Selenium

Wie sie alle wissen sind in Agilen Projekten heutzutage automatisierte Tests unverzichtbar. Vor allem bei Continious Integration spielen sie eine wichtige Rolle, um zu gewährleisten, dass der neue Code fehlerfrei ist und bereits bestehenden Code durch Veränderungen nicht kaputt gemacht wurde.

Wie bereits der Stefan vorher erklärt hat, gibt es das V-Modell, indem verschiedene Teststufen vorhanden sind. Selenium befindet sich in der Stufe der Systemtests. In der Stufe der Systemtests wird das gesamte System getestet, sprich man arbeitet mit dem Produkt wie ein Benutzer.

Selenium ist ein Framework, welches dazu dient, UI Tests zu automatisieren. Durch den automatischen Ablauf der Tests können Kosten und Zeitaufwand gering gehalten werden.

Grundsätzlich besteht die Selenium Familie aus mehreren Tools in der Testautomatisierung.

- Selenium IDE
- Framework Selenium Webdriver

Und auf diese werde ich jetzt näher eingehen

Selenium IDE

Selenium IDE ist ein FireFox Add on. Das Erstellen von Tests ist relativ simpel und hat eigentlich nicht viel mit Programmieren zu tun. Im Prinzip muss man nur einen Aufnahmeknopf drücken und den Test einmal manuell durchführen. Somit ist das meiner Meinung nach die einfachste und schnellste Möglichkeit Selenium Tests zu erstellen.

Jedoch bringt dies auch einige Nachteile wie z.B. dass sich

- das Testen auf den FireFox browser beschränkt und man als Produktanbieter eigentlich alle Browser testen möchte.
- Bedingte Statements können nicht implementiert werden wie z.B. dass ein Test nur erfolgreich ist wenn ein successful Toast erscheint
- Schleifen Statements können auch nicht implementiert. Diese Statements werden Beispielsweise benötigt wenn wenn man alle Elemente einer Liste auswählen möchte.
- Exceptionhandling wird nicht unterstützt

Beispiel Selenium googlen selbst machen (4)

Ich habe das auch bei meiner Diplomarbeit kurz versucht Tests mit Selenium IDE zu erstellen. Kurz vorweg, **(Diplomarbeitsstartseite öffnen)** auf dieser Seite von meiner Diplomarbeit kann man 2 verschiedene Tools starten. Wenn ich jetzt z.B. das Performance Tool starte **(starten)** kommt eine Meldung, dass es gestartet ist und kurz später, dass es erfolgreich beendet wurde. Nachteil bei Selenium DIE ist dass man z.B. bei diesem Test nicht überprüfen kann, ob die Meldung dass das Tool erfolgreich beendet wurde überhaupt erscheint. **(SeleniumIDE Diplomarbeitstests ausführen)**

Selenium – Webdriver

Der Selenium Webdriver ist ein Framework das das Automatisieren von Webanwendungen im Gegensatz zur Selenium IDE in mehreren Browsern unterstützt. Das Erstellen von Tests mit diesem Framwork ist klarerweise viel aufwändiger, da man nicht mehr einfach Tests aufnehmen kann sondern wirklich Tests mit Code schreibt.

Unterstützte sprachen sind unter anderem Java, C# und PHP

Für die Verwendung in Java mit Maven muss man lediglich eine dependency eintragen.

Um jetzt näher auf Selenium Webdriver eingehen zu können möchte ich dies anhand eines Beispiels erklären. Das Beispiel soll lediglich Selenium Googeln und das 1. Suchergebnis auswählen. Mir ist bewusst das Beispiel ist kein klassischer Testfall ist aber es soll letztendlich nur für Erklärungszwecke dienen, dass man besser versteht wie Selenium funktioniert. **(Google test ausführen) (3)**

Um mit Selenium arbeiten zu können muss man eigentlich nur 3 wichtige Bestandteile können.

Wait

Wait ist ein Bestandteil von Selenium, der benutzt wird, um auf Elemente zu warten, bis sie im DOM geladen sind. Hintergrund ist der, dass man immer auf Elemente warten muss, bis sie im DOM enthalten sind, bevor man mit ihnen interagieren kann. Als Benutzer eines Browsers hat man dieses Problem klarerweise nicht, weil man langsamer ist als der Browser. Das Java-Selenium Programm ist jedoch immer schneller wie der Browser und deshalb muss man auf die Elemente warten. Grundsätzlich kann ich empfehlen immer vor einer Interaktion mit einem Element mit wait zu überprüfen, ob dieses Element auch schon im DOM geladen ist. **(wait herzeigen im google testprojekt)**

Selektoren

Durch Selektoren weiß Selenium welche GUI Elemente angesprochen werden sollen, sprich welchen Button man klicken sollte, oder in welches Textfeld man einen Text eintragen möchte. Selektoren sind essentiell, um überhaupt mit Selenium arbeiten. Es gibt grundsätzlich viele Selektoren, mit denen man arbeiten kann, der wichtigste ist meiner Meinung nach XPath Selektoren, da man mit diesem alles abdecken kann.

https://www.guru99.com/locators-in-selenium-ide.html

XPath

```
String selectorGoogleInputField = "//input[@class='gLFyf gsfi']";
```

Grundsätzlich muss man wissen, dass die XPath Selektoren immer mit 2 // beginnen, dann kommt das Element, mit dem man arbeiten möchte und dann und dann kommen mit einem @ davor die Attribute, die man verwenden möchte um das Element eindeutig zu identifizieren. Wenn man einen XPath Selektor erstellen muss geht man auf die zu automatisierende Seite und untersucht die Seite bzw. das Element, mit dem man interagieren möchte. Wie habe ich das jetzt bei der Google Suchautomation gemacht? (zuerst im Programm herzeigen dann bei Google Suche herzeigen)

Wenn man dann Elemente mit Hilfe der Selektoren identifiziert hat, kann man sie z.B. mit der Funktion Click anklicken oder mit der Funktion SendKeys einen Text eintragen auf einem Textfeld. **(selektor herzeigen im google testprojekt)**

Designpattern POM – Page Objetc Model

Grundsätzlich sagt das Page Object Model Design Pattern aus, dass jede Seite im Browser einer eigenen Klasse im Selenium Programm entspricht. Die jeweiligen Funktionen, die man auf der Seite durchführen kann, werden in Methoden in der Klasse ausprogrammiert.

(Google Beispiel öffnen) Um das anhand meines Google Beispiels zu erklären, habe ich mir ein eigenen Package pages erstell, wo alle Seiten, die im Browser geöffnet werden, durch eine Klasse abgebildet sind **(Pages vergleichen mit google Chrome pages)**. Ich habe das noch erweitert mit einer BasePage, von der alle Pages erben. Hier habe ich Selektoren gespeichert, die auf mehren Pages verwendet werden können.

https://www.softwaretestingmaterial.com/page-object-model/

Um jetzt noch sinnvolle Tests herzeigen zu können, habe ich Tests für meiner Diplomarbeit implementiert. Der Test sollte so ausschauen, dass das PerformanceTool gestartet wird und anschließend überprüft wird, ob die Meldung kommt, dass es erfolgreich beendet wurde. **(Test starten)**. So sollte dann eben ein richtiger Selenium Test aussehen **(herzeigen, wo auf die erfolgreich Meldung gewartet wird)**.

Protractor

Jetzt möchte ich noch kurz auf Protractor eingehen. Kurz vorweg Protractor ist auch ein Framework, um Browsertests zu automatisieren. Ich habe von Prof. Stütz die Aufgabe bekommen, herauszufinden, welche Vor- bzw. Nachteile es gegenüber Selenium hat und wann man es genau einsetzt.

Um jetzt die Frage gleich zu beantworten Protractor ist ein Framework das sich auf Angular spezialisiert hat. Bei Angular Webanwendungen bringt es einige Vorteile mit sich, wie dass man sich z.B. das wait erspart bevor man mit einem Element interagieren kann, da dies von Protractor geregelt wird. Das erspart natürlich viel Zeit und Code da man sich das Zusammenstelen von den Selektoren spart. Grundsätzlich kann man Protractor auch z.B. mit vuejs oder react verwenden jedoch funktioniert dann das automatische Warten nicht mehr und man muss es deaktivieren und erst wieder selbst managen.

Da Protractor vor allem in anderen Browsern als auch in Chrome oft fehleranfällig sein kann, wird empfohlen mit Selenium zu arbeiten. Ein weiterer Kritikpunkt ist, dass Selenium im Vergleich zu Protractor schon sehr lange am Markt ist und somit viel mehr Support im Internet geboten wird.

https://www.softwaretestingmaterial.com/page-object-model/

Meiner Meinung nach ist es zwar ein cooles Feature dass man bei Protractor sich das Wait erspart, jedoch habe ich selber bei der Testentwicklung gemerkt, dass das Framework oft unzuverlässig ist und man z.B. bei einem Click auf einem Button zwar nicht warten muss bis es im DOM ist aber man warten muss bis es clickable ist somit erspart man sich genau keinen Code.

(öffnen VS Code Diplomarbeit) Ich habe die Tests meiner Diplomarbeit, die ich mit Selenium geschrieben habe ebenfalls in Protractor implementiert.

Grundsätzlich wird beim Anlegen eines Angular Projekts immer der e2e Ordner auch angelegt, indem man die Protractor Tests implementiert. Um einen Protractor Test schreiben zu können werden grundsätzlich 2 Files benötigt. Ein File, wo die Testfälle erstellt werden und ein Konfigurationsfile, wo das File eingetragen wird, wo sich die tatsächlichen Tests befinden, und noch andere Informationen wie z.B. welcher Browser verwendet werden soll, die defaultadresse, die geöffnet werden soll und das Testframework Jasmine. **(Tests starten mit ng e2e)**