark Collin
Gradle: Ein kompakter Einstieg in das Build-Management-System von Joachim Baumann