Kostenloses eBook

LERNEN selenium-webdriver

Free unaffiliated eBook created from
Stack Overflow contributors.

#selenium-webdriver

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: selenium-webdriver

It is an unofficial and free selenium-webdriver ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official selenium-webdriver.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit dem Selen-Webtreiber

Bemerkungen

In diesem Abschnitt erhalten Sie einen Überblick darüber, was ein Selen-Web-Treiber ist und warum ein Entwickler ihn verwenden möchte.

Es sollte auch alle großen Themen innerhalb des Selen-Webdrivers erwähnen und auf die verwandten Themen verweisen. Da die Dokumentation für Selen-Web-Treiber neu ist, müssen Sie möglicherweise erste Versionen dieser verwandten Themen erstellen.

Versionen

Ausführung	Veröffentlichungsdatum
0.0.1	2016-08-03

Examples

Installation oder Setup

Um mit WebDriver arbeiten zu können, benötigen Sie den entsprechenden Treiber von der Selenium-Site: Selenium HQ Downloads . Hier müssen Sie den Treiber herunterladen, der für die Browser und / oder Plattformen relevant ist, auf denen Sie WebDriver ausführen möchten. Wenn Sie beispielsweise in Chrome getestet haben, werden Sie von der Selenium-Site zu folgendem Ziel geleitet:

https://sites.google.com/a/chromium.org/chromedriver/

Zum Herunterladen von `chromedriver.exe` .

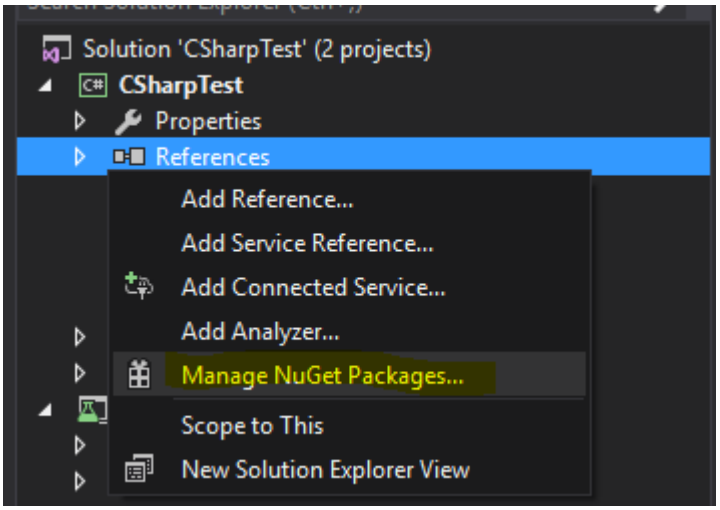
Bevor Sie WebDriver verwenden können, müssen Sie schließlich die entsprechenden Sprachbindungen herunterladen. Wenn Sie beispielsweise C # verwenden, können Sie auf den Download von der Seite für die HQ-Downloads von Selenium zugreifen, um die erforderlichen DLL-Dateien abzurufen oder sie als Pakete in Visual Studio herunterzuladen über den NuGet-Paketmanager.

Die erforderlichen Dateien sollten jetzt heruntergeladen werden. Informationen zur Verwendung von WebDriver finden Sie in der anderen Dokumentation zu `selenium-webdriver` .

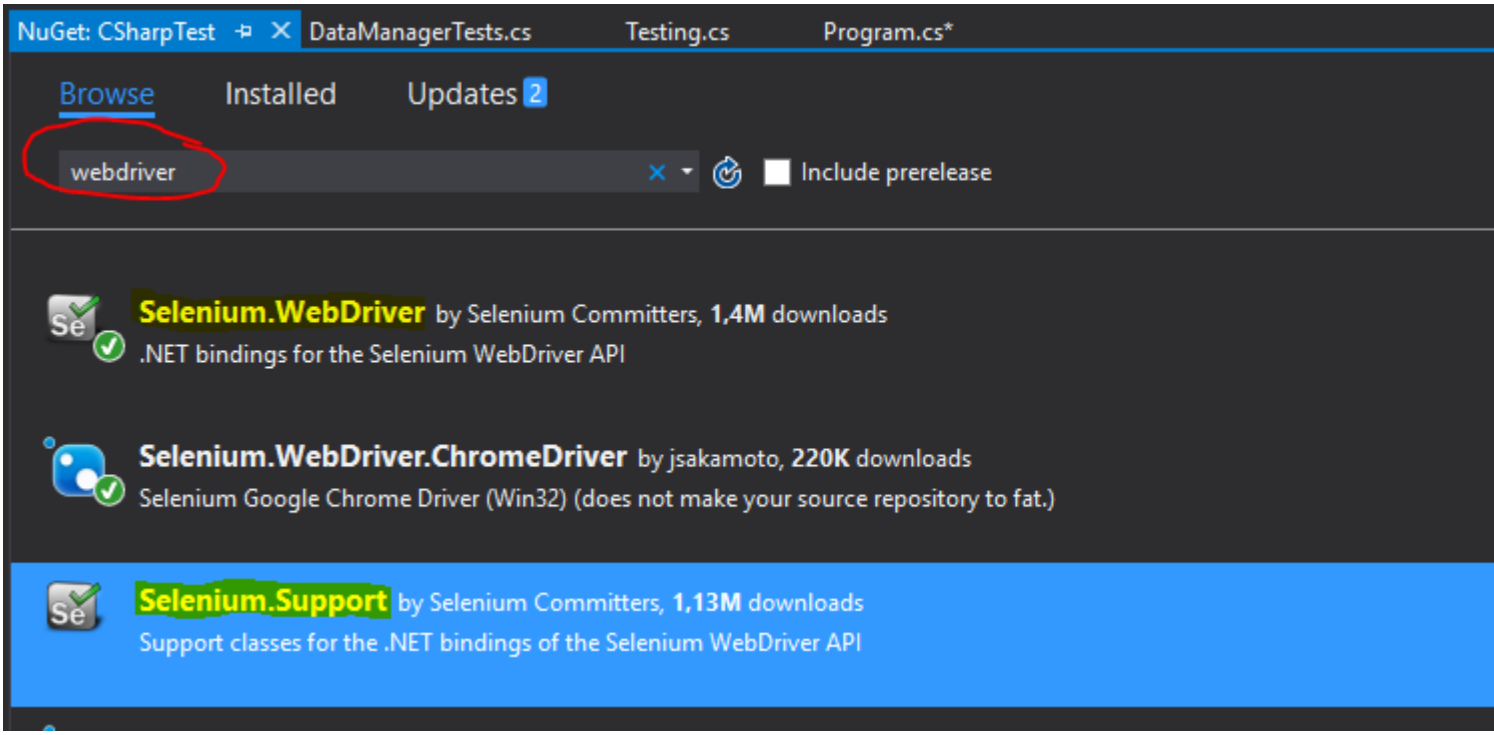
Für Visual Studio [NuGet]

Die einfachste Möglichkeit, Selenium WebDriver zu installieren, ist die Verwendung eines NuGet-Paketmanagers.

Klicken Sie in Ihrem Projekt mit der rechten Maustaste auf "Verweise" und klicken Sie auf "NuGet-Pakete verwalten" wie gezeigt:



Geben Sie dann in das Suchfeld " *Webtreiber* " ein. Sie sollten dann so etwas sehen:



Installieren Sie " **Selenium.WebDriver** " und " **Selenium.Support** " (das Support-Paket enthält zusätzliche Ressourcen, z. B. Warten), indem Sie auf der rechten Seite auf die Schaltfläche Installieren klicken.

Dann können Sie Ihre WebDriver installieren, die Sie verwenden möchten, beispielsweise einen der folgenden:

- Selenium.WebDriver.ChromeDriver (Google Chrome)

- PhantomJS (ohne Kopf)
-

Was ist Selenium WebDriver?

Selenium ist ein Satz von Tools zur Automatisierung von Browsern. Es wird häufig für Webanwendungstests auf mehreren Plattformen verwendet. Unter dem Dach von Selenium stehen einige Tools zur Verfügung, beispielsweise Selenium WebDriver (ehemals Selenium RC), Selenium IDE und Selenium Grid.

WebDriver ist eine Fernbedienung *Schnittstelle* , die Sie manipulieren kann DOM - Elemente in Web - Seiten, sowie das Verhalten von Benutzeragenten zu befehlen. Diese Schnittstelle bietet ein sprachneutrales Drahtprotokoll, das für verschiedene Plattformen implementiert wurde, z.

- GeckoDriver (Mozilla Firefox)
- ChromeDriver (Google Chrome)
- SafariDriver (Apple Safari)
- InternetExplorerDriver (MS InternetExplorer)
- MicrosoftWebDriver oder EdgeDriver (MS Edge)
- OperaChromiumDriver (Opera-Browser)

sowie andere Implementierungen:

- EventFiringWebDriver
- HtmlUnitDriver
- PhantomJSDriver
- RemoteWebDriver

Selenium WebDriver ist eines der Selenium-Tools, das objektorientierte APIs in verschiedenen Sprachen bereitstellt, um mehr Kontrolle und die Anwendung von Standardsoftware-Entwicklungspraktiken zu ermöglichen. Um die Art und Weise, wie ein Benutzer mit einer Webanwendung interagiert, genau zu simulieren, verwendet er "Native OS Level Events" im Gegensatz zu "Synthesized JavaScript Events".

Links zum Verweisen:

- http://www.seleniumhq.org/
- http://www.aosabook.org/de/selenium.html
- https://www.w3.org/TR/webdriver/

Installation oder Einrichtung für Java

Um Tests mit Selenium Webdriver und Java als Programmiersprache zu schreiben, müssen Sie JAR-Dateien von Selenium Webdriver von der Selenium-Website herunterladen.

Es gibt mehrere Möglichkeiten, ein Java-Projekt für den Selenium-Web-Treiber einzurichten. Eine der einfachsten von allen ist die Verwendung von Maven. Maven lädt die erforderlichen Java-Bindungen für den Selenium-Web-Treiber einschließlich aller Abhängigkeiten herunter. Die andere

Möglichkeit ist, die JAR-Dateien herunterzuladen und in Ihr Projekt zu importieren.

Schritte zum Einrichten des Selenium Webdriver-Projekts mit Maven:

1. Installieren Sie maven unter Windows dieses Dokument: https://maven.apache.org/install.html
2. Erstellen Sie einen Ordner mit dem Namen `selenium-learing`
3. Erstellen Sie eine Datei im obigen Ordner mit einem beliebigen Texteditor mit dem Namen `pom.xml`
4. Kopieren Sie den folgenden Inhalt in `pom.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
    <project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
            <modelVersion>4.0.0</modelVersion>
            <groupId>SeleniumLearning</groupId>
            <artifactId>SeleniumLearning</artifactId>
            <version>1.0</version>
            <dependencies>
                <dependency>
                    <groupId>org.seleniumhq.selenium</groupId>
                    <artifactId>selenium-learning</artifactId>
                    <version>3.0.0-beta1</version>
                </dependency>
            </dependencies>
    </project>
```

Hinweis : Stellen Sie sicher, dass die von Ihnen oben angegebene Version die neueste ist. Sie können die neueste Version von hier aus überprüfen: http://docs.seleniumhq.org/download/maven.jsp

5. Führen Sie in der Befehlszeile den folgenden Befehl in das Projektverzeichnis aus.

   ```
   mvn clean install
   ```

Der obige Befehl lädt alle erforderlichen Abhängigkeiten herunter und fügt sie dann dem Projekt hinzu.

6. Schreiben Sie den folgenden Befehl, um ein Eclipse-Projekt zu generieren, das Sie in die Eclipse-IDE importieren können.

   ```
   mvn eclipse:eclipse
   ```

7. Um das Projekt in Eclipse Ide zu importieren, können Sie die folgenden Schritte ausführen

   Öffnen Sie Elipse -> Datei -> Importieren -> Allgemein -> Vorhandenes Projekt in den Arbeitsbereich -> Weiter -> Durchsuchen -> Suchen Sie den Ordner, in dem pom.xml enthalten ist -> OK -> Fertig stellen

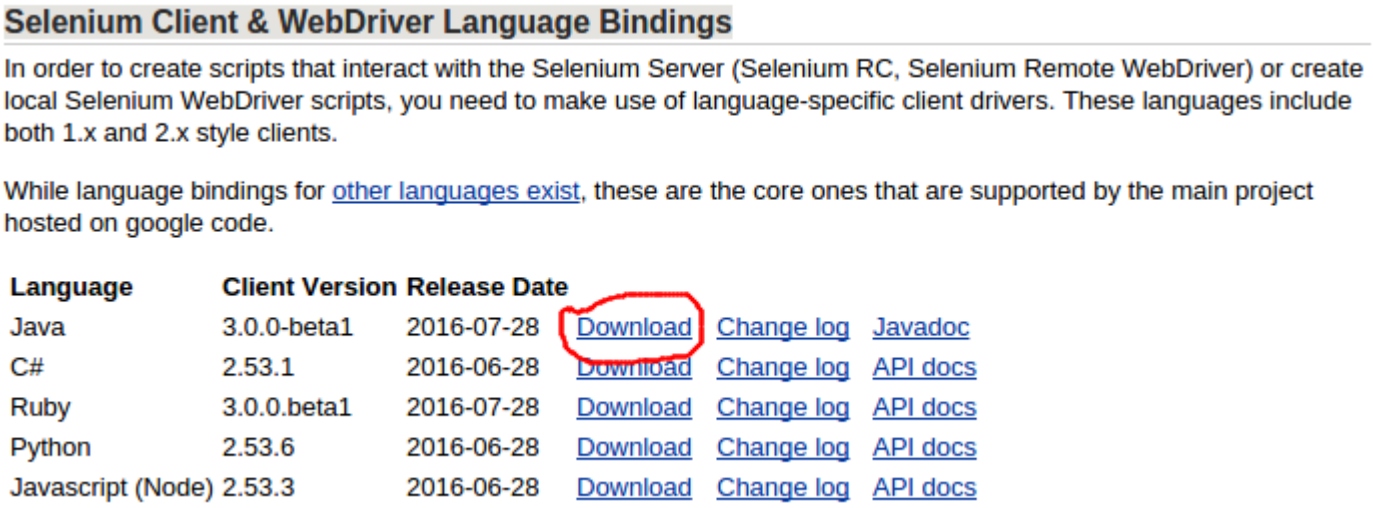
Installieren Sie das m2eclipse-Plugin, indem Sie mit der rechten Maustaste auf Ihr Projekt klicken und Maven -> Dependency Management aktivieren.

Schritte zum Einrichten des Selenium Webdriver-Projekts mithilfe von Jar-Dateien

1. Erstellen Sie ein neues Projekt in Eclipse, indem Sie die folgenden Schritte ausführen.

   Öffnen Sie Elipse -> Datei -> Neu -> Java-Projekt -> Geben Sie einen Namen an (Selen-Lernen) -> Fertig stellen

2. Laden Sie JAR-Dateien von http://www.seleniumhq.org/download/ herunter. Sie müssen sowohl **Selenium Standalone Server** als auch **Selenium Client- und WebDriver-Sprachbindungen** herunterladen. Da in diesem Dokument von Java die Rede ist, müssen Sie nur jar aus dem Java-Bereich herunterladen. Schauen Sie sich den beigefügten Screenshot an.



Selenium Client & WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers. These languages include both 1.x and 2.x style clients.

While language bindings for other languages exist, these are the core ones that are supported by the main project hosted on google code.

Language	Client Version	Release Date			
Java	3.0.0-beta1	2016-07-28	Download	Change log	Javadoc
C#	2.53.1	2016-06-28	Download	Change log	API docs
Ruby	3.0.0.beta1	2016-07-28	Download	Change log	API docs
Python	2.53.6	2016-06-28	Download	Change log	API docs
Javascript (Node)	2.53.3	2016-06-28	Download	Change log	API docs

Hinweis: Selenium Standalone Server ist nur erforderlich, wenn Sie zum Ausführen der Tests einen Remote-Server verwenden möchten. Da dieses Dokument alles über das Einrichten des Projekts hinausgeht, ist es besser, alles an Ort und Stelle zu haben.

3. Die Gläser werden in einer ZIP-Datei heruntergeladen und entpackt. Sie sollten `.jar` direkt sehen können.

4. Klicken Sie in Eclipse mit der rechten Maustaste auf das Projekt, das Sie in Schritt 1 erstellt haben, und führen Sie die folgenden Schritte aus.

   Eigenschaften -> Java-Erstellungspfad -> Registerkarte "Bibliotheken auswählen" -> Klicken Sie auf "Externe Jars hinzufügen" -> Suchen Sie den entpackten JAR-Ordner, den Sie oben heruntergeladen haben -> Wählen Sie alle Jars aus dem `lib` Ordner aus -> Klicken Sie auf "OK". > Suchen Sie denselben entpackten Ordner -> Wählen Sie die jar `client-combined-3.0.0-beta1-nodeps.jar` außerhalb des lib-Ordners aus (`client-combined-3.0.0-beta1-nodeps.jar`) -> `client-combined-3.0.0-beta1-nodeps.jar`

Fügen Sie auf ähnliche Weise den `Selenium Standalone Server` nach dem obigen Schritt hinzu.

5. Jetzt können Sie mit dem Schreiben von Selencode in Ihr Projekt beginnen.

PS : Die obige Dokumentation basiert auf der Selenium-3.0.0-Betaversion, sodass sich die Namen

der angegebenen JAR-Dateien mit der Version ändern können.

Erste Schritte mit dem Selen-Webtreiber online lesen: https://riptutorial.com/de/selenium-webdriver/topic/878/erste-schritte-mit-dem-selen-webtreiber

Kapitel 2: Aktionen (Emulieren komplexer Benutzergesten)

Einführung

Die `Actions` Klasse ermöglicht es uns, genau zu emulieren, wie ein Benutzer mit einer Webseite / Elementen interagieren würde. Mit einer Instanz dieser Klasse können Sie eine Reihe von Aktionen beschreiben, z. B. Klicken, Doppelklicken, Ziehen, Drücken von Tasten usw. Wenn diese Aktionen beschrieben sind, müssen Sie die Aktionen erstellen, um die Aktionen auszuführen (`.Build()`) und weist sie an, ausgeführt zu werden (`.Perform()`). Also müssen wir beschreiben, bauen, durchführen. Die folgenden Beispiele werden darauf eingehen.

Syntax

- dragAndDrop (WebElement-Quelle, WebElement-Ziel)
- dragAndDropBy (WebElement-Quelle, int xOffset, int yOffset)
- ausführen()

Parameter

Parameter	Einzelheiten
Quelle	Element zum Emulieren der Schaltfläche unten.
Ziel	Element zum Bewegen und Loslassen der Maus.
xOffset	x Koordinate zum Umzug.
yOffset	y koordinieren, um zu bewegen.

Bemerkungen

Dieser Abschnitt enthält Informationen zur Actions-Klasse von Selenium WebDriver. Die Actions-Klasse bietet Ihnen praktische Methoden zum Ausführen komplexer Benutzergesten wie Ziehen und Ablegen, Halten und Klicken usw.

Examples

Drag & Drop

C

```
using OpenQA.Selenium;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Interactions;

namespace WebDriverActions
{
    class WebDriverTest
    {
        static void Main()
        {
            IWebDriver driver = new FirefoxDriver();

            driver.Navigate().GoToUrl("");
            IWebElement source = driver.FindElement(By.CssSelector(""));
            IWebElement target = driver.FindElement(By.CssSelector(""));
            Actions action = new Actions(driver);
            action.DragAndDrop(source, target).Perform();
        }
    }
}
```

Das Obige sucht nach einem `IWebElement` , `source` , und zieht es an das zweite `IWebElement target` und `IWebElement` .

Java

Drag & Drop mit Quell- und Ziel-Webelement.

Eine bequeme Methode, bei der an der Position des Quellelements ein Klicken und Halten ausgeführt wird, bewegt sich an die Position des Zielelements und gibt dann die Maus frei.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.interactions.Actions;

/**
 * Drag and Drop test using source and target webelement
 */
public class DragAndDropClass {
    public static void main(String[] args) {
        WebDriver driver = new FirefoxDriver();
        driver.get("");
        WebElement source = driver.findElement(By.cssSelector(""));
        WebElement target = driver.findElement(By.cssSelector(""));
        Actions action = new Actions(driver);
        action.build();
        action.dragAndDrop(source, target).perform();
    }
}
```

Ziehen Sie ein Element und legen Sie es an einem bestimmten Versatz ab.

Eine bequeme Methode, bei der das Klicken und Halten an der Position des Quellelements

ausgeführt wird, um einen bestimmten Versatz (x und y, beide Ganzzahlen) verschoben wird, und die Maus losgelassen wird.

```
WebElement source = driver.findElement(By.cssSelector(""));
Actions action = new Actions(driver);
action.build()
action.dragAndDropBy(source, x, y).perform(); // x and y are integers value
```

Zu Element wechseln

C

Angenommen, Sie möchten testen, dass eine Dropdown-Liste angezeigt wird, wenn Sie mit der Maus über ein Element fahren. Sie können den Inhalt dieser Liste überprüfen oder eine Option aus der Liste auswählen.

Erstellen Sie zuerst eine Aktion, um den Mauszeiger über das Element zu bewegen *(z. B. hat mein Element den Linktext "Admin")* :

```
Actions mouseHover = new Actions(driver);
mouseHover.MoveToElement(driver.FindElement(By.LinkText("Admin"))).Perform();
```

Im obigen Beispiel:

- Sie haben die Aktion `mouseHover`
- Sie haben dem `driver` befohlen, zu einem bestimmten Element zu wechseln
- Von hier aus können Sie andere ausführen `Actions` mit dem `mouseHover` Objekt oder beim Testen mit Ihrem fortsetzen `driver` Objekt

Dieser Ansatz ist besonders nützlich, wenn Sie auf ein Element klicken, das eine andere Funktion als das Bewegen über das Element ausführt.

Ein vollständiges Beispiel:

```
Actions mouseHover = new Actions(driver);
mouseHover.MoveToElement(driver.FindElement(By.LinkText("Admin"))).Perform();

Assert.IsTrue(driver.FindElement(By.LinkText("Edit Record")).Displayed);
Assert.IsTrue(driver.FindElement(By.LinkText("Delete Record")).Displayed);
```

Aktionen (Emulieren komplexer Benutzergesten) online lesen: https://riptutorial.com/de/selenium-webdriver/topic/4849/aktionen--emulieren-komplexer-benutzergesten-

Kapitel 3: Auffinden von Webelementen

Syntax

- ByChained (params By [] bys)

Bemerkungen

Elemente werden in Selen mithilfe von *Locators* und der `By` Klasse gefunden. Um ein robustes Automatisierungsprojekt mit Selenium zu erstellen, sollten Locators für Web Elements auf intelligente Weise verwendet werden. Die Locators sollten **deskriptiv und eindeutig sein, und es ist unwahrscheinlich,** dass sich die Locators **ändern,** sodass Sie zum Beispiel in Tests keine falschen Ergebnisse erhalten. Die Priorität ist zu verwenden:

1. **ID** - da es einzigartig ist und Sie genau das Element erhalten, das Sie möchten.
2. **Klassenname** - Beschreibt den Namen und kann in einem bestimmten Kontext eindeutig sein.
3. **CSS** (bessere Leistung als xpath) - Für kompliziertere Selektoren.
4. **XPATH** - Wo CSS nicht verwendet werden kann (XPATH Axis), z. B. `div::parent` .

Der Rest der Locators ist anfällig für Änderungen oder Rendering und sollte möglichst vermieden werden.

Faustregel: Wenn Ihr Code ein bestimmtes Element nicht finden kann, könnte dies daran liegen, dass Ihr Code nicht auf das Herunterladen aller DOM-Elemente gewartet hat. Erwägen Sie, Ihrem Programm zu sagen, dass es eine kurze Zeit "warten" soll (versuchen Sie es mit 3-5 Sekunden und erhöhen Sie es dann nach Bedarf langsam), bevor Sie das Element suchen. Hier ist ein Beispiel aus Python, das aus dieser Frage stammt :

```
from selenium import webdriver
import time

browser = webdriver.Firefox()
browser.get("https://app.website.com")

reports_element = browser.find_element_by_xpath("//button[contains(text(), 'Reports')]")

# Element not found! Try giving time for the browser to download all DOM elements:
time.sleep(10)

reports_element = browser.find_element_by_xpath("//button[contains(text(), 'Reports')]")
# This returns correct value!
```

Examples

Seitenelemente mit WebDriver suchen

Um mit WebElements auf einer Webseite zu interagieren, müssen wir zuerst den Ort des Elements identifizieren.

Mit ist das Schlüsselwort in Selen verfügbar.

Sie können die Elemente durch suchen.

1. **Nach** *ID*
2. **Nach** *Klassenname*
3. **Von** *TagName*
4. **Nach** *Name*
5. **Durch** *Link-Text*
6. **Durch** *teilweise Link-Text*
7. **Mit der** *CSS-Auswahl*
8. **Mit** *XPath*
9. **Verwendung von** *JavaScript*

Betrachten Sie das folgende Skriptbeispiel

```
<form name="loginForm">
Login Username: <input id="username" name="login" type="text" />
Password: <input id="password" name="password" type="password" />
<input name="login" type="submit" value="Login" />
```

Im obigen Code werden der Benutzername und das Passwort mithilfe von IDs festgelegt. Jetzt identifizieren Sie die Elemente mit der ID.

```
driver.findElement(By.id(username));

driver.findElement(By.id(password));
```

Da Selen 7 verschiedene Sprachen unterstützt, erhalten Sie in diesem Dokument eine Idee, die Elemente in allen Sprachen zu finden.

Nach ID

Beispiel für das Finden eines Elements mit der ID:

```
<div id="coolestWidgetEvah">...</div>

Java       -  WebElement element = driver.findElement(By.id("coolestWidgetEvah"));
C#         -  IWebElement element = driver.FindElement(By.Id("coolestWidgetEvah"));
Python     -  element = driver.find_element_by_id("coolestWidgetEvah")
Ruby       -  element = driver.find_element(:id, "coolestWidgetEvah")
JavaScript/Protractor -  var elm = element(by.id("coolestWidgetEvah"));
```

Nach Klassenname

Beispiel für das Finden eines Elements mithilfe des Klassennamens:

```
<div class="cheese"><span>Cheddar</span></div>

Java       -  WebElement element = driver.findElement(By.className("cheese"));
C#         -  IWebElement element = driver.FindElement(By.ClassName("cheese"));
Python     -  element = driver.find_element_by_class_name("cheese")
Ruby       -  cheeses = driver.find_elements(:class, "cheese")
JavaScript/Protractor -  var elm = element(by.className("cheese"));
```

Nach Tag-Name

Beispiel für das Finden eines Elements anhand des Tagnamens:

```
<iframe src="..."></iframe>

Java       -  WebElement element = driver.findElement(By.tagName("iframe"));
C#         -  IWebElement element = driver.FindElement(By.TagName("iframe"));
Python     -  element = driver.find_element_by_tag_name("iframe")
Ruby       -  frame = driver.find_element(:tag_name, "iframe")
JavaScript/Protractor -  var elm = element(by.tagName("iframe"));
```

Namentlich

Beispiel für das Finden eines Elements anhand des Namens:

```
<input name="cheese" type="text"/>

Java       -  WebElement element = driver.findElement(By.name("cheese"));
C#         -  IWebElement element = driver.FindElement(By.Name("cheese"));
Python     -  element = driver.find_element_by_name("cheese")
Ruby       -  cheese = driver.find_element(:name, "cheese")
JavaScript/Protractor -  var elm = element(by.name("cheese"));
```

Durch Link-Text

Beispiel für das Finden eines Elements mit Linktext:

```
<a href="http://www.google.com/search?q=cheese">cheese</a>>

Java       -  WebElement element = driver.findElement(By.linkText("cheese"));
C#         -  IWebElement element = driver.FindElement(By.LinkText("cheese"));
Python     -  element = driver.find_element_by_link_text("cheese")
Ruby       -  cheese = driver.find_element(:link, "cheese")
JavaScript/Protractor -  var elm = element(by.linkText("cheese"));
```

Durch teilweise Link-Text

Beispiel für die Suche nach einem Element mit teilweise Verknüpfungstext:

```
<a href="http://www.google.com/search?q=cheese">search for cheese</a>>

Java       -  WebElement element = driver.findElement(By.partialLinkText("cheese"));
C#         -  IWebElement element = driver.FindElement(By.PartialLinkText("cheese"));
Python     -  element = driver.find_element_by_partial_link_text("cheese")
Ruby       -  cheese = driver.find_element(:partial_link_text, "cheese")
JavaScript/Protractor -  var elm = element(by.partialLinkText("cheese"));
```

Mit CSS-Selektoren

Beispiel für das Finden eines Elements mit CSS-Selectors:

```
<div id="food" class="dairy">milk</span>

Java       -  WebElement element = driver.findElement(By.cssSelector("#food.dairy")); //# is
used to indicate id and . is used for classname.
C#         -  IWebElement element = driver.FindElement(By.CssSelector("#food.dairy"));
Python     -  element = driver.find_element_by_css_selector("#food.dairy")
Ruby       -  cheese = driver.find_element(:css, "#food span.dairy.aged")
JavaScript/Protractor -  var elm = element(by.css("#food.dairy"));
```

Hier ist ein Artikel zum Erstellen von CSS-Selektoren:
http://www.w3schools.com/cssref/css_selectors.asp

Mit XPath

Beispiel für das Finden eines Elements mit XPath:

```
<input type="text" name="example" />

Java       -  WebElement element = driver.findElement(By.xpath("//input"));
C#         -  IWebElement element = driver.FindElement(By.XPath("//input"));
Python     -  element = driver.find_element_by_xpath("//input")
Ruby       -  inputs = driver.find_elements(:xpath, "//input")
JavaScript/Protractor -  var elm = element(by.xpath("//input"));
```

Hier ist ein Artikel über XPath: http://www.w3schools.com/xsl/xpath_intro.asp

---

Verwendung von JavaScript

Sie können ein beliebiges Javascript ausführen, um ein Element zu finden. Solange Sie ein DOM-Element zurückgeben, wird es automatisch in ein WebElement-Objekt konvertiert.

Einfaches Beispiel für eine Seite, auf der jQuery geladen ist:

```
Java       -  WebElement element = (WebElement)
((JavascriptExecutor)driver).executeScript("return $('.cheese')[0]");

C#         -  IWebElement element = (IWebElement)
((IJavaScriptExecutor)driver).ExecuteScript("return $('.cheese')[0]");

Python     -  element = driver.execute_script("return $('.cheese')[0]");
Ruby       -  element = driver.execute_script("return $('.cheese')[0]")
JavaScript/Protractor -
```

Bitte beachten Sie: Diese Methode funktioniert nicht, wenn Ihr bestimmter WebDriver kein JavaScript unterstützt, z. B. SimpleBrowser .

Auswahl nach mehreren Kriterien [C #]

Es ist auch möglich, Selektoren zusammen zu verwenden. Verwenden Sie dazu das `OpenQA.Selenium.Support.PageObjects.ByChained` Objekt:

```
element  = driver.FindElement(new ByChained(By.TagName("input"), By.ClassName("class"));
```

Beliebig viele `By` s können verkettet werden und werden als AND-Typ-Auswahl verwendet (dh alle `By` s stimmen überein).

Elemente auswählen, bevor die Seite nicht mehr geladen wird

Beim Aufruf von `driver.Navigate().GoToUrl(url);` wird die Codeausführung angehalten, bis die Seite vollständig geladen ist. Dies ist manchmal nicht erforderlich, wenn Sie nur Daten extrahieren möchten.

Hinweis: Die folgenden Codebeispiele können als Hacks angesehen werden. Es gibt keinen "offiziellen" Weg, dies zu tun.

Erstellen Sie einen neuen Thread

Erstellen und starten Sie einen Thread zum Laden einer Webseite, und verwenden Sie dann Wait .

C #

```
using (var driver = new ChromeDriver())
{
    new Thread(() =>
    {
        driver.Navigate().GoToUrl("http://stackoverflow.com");
    }).Start();

    new WebDriverWait(driver, TimeSpan.FromSeconds(10))
      .Until(ExpectedConditions.ElementIsVisible(By.XPath("//div[@class='summary']/h3/a")));
}
```

Verwenden Sie Timeouts

Mit einem WebDriverTimeout können Sie eine Seite laden. Nach einer bestimmten Zeit wird eine Ausnahme ausgelöst, die das Laden der Seite stoppt. Im catch-Block können Sie Wait verwenden .

C #

```
using (var driver = new ChromeDriver())
{
    driver.Manage().Timeouts().SetPageLoadTimeout(TimeSpan.FromSeconds(5));

    try
    {
        driver.Navigate().GoToUrl("http://stackoverflow.com");
    }
    catch (WebDriverTimeoutException)
    {
        new WebDriverWait(driver, TimeSpan.FromSeconds(10))
          .Until(ExpectedConditions.ElementIsVisible
            (By.XPath("//div[@class='summary']/h3/a")));
    }
}
```

Das Problem : Wenn Sie das Timeout zu kurz einstellen, wird die Seite nicht mehr geladen, unabhängig davon, ob das gewünschte Element vorhanden ist. Wenn Sie das Timeout zu lange einstellen, wird der Leistungsvorteil aufgehoben.

Auffinden von Webelementen online lesen: https://riptutorial.com/de/selenium-webdriver/topic/3991/auffinden-von-webelementen

Kapitel 4: Ausnahmen in Selenium-WebDriver

Einführung

Es gibt eine Reihe von Ausnahmen, die bei Verwendung eines Web-Treibers ausgelöst werden können. Die folgenden Beispiele sollen eine Vorstellung davon vermitteln, was sie bedeuten.

Examples

Python-Ausnahmen

Selen-Ausnahmedokumentation

ElementNotInteractableException: Wird ausgelöst, wenn ein Element im DOM vorhanden ist, Interaktionen mit diesem Element jedoch aufgrund der **Zeichenreihenfolge** auf ein anderes Element treffen

- **ElementNotSelectableException: Wird ausgelöst,** wenn versucht wird, ein nicht **auswählbares** Element auszuwählen. Beispiele für nicht auswählbare Elemente:
  - Skript
- **ElementNotVisibleException: Wird ausgelöst,** wenn ein Element im DOM vorhanden ist, jedoch nicht sichtbar ist und daher nicht mit ihm interagieren kann. Am häufigsten beim Versuch, Text eines Elements anzuklicken oder zu lesen, das nicht sichtbar ist.
- **ErrorInResponseException: Wird ausgelöst,** wenn auf der Serverseite ein Fehler aufgetreten ist. Dies kann bei der Kommunikation mit der Firefox-Erweiterung oder dem Remote-Treiberserver auftreten.
- **ImeActivationFailedException: Wird ausgelöst,** wenn eine IME-Engine aktiviert wurde.
- **ImeNotAvailableException: Wird ausgelöst,** wenn keine IME-Unterstützung verfügbar ist. Diese Ausnahme wird bei jedem IME-bezogenen Methodenaufruf ausgelöst, wenn auf dem Computer keine IME-Unterstützung verfügbar ist.
- **InvalidArgumentException:** Die an einen Befehl übergebenen Argumente sind entweder ungültig oder fehlerhaft.
- **InvalidCookieDomainException: Wird ausgelöst,** wenn versucht wird, ein Cookie unter einer anderen Domäne als der aktuellen URL hinzuzufügen.
- **InvalidElementStateException: Wird ausgelöst,** wenn eine Aktion zu einem ungültigen Status für ein Element führen würde. Unterklassen:
  - ElementNotInteractableException
  - ElementNotSelectableException
  - ElementNotVisibleException
- **InvalidSelectorException: Wird ausgelöst,** wenn der Selektor, mit dem ein Element gesucht wird, kein WebElement zurückgibt. Derzeit geschieht dies nur, wenn der Selektor ein xpath-Ausdruck ist und entweder syntaktisch ungültig ist (dh es handelt sich nicht um einen xpath-Ausdruck) oder der Ausdruck keine WebElements auswählt (z. B. "count (// input)").

- **InvalidSwitchToTargetException: Wird ausgelöst,** wenn das zu schaltende **Frame-** oder Fensterziel nicht vorhanden ist.
- **MoveTargetOutOfBoundsException: Wird ausgelöst,** wenn das Ziel, das für die move () - Methode von **ActionsChains** bereitgestellt wird, ungültig ist, dh außerhalb des Dokuments liegt.
- **NoAlertPresentException: Wird ausgelöst,** wenn keine Warnung **angezeigt wird** . Dies kann durch den Aufruf einer Operation in der Alert () - Klasse verursacht werden, wenn noch keine Warnung auf dem Bildschirm angezeigt wird.
- **NoSuchAttributeException: Wird ausgelöst,** wenn das Attribut des Elements nicht gefunden wurde. Möglicherweise möchten Sie überprüfen, ob das Attribut in dem bestimmten Browser vorhanden ist, mit dem Sie testen. Einige Browser haben möglicherweise unterschiedliche Eigenschaftennamen für dieselbe Eigenschaft. (IE8's .innerText vs. Firefox .textContent)
- **NoSuchElementException: Wird ausgelöst,** wenn das Element nicht gefunden wurde. Wenn Sie auf diese Ausnahme stoßen, können Sie Folgendes überprüfen:
    - Überprüfen Sie Ihren Selektor in Ihrem find_by ...
    - Das Element ist möglicherweise zum Zeitpunkt des Suchvorgangs noch nicht auf dem Bildschirm (die Webseite wird noch geladen). Siehe selenium.webdriver.support.wait.WebDriverWait (), um zu erfahren, wie ein Wait-Wrapper geschrieben wird, der auf das Erscheinen eines Elements wartet.
- **NoSuchFrameException: Wird ausgelöst,** wenn das **umzuschaltende Bildziel** nicht vorhanden ist.
- **NoSuchWindowException: Wird ausgelöst,** wenn das **umzuschaltende** Fensterziel nicht vorhanden ist. Um den aktuellen Satz aktiver Fenstergriffe zu finden, können Sie eine Liste der aktiven Fenstergriffe auf folgende Weise erhalten:
  `print driver.window_handles`
- **RemoteDriverServerException:**
- **StaleElementReferenceException: Wird ausgelöst,** wenn ein Verweis auf ein Element jetzt "veraltet" ist. Veraltet bedeutet, dass das Element nicht mehr im DOM der Seite angezeigt wird. Mögliche Ursachen für StaleElementReferenceException sind unter anderem:
    - Sie befinden sich nicht mehr auf derselben Seite, oder die Seite wurde möglicherweise aktualisiert, seit das Element gefunden wurde.
    - Das Element wurde möglicherweise entfernt und erneut zum Bildschirm hinzugefügt, da es gefunden wurde. Wie ein Element, das verschoben wird. Dies kann in der Regel bei einem Javascript-Framework der Fall sein, wenn Werte aktualisiert und der Knoten neu erstellt wird.
    - Das Element befindet sich möglicherweise in einem iframe oder einem anderen Kontext, der aktualisiert wurde.
- **TimeoutException: Wird ausgelöst,** wenn ein Befehl nicht rechtzeitig ausgeführt wird.
- **UnableToSetCookieException: Wird ausgelöst,** wenn ein Treiber keinen Cookie setzen kann.
- **UnexpectedAlertPresentException:** Wird ausgelöst, wenn eine unerwartete Warnung angezeigt wird. Wird normalerweise ausgelöst, wenn ein erwarteter Modal die Ausführung weiterer Befehle blockiert.
- **UnexpectedTagNameException:** Wird ausgelöst, wenn eine Supportklasse kein erwartetes Webelement abruft.

- **WebDriverException:** Basis-Webtreiber-Ausnahme. Alle Webdriver-Ausnahmen verwenden entweder WebDriverException oder InvalidStateException als übergeordnete Klasse.

Ausnahmen in Selenium-WebDriver online lesen: https://riptutorial.com/de/selenium-webdriver/topic/10546/ausnahmen-in-selenium-webdriver

Kapitel 5: Basic Selenium Webdriver-Programm

Einführung

Dieses Thema soll das grundlegende Webtreiberprogramm in selengestützten Sprachen wie C #, Groovy, Java, Perl, PHP, Python und Ruby zeigen.

Journey beinhaltet das Öffnen des Browsertreibers -> Google Page -> Herunterfahren des Browsers

Examples

C

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

namespace BasicWebdriver
{
    class WebDriverTest
    {
        static void Main()
        {
            using (var driver = new ChromeDriver())
            {
                driver.Navigate().GoToUrl("http://www.google.com");
            }
        }
    }
}
```

Das obige "Programm" navigiert zur Google-Startseite und schließt den Browser nach dem vollständigen Laden der Seite.

```
using (var driver = new ChromeDriver())
```

Dadurch wird ein neues WebDriver-Objekt mithilfe der `IWebdriver` Schnittstelle `IWebdriver` und eine neue Browserfensterinstanz erstellt. In diesem Beispiel verwenden wir `ChromeDriver` (dies könnte jedoch durch den entsprechenden Treiber für den gewünschten Browser ersetzt werden). Wir wickeln diese mit einer `using` Aussage, weil `IWebDriver` implementiert `IDisposable` , so brauchen nicht explizit in den Typ `driver.Quit();` .

Falls Sie Ihren WebDriver nicht mit NuGet heruntergeladen haben, müssen Sie ein Argument in Form eines Pfads zu dem Verzeichnis übergeben, in dem sich der Treiber "chromedriver.exe" befindet.

Navigieren

```
driver.Navigate().GoToUrl("http://www.google.com");
```

und

```
driver.Url = "http://www.google.com";
```

Beide Linien machen dasselbe. Sie weisen den Treiber an, zu einer bestimmten URL zu navigieren und zu warten, bis die Seite geladen ist, bevor sie zur nächsten Anweisung wechselt.

Es gibt andere Methoden, die mit der Navigation verknüpft sind, z. B. `Back()` , `Forward()` oder `Refresh()` .

Danach wird der `using` sicher beendet und das Objekt entsorgt.

Python

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

def set_up_driver():
    path_to_chrome_driver = 'chromedriver'
    return webdriver.Chrome(executable_path=path_to_chrome_driver)

def get_google():
    driver = set_up_driver()
    driver.get('http://www.google.com')
    tear_down(driver)

def tear_down(driver):
    driver.quit()

if '__main__' == __name__:
    get_google()
```

Das obige "Programm" navigiert zur Google-Startseite und schließt dann den Browser, bevor Sie den Vorgang abschließen.

```
if '__main__' == __name__:
    get_google()
```

Zuerst haben wir unsere Hauptfunktion, unseren Einstiegspunkt in das Programm, der `get_google()` .

```
def get_google():
    driver = set_up_driver()
```

`get_google()` erstellt zunächst unsere `driver` über `set_up_driver()` :

```
def set_up_driver():
    path_to_chrome_driver = 'chromedriver'
    return webdriver.Chrome(executable_path=path_to_chrome_driver)
```

Dabei geben wir an, wo sich `chromedriver.exe` befindet, und instanziieren unser `chromedriver.exe` mit diesem Pfad. Der Rest von `get_google()` navigiert zu Google:

```
driver.get('http://www.google.com')
```

`tear_down()` dann `tear_down()` übergibt das `tear_down()` :

```
tear_down(driver)
```

`tear_down()` enthält lediglich eine Zeile, um unser `tear_down()` herunterzufahren:

```
driver.quit()
```

Dadurch wird der Treiber angewiesen, alle geöffneten Browserfenster zu schließen und das Browserobjekt zu löschen, da nach diesem Aufruf kein anderer Code vorhanden ist. Dadurch wird das Programm effektiv beendet.

Java

Der folgende Code besteht aus 3 Schritten.

1. Chrome-Browser öffnen
2. Google-Seite öffnen
3. Fahren Sie den Browser herunter

```
import org.openqa.selenium;
import org.openqa.selenium.chrome;

public class WebDriverTest {
    public static void main(String args[]) {
        System.setProperty("webdriver.chrome.driver", "C:\\path\\to\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();

        driver.get("http://www.google.com");
        driver.quit();
    }
}
```

Das obige "Programm" navigiert zur Google-Startseite und schließt dann den Browser, bevor Sie den Vorgang abschließen.

```
System.setProperty("webdriver.chrome.driver", "C:\\path\\to\\chromedriver.exe");
WebDriver driver = new ChromeDriver();
```

In der ersten Zeile wird angegeben, wo sich die ausführbare Datei `ChromeDriver` (chromedriver.exe) befindet. Dann erstellen wir unser `ChromeDriver()` durch Aufruf des `ChromeDriver()` . Wieder könnten

wir unseren Konstruktor hier für jeden Browser / jede Plattform aufrufen.

```
driver.get("http://www.google.com");
```

Dadurch wird unser Treiber angewiesen, zur angegebenen URL zu navigieren: http://www.google.com . Der Java WebDriver API bietet die `get()` Methode direkt auf der WebDriver Schnittstelle, obwohl weitere Navigationsmethoden können über die gefunden werden `navigate()` Methode, zB `driver.navigate.back()` .

Sobald die Seite geladen ist, rufen wir sofort an:

```
driver.quit();
```

Dies weist den Treiber an, alle geöffneten Browserfenster zu schließen und das Treiberobjekt zu löschen, da nach diesem Aufruf kein anderer Code vorhanden ist, wodurch das Programm effektiv beendet wird.

```
driver.close();
```

Ist eine Anweisung (hier nicht dargestellt) an den Treiber, nur das aktive Fenster zu schließen, würden wir in diesem Fall, da wir nur ein einziges Fenster haben, identische Ergebnisse beim Aufruf von `quit()` .

Java - Best Practice bei Seitenklassen

Anwendungsfall: Melden Sie sich beim FB-Konto an

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class FaceBookLoginTest {
    private static WebDriver driver;
    HomePage homePage;
    LoginPage loginPage;
    @BeforeClass
    public void openFBPage(){
        driver = new FirefoxDriver();
        driver.get("https://www.facebook.com/");
        loginPage = new LoginPage(driver);
    }
    @Test
    public void loginToFB(){
       loginPage.enterUserName("username");
       loginPage.enterPassword("password");
       homePage = loginPage.clickLogin();
        System.out.println(homePage.getUserName());
    }

}
```

Seitenklassen: Login-Seite & Startseite Login-Seitenklasse:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

public class LoginPage {

    WebDriver driver;
    public LoginPage(WebDriver driver){
        this.driver = driver;
    }
    @FindBy(id="email")
    private WebElement loginTextBox;

    @FindBy(id="pass")
    private WebElement passwordTextBox;

    @FindBy(xpath = ".//input[@data-testid='royal_login_button']")
    private WebElement loginBtn;

    public void enterUserName(String userName){
        if(loginTextBox.isDisplayed()) {
            loginTextBox.clear();
            loginTextBox.sendKeys(userName);
        }
        else{
            System.out.println("Element is not loaded");
        }
    }
    public void enterPassword(String password){
        if(passwordTextBox.isDisplayed()) {
            passwordTextBox.clear();
            passwordTextBox.sendKeys(password);
        }
        else{
            System.out.println("Element is not loaded");
        }
    }
    public HomePage clickLogin(){
        if(loginBtn.isDisplayed()) {
            loginBtn.click();
        }
        return new HomePage(driver);
    }
}
```

Homepage-Klasse:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

public class HomePage {

    WebDriver driver;
    public HomePage(WebDriver driver){
        this.driver = driver;
    }
```

```
    @FindBy(xpath=".//a[@data-testid='blue_bar_profile_link']/span")
    private WebElement userName;

    public String getUserName(){
        if(userName.isDisplayed()) {
            return userName.getText();
        }
        else {
            return "Username is not present";
        }
    }

}
```

Basic Selenium Webdriver-Programm online lesen: https://riptutorial.com/de/selenium-webdriver/topic/3990/basic-selenium-webdriver-programm

Kapitel 6: Behandeln Sie eine Warnung

Examples

Selen mit Java

So behandeln Sie eine Popup-Warnung in Java mit Selenium:

Es gibt drei Arten von Popups.

1. **Einfache Warnung** : Warnung ("Dies ist eine einfache Warnung");
2. **Bestätigungsalarm** : var popuResult = bestätigen ("Bestätigen Sie das Aufrufen mit OK und Abbrechen");
3. **Eingabeaufforderung** : var person = prompt ("Gefällt Ihnen stackoverflow?", "Ja / Nein");

Welche Art von Popup in ihrem Testfall behandelt werden muss, hängt vom Benutzer ab.

Entweder du kannst

1. accept () um die Benachrichtigung zu akzeptieren
2. Ablehnen () Abmelden der Warnung
3. getText () Ermittelt den Text der Warnung
4. sendKeys () Um dem Alarm einen Text zu schreiben

Für eine einfache Benachrichtigung:

```
Alert simpleAlert = driver.switchTo().alert();
String alertText = simpleAlert.getText();
System.out.println("Alert text is " + alertText);
simpleAlert.accept();
```

Für Bestätigungsalarm:

```
Alert confirmationAlert = driver.switchTo().alert();
String alertText = confirmationAlert.getText();
System.out.println("Alert text is " + alertText);
confirmationAlert.dismiss();
```

Für sofortige Benachrichtigung:

```
Alert promptAlert  = driver.switchTo().alert();
String alertText = promptAlert .getText();
System.out.println("Alert text is " + alertText);
//Send some text to the alert
promptAlert .sendKeys("Accepting the alert");
Thread.sleep(4000); //This sleep is not necessary, just for demonstration
promptAlert .accept();
```

nach ihren bedürfnissen.

Eine andere Möglichkeit, dies zu tun, besteht darin, Ihren Code in einen Try-Catch zu packen:

```
try{
   // Your logic here.
} catch(UnhandledAlertException e){
  Alert alert = driver.switchTo().alert();
  alert.accept();
}
// Continue.
```

C

So schließen Sie eine Popup-Warnung in C # mit Selenium:

```
IAlert alert = driver.SwitchTo().Alert();
// Prints text and closes alert
System.out.println(alert.Text);
alert.Accept();
or
alert.Dismiss();
```

nach ihren bedürfnissen.

Eine andere Möglichkeit, dies zu tun, besteht darin, Ihren Code in einen Try-Catch zu packen:

```
try{
   // Your logic here.
} catch(UnhandledAlertException e){
  var alert = driver.SwitchTo().Alert();
  alert.Accept();
}
// Continue.
```

Python

Es gibt mehrere Möglichkeiten, in `Python` zum Popup-Alarm zu wechseln:

1. *Veraltet* :

```
alert = driver.switch_to_alert()
```

2. *`switch_to`* :

```
alert = driver.switch_to.alert
```

3. *`ExplicitWait`* :

```
 from selenium.webdriver.common.by import By
 from selenium.webdriver.support.ui import WebDriverWait
```

```
from selenium.webdriver.support import expected_conditions as EC

alert = WebDriverWait(driver, TIMEOUT_IN_SECONDS).until(EC.alert_is_present())
```

4. *Durch die Deklaration der Instanz der `Alert` Klasse* :

```
from selenium.webdriver.common.alert import Alert

alert = Alert(driver)
```

So füllen Sie das Eingabefeld im Popup aus, das durch die `JavaScript prompt()` ausgelöst wird:

```
alert.send_keys('Some text to send')
```

So bestätigen Sie das Dialogfenster *:

```
alert.accept()
```

Ablehnen:

```
alert.dismiss()
```

So rufen Sie Text aus dem Popup-Fenster ab:

```
alert.text
```

** PS `alert.dismiss()` kann verwendet werden, um Popups zu bestätigen, die durch `JavaScript alert()` sowie `alert.confirm()` ausgelöst werden.*

Behandeln Sie eine Warnung online lesen: https://riptutorial.com/de/selenium-webdriver/topic/6048/behandeln-sie-eine-warnung

Kapitel 7: Einstellen / Abrufen der Browserfenstergröße

Einführung

Einstellen oder Abrufen der Fenstergröße eines Browsers während der Automatisierung

Syntax

- driver.manage (). window (). maximize ();
- driver.manage (). window (). setSize (*DimensionObject*);
- driver.manage (). window (). getSize ()

Examples

JAVA

Stellen Sie die maximale Browserfenstergröße ein:

```
//Initialize Browser
System.setProperty("webdriver.gecko.driver", "E:\\path\\to\\geckodriver.exe");
WebDriver driver = new FirefoxDriver();
driver.get("https://www.google.com/");

//Set Browser window size
driver.manage().window().maximize();
```

Spezifische Fenstergröße einstellen:

```
//Initialize Browser
System.setProperty("webdriver.gecko.driver", "E:\\path\\to\\geckodriver.exe");
WebDriver driver = new FirefoxDriver();
driver.get("https://www.google.com/");

//Initialize Dimension class object and set Browser window size
org.openqa.selenium.Dimension d = new org.openqa.selenium.Dimension(400, 500);
driver.manage().window().setSize(d);
```

Browserfenstergröße abrufen:

```
//Initialize Browser
System.setProperty("webdriver.gecko.driver", "E:\\path\\to\\geckodriver.exe");
WebDriver driver = new FirefoxDriver();
driver.get("https://www.google.com/");

//Get Browser window size and print on console
System.out.println(driver.manage().window().getSize());
```

Einstellen / Abrufen der Browserfenstergröße online lesen: https://riptutorial.com/de/selenium-webdriver/topic/10093/einstellen---abrufen-der-browserfenstergro-e

Kapitel 8: Fehlerbehandlung bei der Automatisierung mit Selen

Examples

Python

`WebDriverException` ist eine grundlegende `Selenium-WebDriver` Ausnahme, mit der alle anderen `Selenium-WebDriver` Ausnahmen `Selenium-WebDriver`

Um eine Ausnahme abfangen zu können, muss sie zuerst importiert werden:

```
from selenium.common.exceptions import WebDriverException as WDE
```

und dann:

```
try:
    element = driver.find_element_by_id('ID')
except WDE:
    print("Not able to find element")
```

Auf die gleiche Weise können Sie weitere spezifischere Ausnahmen importieren:

```
from selenium.common.exceptions import ElementNotVisibleException
from selenium.common.exceptions import NoAlertPresentException
...
```

Wenn Sie nur eine Ausnahmemeldung extrahieren möchten:

```
from selenium.common.exceptions import UnexpectedAlertPresentException

try:
    driver.find_element_by_tag_name('a').click()
except UnexpectedAlertPresentException as e:
    print(e.__dict__["msg"])
```

Fehlerbehandlung bei der Automatisierung mit Selen online lesen: https://riptutorial.com/de/selenium-webdriver/topic/9548/fehlerbehandlung-bei-der-automatisierung-mit-selen

Kapitel 9: Frames wechseln

Syntax

- **Java**
- driver.switchTo (). frame (Stringname);
- driver.switchTo (). frame (String-ID);
- driver.switchTo (). frame (int index);
- driver.switchTo (). frame (WebElement frameElement);
- driver.switchTo (). defaultContent ();
- **C #**
- driver.SwitchTo (). Frame (int frameIndex);
- driver.SwitchTo (). Frame (IWebElement frameElement);
- driver.SwitchTo (). Frame (Zeichenfolge frameName);
- driver.SwitchTo (). DefaultContent ();
- **Python**
- driver.switch_to_frame (nameOrId)
- driver.switch_to.frame (nameOrId)
- driver.switch_to_frame (index)
- driver.switch_to.frame (Index)
- driver.switch_to_frame (frameElement)
- driver.switch_to.frame (frameElement)
- driver.switch_to_default_content ()
- driver.switch_to.default_content ()
- **JavaScript**
- driver.switchTo (). frame (nameOrId)
- driver.switchTo (). Frame (Index)
- driver.switchTo (). defaultContent ()

Parameter

Parameter	Einzelheiten
nameOrId	Wählen Sie einen Rahmen anhand des Namens der ID aus.
Index	Wählen Sie einen Frame anhand seines nullbasierten Index aus.
frameElement	Wählen Sie einen Frame mit dem zuvor lokalisierten WebElement aus

Examples

So wechseln Sie mit Java zu einem Frame

Für eine Instanz, wenn der HTML-Quellcode einer HTML-Ansicht oder eines HTML-Elements von einem iframe wie folgt umschlossen wird:

```
<iframe src="../images/eightball.gif" name="imgboxName" id="imgboxId">
   <p>iframes example</p>
   <a href="../images/redball.gif" target="imgbox">Red Ball</a>
</iframe><br />
```

Um eine Aktion an den Web-Elementen des Iframes auszuführen, müssen Sie zunächst den Fokus auf den Iframe setzen, indem Sie eine der folgenden Methoden verwenden:

Frame- ID verwenden (sollte nur verwendet werden, wenn Sie die ID des Iframes kennen).

```
driver.switchTo().frame("imgboxId"); //imgboxId - Id of the frame
```

Frame-Name verwenden (sollte nur verwendet werden, wenn Sie den Namen des Iframes kennen).

```
driver.switchTo().frame("imgboxName"); //imgboxName - Name of the frame
```

Frame-Index verwenden (sollte nur verwendet werden, wenn Sie nicht über die ID oder den Namen des Iframes verfügen), wobei der Index die Position des Iframes unter allen Frames definiert.

```
driver.switchTo().frame(0); //0 - Index of the frame
```

Hinweis: Wenn Sie drei Frames auf der Seite haben, befindet sich der erste Frame im Index 0, der zweite im Index 1 und der dritte im Index 2.

Verwendung von zuvor lokalisiertem Webelement (sollte nur verwendet werden, wenn Sie den Frame bereits gefunden und als `WebElement`).

```
driver.switchTo().frame(frameElement); //frameElement - webelement that is the frame
```

Also, um auf den `Red Ball` zu klicken:

```
driver.switchTo().frame("imgboxId");
driver.findElement(By.linkText("Red Ball")).Click();
```

Um mit Java aus einem Frame herauszukommen

So wechseln Sie den Fokus auf das Hauptdokument oder den ersten Frame der Seite. Sie müssen die folgende Syntax verwenden.

```
driver.switchTo().defaultContent();
```

Mit C # zu einem Frame wechseln

1. Wechseln Sie zu einem Frame nach Index.

Hier wechseln wir zu Index 1. Index bezieht sich auf die Reihenfolge der Frames auf der Seite. Dies sollte als letzter Ausweg verwendet werden, da Frame-ID oder -Namen viel zuverlässiger sind.

```
driver.SwitchTo().Frame(1);
```

2. Wechseln Sie zu einem Frame nach Name

```
driver.SwitchTo().Frame("Name_Of_Frame");
```

3. Wechseln Sie zu einem Frame nach Titel, Id oder anderen, indem Sie IWebElement übergeben

Wenn Sie nach ID oder Titel zu einem Frame wechseln möchten, müssen Sie ein Webelement als Parameter übergeben:

```
driver.SwitchTo().Frame(driver.FindElement(By.Id("ID_OF_FRAME")));
driver.SwitchTo().Frame(driver.FindElement(By.CssSelector("iframe[title='Title_of_Frame']")));
```

Beachten Sie auch, dass Sie einige Sekunden warten müssen, wenn Ihr Bild einige Sekunden dauert.

```
new WebDriverWait(driver, TimeSpan.FromSeconds(10))
    .Until(ExpectedConditions.ElementIsVisible(By.Id("Id_Of_Frame")));
```

Raus aus einem Rahmen:

```
driver.SwitchTo().DefaultContent()
```

So verlassen Sie einen Frame mit C

So wechseln Sie den Fokus auf das Hauptdokument oder den ersten Frame der Seite. Sie müssen die folgende Syntax verwenden.

webDriver.SwitchTo (). DefaultContent ();

Wechseln zwischen untergeordneten Frames eines übergeordneten Frames.

Stellen Sie sich vor, Sie haben einen übergeordneten Frame (Frame-Parent). und 2 untergeordnete Frames (Frame_Son, Frame_Daughter). Schauen wir uns verschiedene Bedingungen an und wie man damit umgeht.

1. Vom Elternteil zum Sohn oder zur Tochter:

```
 driver.switchTo().frame("Frame_Son");
 driver.switchTo().frame("Frame_Daughter");
```

2.Von Sohn zu übergeordnetem Element: Wenn das übergeordnete Bild das Standardbild ist, wechseln Sie zum Standardbild, ansonsten wechseln Sie vom Standardbild zum übergeordneten Bild. Sie können jedoch nicht direkt vom Sohn zum Elternteil wechseln.

```
driver.switchTo().defaultContent();
driver.switchTo().frame("Frame_Parent");
```

3.Von Sohn zu Tochter: Wenn Ihre Schwester einen Fehler macht, schreien Sie sie nicht an, erreichen Sie einfach Ihre Eltern. In ähnlicher Weise geben Sie die Kontrolle an den übergeordneten Frame und dann an den Tochterframe.

```
driver.switchTo().defaultContent();
driver.switchTo().frame("Frame_Parent");
driver.switchTo().frame("Frame_Daughter");
```

Warten Sie, bis Ihre Bilder geladen sind

In einigen Fällen wird Ihr Frame möglicherweise nicht sofort angezeigt und Sie müssen wahrscheinlich warten, bis der Frame geladen ist, um zu wechseln. Andernfalls haben Sie NoSuchFrameException.

Daher ist es immer eine gute Wahl, um zu warten, bevor Sie wechseln. Im Folgenden finden Sie eine ideale Methode, um zu warten, bis ein Frame geladen ist.

```
try{
        new WebDriverWait(driver, 300).ignoring(StaleElementReferenceException.class).
                        ignoring(WebDriverException.class).

until(ExpectedConditions.visibilityOf((driver.findElement(By.id("cpmInteractionDivFrame"))));}
    catch{
```

// löst nur dann eine Ausnahme aus, wenn Ihr Frame in Ihrer Wartezeit 300 Sekunden nicht sichtbar ist}

Frames wechseln online lesen: https://riptutorial.com/de/selenium-webdriver/topic/4589/frames-wechseln

Kapitel 10: Headless-Browser

Examples

PhantomJS [C #]

`PhantomJS` ist ein voll ausgestatteter, Headless-Webbrowser **mit** JavaScript-Unterstützung.

Bevor Sie beginnen, müssen Sie einen PhantomJS- Treiber herunterladen. Stellen Sie sicher, dass Sie diesen Code am Anfang Ihres Codes angeben :

```
using OpenQA.Selenium;
using OpenQA.Selenium.PhantomJS;
```

Toll, jetzt zur Initialisierung:

```
var driver = new PhantomJSDriver();
```

Dadurch wird einfach eine neue Instanz der PhantomJSDriver-Klasse erstellt. Sie können es dann wie jeden WebDriver verwenden:

```
using (var driver = new PhantomJSDriver())
{
    driver.Navigate().GoToUrl("http://stackoverflow.com/");

    var questions = driver.FindElements(By.ClassName("question-hyperlink"));

    foreach (var question in questions)
    {
        // This will display every question header on StackOverflow homepage.
        Console.WriteLine(question.Text);
    }
}
```

Das funktioniert gut. Das Problem, auf das Sie wahrscheinlich gestoßen sind, ist, dass `PhantomJS` beim Arbeiten mit der Benutzeroberfläche ein neues Konsolenfenster öffnet, das in den meisten Fällen nicht wirklich erwünscht ist. Glücklicherweise können wir das Fenster ausblenden und die Leistung mit `PhantomJSOptions` und `PhantomJSDriverService` sogar geringfügig verbessern. Vollständiges Arbeitsbeispiel unten:

```
// Options are used for setting "browser capabilities", such as setting a User-Agent
// property as shown below:
var options = new PhantomJSOptions();
options.AddAdditionalCapability("phantomjs.page.settings.userAgent",
"Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:25.0) Gecko/20100101 Firefox/25.0");

// Services are used for setting up the WebDriver to your likings, such as
// hiding the console window and restricting image loading as shown below:
var service = PhantomJSDriverService.CreateDefaultService();
service.HideCommandPromptWindow = true;
```

```
service.LoadImages = false;

// The same code as in the example above:
using (var driver = new PhantomJSDriver(service, options))
{
    driver.Navigate().GoToUrl("http://stackoverflow.com/");

    var questions = driver.FindElements(By.ClassName("question-hyperlink"));

    foreach (var question in questions)
    {
        // This will display every question header on StackOverflow homepage.
        Console.WriteLine(question.Text);
    }
}
```

Pro-Tipp: Klicken Sie auf eine Klasse (z. B. den `PhantomJSDriverService`) und drücken Sie F12, um genau zu sehen, was sie enthalten, und beschreiben Sie kurz, was sie tun.

SimpleBrowser [C #]

`SimpleBrowser` ist ein `SimpleBrowser` WebDriver **ohne** JavaScript-Unterstützung.

Es ist erheblich schneller als ein zuvor genanntes `PhantomJS` , ist jedoch in `PhantomJS` auf die Funktionalität auf einfache Aufgaben ohne ausgefallene Funktionen beschränkt.

Zuerst müssen Sie das SimpleBrowser.WebDriver- Paket herunterladen und dann diesen Code am Anfang einfügen:

```
using OpenQA.Selenium;
using SimpleBrowser.WebDriver;
```

Hier ist ein kurzes Beispiel, wie man es benutzt:

```
using (var driver = new SimpleBrowserDriver())
{
    driver.Navigate().GoToUrl("http://stackoverflow.com/");

    var questions = driver.FindElements(By.ClassName("question-hyperlink"));

    foreach (var question in questions)
    {
        // This will display every question header on StackOverflow homepage.
        Console.WriteLine(question.Text);
    }
}
```

Headless Browser in Java

HTMLUnitDriver

HTMLUnitDriver ist die einfachste Implementierung eines Headless-Browsers (ohne GUI) für Webdriver, der auf HtmlUnit basiert. Es modelliert HTML-Dokumente und stellt eine API zur

Verfügung, mit der Sie Seiten aufrufen, Formulare ausfüllen, auf Links klicken können, wie Sie es in Ihrem normalen Browser tun. Es unterstützt JavaScript und arbeitet mit AJAX-Bibliotheken. Es wird zum Testen und Abrufen von Daten von der Website verwendet.

Beispiel: Verwendung von HTMLUnitDriver zum Abrufen einer Fragenliste von `http://stackoverflow.com/` .

```
import java.util.List;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.htmlunit.HtmlUnitDriver;

class testHeadlessDriver{
            private void getQuestions() {
                    WebDriver driver = new HtmlUnitDriver();
                    driver.get("http://stackoverflow.com/");
                    driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);
                    List<WebElement> questions = driver.findElements(By.className("question-
hyperlink"));
                    questions.forEach((question) -> {
                        System.out.println(question.getText());
                    });
                   driver.close();
                }
    }
```

Dies ist mit jedem anderen Browser (Mozilla Firefox, Google Chrome, IE) identisch, es verfügt jedoch nicht über eine grafische Benutzeroberfläche.

Headless-Browser online lesen: https://riptutorial.com/de/selenium-webdriver/topic/3931/headless-browser

Kapitel 11: HTML-Berichte

Einführung

Dieses Thema behandelt die Erstellung von HTML-Berichten für Selentests. Es gibt verschiedene Arten von Plugins für die Berichterstellung. Allure, ExtentReports und ReportNG werden häufig verwendet.

Examples

ExtentReports

In diesem Beispiel wird die Implementierung von ExtentReports in Selenium mit TestNG, Java und Maven beschrieben.

ExtentReports sind in zwei Versionen verfügbar: Community und Commercial. Zur Vereinfachung und zu Demonstrationszwecken verwenden wir die Community-Version.

1. Abhängigkeit

Fügen Sie die Abhängigkeit in Ihrer Maven-Datei "pom.xml" für Bereichsberichte hinzu.

```
<dependency>
     <groupId>com.aventstack</groupId>
     <artifactId>extentreports</artifactId>
     <version>3.0.6</version>
 </dependency>
```

2. Plugins konfigurieren

Konfigurieren Sie das maven surefire Plugin wie folgt in pom.xml

```
 <build>
 <defaultGoal>clean test</defaultGoal>
 <plugins>
     <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-compiler-plugin</artifactId>
         <version>3.6.1</version>
         <configuration>
             <source>${jdk.level}</source>
             <target>${jdk.level}</target>
         </configuration>
     </plugin>
     <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-surefire-plugin</artifactId>
         <version>2.19.1</version>
         <configuration>
             <suiteXmlFiles>
```

```
                    <suiteXmlFile>testng.xml</suiteXmlFile>
                </suiteXmlFiles>
            </configuration>
        </plugin>
    </plugins>
</build>
```

3. Beispieltest mit ExtentReports

Erstellen Sie nun einen Test mit dem Namen test.java

```
public class TestBase {
    WebDriver driver;

    ExtentReports extent;
    ExtentTest logger;
    ExtentHtmlReporter htmlReporter;
    String htmlReportPath = "C:\\Screenshots/MyOwnReport.html"; //Path for the HTML report to
be saved

    @BeforeTest
    public void setup(){
        htmlReporter = new ExtentHtmlReporter(htmlReportPath);
        extent = new ExtentReports();
        extent.attachReporter(htmlReporter);

        System.setProperty("webdriver.chrome.driver", "pathto/chromedriver.exe");
        driver = new ChromeDriver();

    }

    @Test
    public void test1(){
        driver.get("http://www.google.com/");
        logger.log(Status.INFO, "Opened site google.com");
        assertEquals(driver.getTitle()), "Google");
        logger.log(Status.PASS, "Google site loaded");
    }

    @AfterMethod
    public void getResult(ITestResult result) throws Exception {
        if (result.getStatus() == ITestResult.FAILURE)
        {
            logger.log(Status.FAIL, MarkupHelper.createLabel(result.getName() + " Test case
FAILED due to below issues:", ExtentColor.RED));
            logger.fail(result.getThrowable());
        }
        else if (result.getStatus() == ITestResult.SUCCESS)
        {
            logger.log(Status.PASS, MarkupHelper.createLabel(result.getName() + " Test Case
PASSED", ExtentColor.GREEN));
        }
        else if (result.getStatus() == ITestResult.SKIP)
        {
            logger.log(Status.SKIP, MarkupHelper.createLabel(result.getName() + " Test Case
SKIPPED", ExtentColor.BLUE));
        }
     }

    @AfterTest
```

```
    public void testend() throws Exception {
        extent.flush();
    }

    @AfterClass
    public void tearDown() throws Exception {
        driver.close();
    }
```

Allure Reports

In diesem Beispiel wird die Implementierung von Allure Reports in Selenium mit TestNG, Java und Maven beschrieben.

Maven-Konfiguration

Repository

Fügen Sie folgenden Code hinzu, um das jcenter-Repository zu konfigurieren

```
<repository>
            <id>jcenter</id>
            <name>bintray</name>
            <url>http://jcenter.bintray.com</url>
</repository>
```

Abhängigkeit

Fügen Sie Ihrer pom.xml folgende Abhängigkeiten hinzu

```
<dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjweaver</artifactId>
        <version>${aspectj.version}</version>
    </dependency>
    <dependency>
        <groupId>ru.yandex.qatools.allure</groupId>
        <artifactId>allure-testng-adaptor</artifactId>
        <version>1.5.4</version>
    </dependency>
```

Surefire-Plugin-Konfiguration

```
<plugin>
        <groupId> org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.20</version>
        <configuration>
            <argLine>-
```

```
javaagent:${settings.localRepository}/org/aspectj/aspectjweaver/${aspectj.version}/aspectjweaver-
${aspectj.version}.jar
            </argLine>
            <properties>
                <property>
                    <name>listener</name>
                    <value>ru.yandex.qatools.allure.testng.AllureTestListener</value>
                </property>
            </properties>
            <suiteXmlFiles>testng.xml</suiteXmlFiles>
            <testFailureIgnore>false</testFailureIgnore>
        </configuration>
    </plugin>
```

Mustertest für Allure Report

Erstellen Sie einen Beispieltest mit dem Namen test.java

```
public class test{
    WebDriver driver;
    WebDriverWait wait;

    @BeforeMethod
    public void setup() {
        System.setProperty("webdriver.chrome.driver", "path to/chromedriver.exe");
        driver = new ChromeDriver();
        driver.get("https://www.google.com/");
        wait = new WebDriverWait(driver,50);
    }

    @Title("Title check")
    @Description("Checking the title of the loaded page.")
    @Test
    public void searchTest(){
        String title = driver.getTitle();
        LogUtil.log("Title Fetched: "+title);
        assertEquals(title,"Google");
        LogUtil.log("Test Passed. Expected: Google | Actual: "+title);
        System.out.println("Page Loaded");
    }

    @AfterMethod
    public void teardown(){
        driver.close();
    }
}
```

In der obigen Klasse haben wir die Klasse LogUtiil verwendet. Dies geschieht einfach, um die **Schritte** in unserem Test zu protokollieren. Unten ist der Code für das gleiche

LogUtil.java

```
public final class LogUtil {

    private LogUtil() {
    }
```

```
    @Step("{0}")
    public static void log(final String message){
        //intentionally empty
    }
}
```

Hier

@Title ("") fügt den Titel zu Ihrem Test in Allure Report hinzu

@Description ("") fügt die Beschreibung Ihrem Test hinzu

@Step ("") fügt für den Test einen Schritt im **Verlockungsbericht** hinzu

Bei der Ausführung wird eine XML-Datei im Ordner "target / allure-results /" erzeugt.

Abschlussbericht mit Jenkins

Wenn Sie Jenkins mit installiertem Allure Report-Plugin ausführen, rendert Jenkins den Bericht automatisch in Ihrem Job.

Abschlussbericht ohne Jenkins

Für diejenigen, die keine Jenkins haben, verwenden Sie die folgende Befehlszeile, um den HTML-Bericht zu erstellen. Allure CLI ist eine Java-Anwendung, die für alle Plattformen verfügbar ist. Sie müssen Java 1.7+ manuell installieren, bevor Sie Allure CLI verwenden.

Debian

Für Debian-basierte Repositorys stellen wir eine PPA bereit, sodass die Installation unkompliziert ist: Installieren Sie Allure CLI für Debian

```
$ sudo apt-add-repository ppa:yandex-qatools/allure-framework
$ sudo apt-get update
$ sudo apt-get install allure-commandline
```

Unterstützte Distributionen sind: vertrauenswürdig und präzise. Nach der Installation steht Ihnen der Allure-Befehl zur Verfügung.

Mac OS

Sie können Allure CLI über Homebrew installieren.

```
$ brew tap qatools/formulas
$ brew install allure-commandline
```

Nach der Installation steht Ihnen der Allure-Befehl zur Verfügung.

Windows und andere Unix

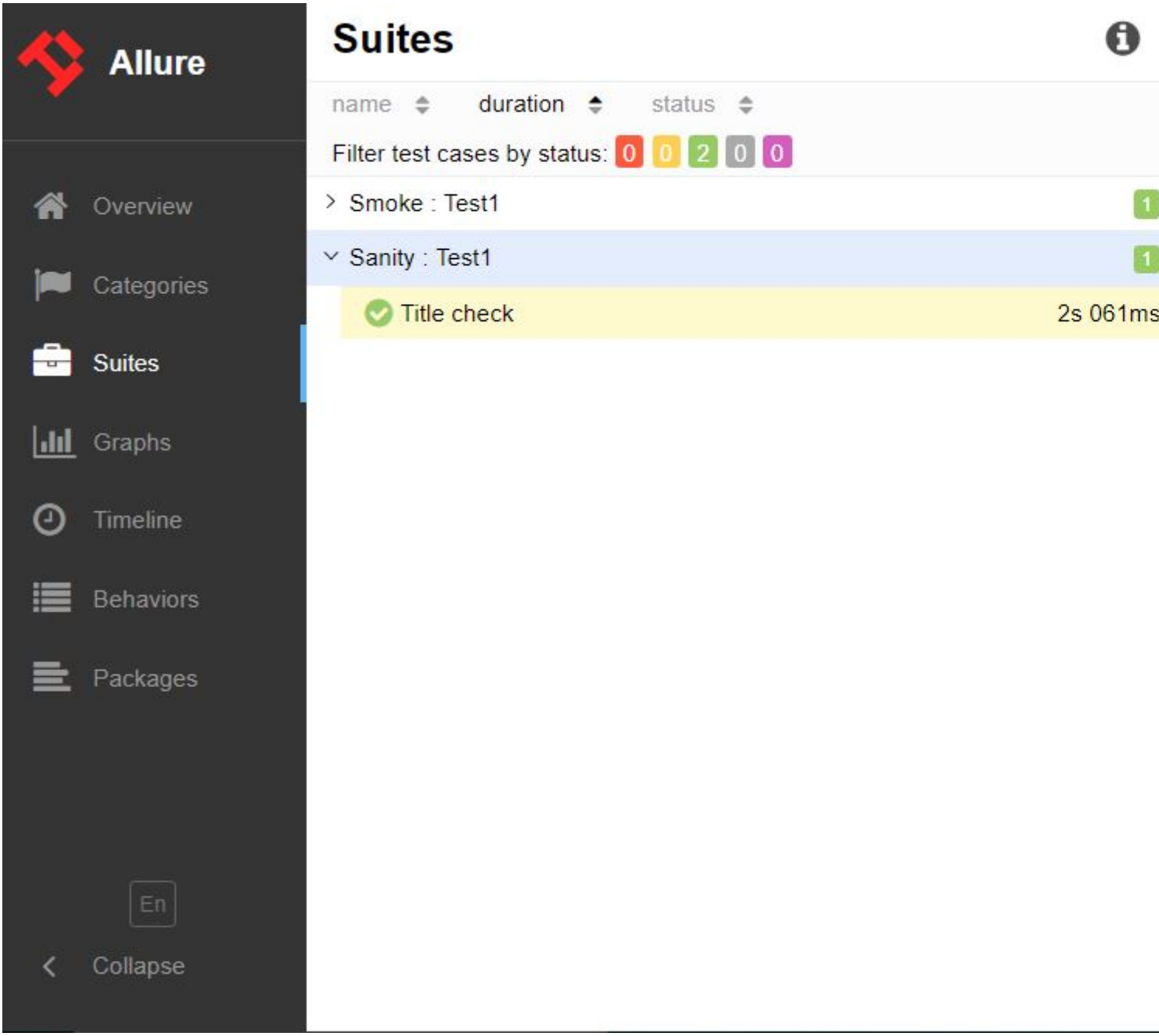
1. Laden Sie die neueste Version als ZIP-Archiv von https://github.com/allure-framework/allure-

core/releases/latest herunter.

2. Entpacken Sie das Archiv in das Allure-Commandline-Verzeichnis. Navigieren Sie zum bin-Verzeichnis.
3. Verwenden Sie allure.bat für Windows und allure für andere Unix-Plattformen.

Geben Sie in der Befehlszeile / im Terminal einfach die folgende Syntax ein und der Bericht wird im Allure-Report-Ordner generiert

```
$ allure generate directory-with-results/
```



HTML-Berichte online lesen: https://riptutorial.com/de/selenium-webdriver/topic/10721/html-berichte

Kapitel 12: Interaktion mit dem Webelement

Examples

C

Inhalt des Elements löschen (im Allgemeinen Textfeld)

```
interactionWebElement.Clear();
```

Daten in ein Element eingeben (im Allgemeinen Textfeld)

```
interactionWebElement.SendKeys("Text");
```

Speichern des Werts des Elements

```
string valueinTextBox = interactionWebElement.GetAttribute("value");
```

Text des Elements speichern

```
string textOfElement = interactionWebElement.Text;
```

Klicken Sie auf ein Element

```
interactionWebElement.Click();
```

Ein Formular absenden

```
interactionWebElement.Submit();
```

Identifizieren der Sichtbarkeit eines Elements auf der Seite

```
bool isDisplayed=interactionWebElement.Displayed;
```

Bestimmen des Status eines Elements auf der Seite

```
bool isEnabled = interactionWebElement.Enabled;

bool isSelected=interactionWebElement.Selected;
```

Untergeordnetes Element vonactionWebElement suchen

```
IWebElement childElement = interactionWebElement.FindElement(By.Id("childElementId"));
```

Suche nach untergeordneten Elementen vonactionWebElement

```
Ilist<IWebElement> childElements =
interactionWebElement.FindElements(By.TagName("childElementsTagName"));
```

Java

Löschen des Inhalts eines Webelements: (Hinweis: Wenn Sie Benutzeraktionen in Tests simulieren, sollten Sie die Rücktaste senden.

```
interactionWebElement.clear();
```

Eingeben von Daten - Simulieren von gesendeten Tastenanschlägen:

```
interactionWebElement.sendKeys("Text");
interactionWebElement.sendKeys(Keys.CONTROL + "c"); // copy to clipboard.
```

Den Wert des Attributs eines Elements abrufen:

```
interactionWebElement.getAttribute("value");
interactionWebElement.getAttribute("style");
```

Elementtext abrufen:

```
String elementsText = interactionWebElement.getText();
```

Auswahl aus dem Dropdown:

```
Select dropDown = new Select(webElement);
dropDown.selectByValue(value);
```

Selbsterklärend:

```
interactionWebElement.click();
interactionWebElement.submit(); //for forms
interactionWebElement.isDisplayed();
interactionWebElement.isEnabled(); // for exampale - is clickable.
interactionWebElement.isSelected(); // for radio buttons.
```

Aktionen mit `org.openqa.selenium.interactions.Actions` :

Ziehen und loslassen:

```
Action dragAndDrop = builder.clickAndHold(someElement)
   .moveToElement(otherElement)
   .release(otherElement)
   .build();

dragAndDrop.perform();
```

Wählen Sie mehrere aus:

```
Action selectMultiple = builder.keyDown(Keys.CONTROL)
   .click(someElement)
   .click(someOtherElement)
   .keyUp(Keys.CONTROL);

dragAndDrop.perform();
```

Selbsterklärend (mit Builder):

```
builder.doubleClick(webElement).perform();
builder.moveToElement(webElement).perform(); //hovering
```

Hier finden Sie weitere Beispiele für erweiterte Aktionen und eine vollständige Liste.

Javascript verwenden:

```
// Scroll to view element:
((JavascriptExecutor) driver).executeJavaScript("arguments[0].scrollIntoView(true);",
webElement);
```

Interaktion mit dem Webelement online lesen: https://riptutorial.com/de/selenium-webdriver/topic/4280/interaktion-mit-dem-webelement

Kapitel 13: Interaktion mit den Browserfenstern

Examples

Aktives Fenster verwalten

C

Maximierung des Fensters

```
driver.Manage().Window.Maximize();
```

Dies ist ziemlich einfach und stellt sicher, dass unser derzeit aktives Fenster maximiert wird.

Position des Fensters

```
driver.Manage().Window.Position = new System.Drawing.Point(1, 1);
```

Hier verschieben wir im Wesentlichen das gerade aktive Fenster an eine neue Position. Im `Point` Objekt stellen wir `x` und `y` -Koordinaten bereit. Diese werden dann als Versatz von der oberen linken Ecke des Bildschirms verwendet, um zu bestimmen, wo das Fenster platziert werden soll. Beachten Sie, dass Sie die Fensterposition auch in einer Variablen speichern können:

```
System.Drawing.Point windowPosition = driver.Manage().Window.Position;
```

Größe des Fensters

Das Festlegen und Abrufen der Fenstergröße verwendet dieselbe Syntax wie die Position:

```
driver.Manage().Window.Size = new System.Drawing.Size(100, 200);
System.Drawing.Size windowSize = driver.Manage().Window.Size;
```

URL des Fensters

Wir können die aktuelle URL des aktiven Fensters erhalten:

```
string url = driver.Url;
```

Wir können auch die URL für das aktive Fenster festlegen, wodurch der Treiber zum neuen Wert navigiert:

```
driver.Url = "http://stackoverflow.com/";
```

Fenstergriffe

Wir können das Handle für das aktuelle Fenster erhalten:

```
string handle = driver.CurrentWindowHandle;
```

Und wir können die Griffe für alle offenen Fenster erhalten:

```
IList<String> handles = driver.WindowHandles;
```

Python

Maximierung des Fensters

```
driver.maximize_window()
```

Holen Sie sich die Position des Fensters

```
driver.get_window_position() # returns {'y', 'x'} coordinates
```

Position des Fensters einstellen

```
driver.set_window_position(x, y) # pass 'x' and 'y' coordinates as arguments
```

Holen Sie sich die Größe des Fensters

```
driver.get_window_size() # returns {'width', 'height'} values
```

Größe des Fensters einstellen

```
driver.set_window_size(width, height) # pass 'width' and 'height' values as arguments
```

Aktueller Seitentitel

```
driver.title
```

Aktuelle URL

```
driver.current_url
```

Fenstergriffe

```
driver.current_window_handle
```

Liste der aktuell geöffneten Fenster

```
driver.window_handles
```

Das aktuelle Browserfenster schließen

Wechseln Sie zu der neu geöffneten Registerkarte. Schließen Sie die aktuellen Fenster (in diesem Fall die neue Registerkarte). Wechseln Sie wieder zum ersten Fenster.

WINKELMESSER:

```
browser.getAllWindowHandles().then(function (handles) {
    browser.driver.switchTo().window(handles[1]);
    browser.driver.close();
    browser.driver.switchTo().window(handles[0]);
});
```

JAVA-Selen:

```
        Set<String> handlesSet = driver.getWindowHandles();
        List<String> handlesList = new ArrayList<String>(handlesSet);
        driver.switchTo().window(handlesList.get(1));
        driver.close();
        driver.switchTo().window(handlesList.get(0));
```

Behandeln Sie mehrere Fenster

Python

Meistens verwendetes Szenario:

1. *Seite in neuem Fenster öffnen*
2. *wechseln Sie dazu*
3. *etwas tun*
4. *Schließe es*
5. *zurück zum übergeordneten Fenster wechseln*

```
# Open "Google" page in parent window
driver.get("https://google.com")

driver.title # 'Google'

# Get parent window
parent_window = driver.current_window_handle

# Open "Bing" page in child window
driver.execute_script("window.open('https://bing.com')")

# Get list of all windows currently opened (parent + child)
all_windows = driver.window_handles

# Get child window
child_window = [window for window in all_windows if window != parent_window][0]
```

```
# Switch to child window
driver.switch_to.window(child_window)

driver.title # 'Bing'

# Close child window
driver.close()

# Switch back to parent window
driver.switch_to.window(parent_window)

driver.title # 'Google'
```

Interaktion mit den Browserfenstern online lesen: https://riptutorial.com/de/selenium-webdriver/topic/5181/interaktion-mit-den-browserfenstern

Kapitel 14: Javascript in der Seite ausführen

Syntax

- object ExecuteAsyncScript (Zeichenfolgen-Skript, params object [] args);
- object ExecuteScript (Zeichenfolgen-Skript, params object [] args);

Examples

C

Um JavaScript in einer `IWebDriver` Instanz auszuführen, `IWebDriver` Sie den `IWebDriver` in eine neue Schnittstelle, `IJavaScriptExecutor` , `IJavaScriptExecutor`

```
IWebDriver driver;
IJavaScriptExecutor jsDriver = driver as IJavaScriptExecutor;
```

Sie können jetzt auf alle in der `IJavaScriptExecutor` Instanz verfügbaren Methoden `IJavaScriptExecutor` denen Sie Javascript ausführen können. Beispiel:

```
jsDriver.ExecuteScript("alert('running javascript');");
```

Python

Um Javascript in Python auszuführen, verwenden Sie `execute_script("javascript script here")` . execute_script wird in einer Web-Treiber-Instanz aufgerufen und kann ein beliebiges Javascript sein.

```
from selenium import webdriver
driver = webdriver.Chrome()
driver.execute_script("alert('running javascript');")
```

Java

Um Javascript in Java auszuführen, erstellen Sie einen neuen Web-Treiber, der Javascript unterstützt. Um die Funktion `executeScript()` können, muss entweder der Treiber in einen `JavascriptExecutor` werden oder eine neue Variable kann auf den Wert des gegossenen Treibers gesetzt werden: `((JavascriptExecutor)driver) driver.executeScript()` nimmt einen String mit gültigem Javascript auf.

```
WebDriver driver = new ChromeDriver();
JavascriptExecutor JavascriptExecutor = ((JavascriptExecutor)driver);
JavascriptExecutor.executeScript("alert('running javascript');");
```

Rubin

```
require "selenium-webdriver"

driver = Selenium::WebDriver.for :chrome
driver.execute_script("alert('running javascript');")
```

Javascript in der Seite ausführen online lesen: https://riptutorial.com/de/selenium-webdriver/topic/6986/javascript-in-der-seite-ausfuhren

Kapitel 15: Klasse auswählen

Syntax

- **Java**
- Alle abwählen()
- deselectByIndex (int index)
- deselectByValue (java.lang.String-Wert)
- deselectByVisibleText (java.lang.String)
- getAllSelectedOptions ()
- getFirstSelectedOption ()
- getOptions ()
- isMultiple ()
- selectByIndex (int index)
- selectByValue (java.lang.String-Wert)
- selectByVisibleText (java.lang.String-Text)

Parameter

Parameter	Einzelheiten
Index	Die Option an diesem Index wird ausgewählt
Wert	Der Wert, gegen den abgeglichen werden soll
Text	Der sichtbare Text, gegen den abgeglichen werden soll

Bemerkungen

`Select` Klasse von Selenium WebDriver bietet nützliche Methoden für die Interaktion mit `select` Optionen. Der Benutzer kann Vorgänge in einer Auswahl-Dropdown-Liste ausführen und auch die Auswahl mit den folgenden Methoden aufheben.

In **C # ist** die Select-Klasse eigentlich `SelectElement`

Examples

Verschiedene Möglichkeiten zur Auswahl aus der DropDown-Liste

Unten ist eine HTML-Seite

```
<html>
<head>
<title>Select Example by Index value</title>
```

```
</head>
<body>
<select name="Travel"><option value="0" selected> Please select</option>
<option value="1">Car</option>
<option value="2">Bike</option>
<option value="3">Cycle</option>
<option value="4">Walk</option>
</select>
</body>
</html>
```

JAVA

Wählen Sie Nach Index

So wählen Sie die Option über Index mit Java aus

```
public class selectByIndexExample {
    WebDriver driver;
    @Test
    public void selectSamples()
    {
        driver = new FirefoxDriver();
        driver.get("URL GOES HERE");
        WebElement element=driver.findElement(By.name("Travel")); //This is the 'Select'
element locator
        Select sel=new Select(element);
        sel.selectByIndex(1); //This will select the first 'Option' from 'Select' List i.e.
Car
    }
}
```

Nach Wert auswählen

```
public class selectByValueExample {
    WebDriver driver;
    @Test
    public void selectSamples()
    {
        driver = new FirefoxDriver();
        driver.get("URL GOES HERE");
        WebElement element=driver.findElement(By.name("Travel")); //This is the 'Select'
element locator
        Select sel=new Select(element);
        sel.selectByValue("Bike"); //This will select the 'Option' from 'Select' List which
has value as "Bike".
        //NOTE: This will be case sensitive
    }
}
```

Wählen Sie Nach Sichtbarkeitstext aus

```
public class selectByVisibilityTextExample {
    WebDriver driver;
    @Test
    public void selectSamples()
    {
        driver = new FirefoxDriver();
        driver.get("URL GOES HERE");
        WebElement element=driver.findElement(By.name("Travel")); //This is the 'Select'
element locator
        Select sel=new Select(element);
        sel.selectByVisibleText("Cycle"); //This will select the 'Option' from 'Select' List
who's visibility text is "Cycle".
        //NOTE: This will be case sensitive
    }
}
```

C

Alle folgenden Beispiele basieren auf der generischen `IWebDriver` Schnittstelle

Wählen Sie Nach Index

```
IWebElement element=driver.FindElement(By.name("Travel"));
SelectElement selectElement = new SelectElement(title);
selectElement.SelectByIndex(0);
```

Nach Wert auswählen

```
IWebElement element=driver.FindElement(By.name("Travel"));
SelectElement selectElement = new SelectElement(title);
selectElement.SelectByIndex("1");
//NOTE: This will be case sensitive
```

Nach Text auswählen

```
IWebElement element=driver.FindElement(By.name("Travel"));
SelectElement selectElement = new SelectElement(title);
selectElement.SelectByText("Walk");
```

Klasse auswählen online lesen: https://riptutorial.com/de/selenium-webdriver/topic/6426/klasse-auswahlen

Kapitel 16: Navigation

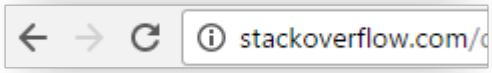
Syntax

- **C #**
- Zurück ()
- void Weiterleiten ()
- void GotToUrl (String-URL)
- Aktualisierung aufheben ()
- **Python**
- driver.back ()
- driver.forward ()
- driver.get ("URL")
- driver.refresh ()
- **Java**
- driver.navigate (). back ();
- driver.navigate (). forward ();
- driver.navigate (). to ("URL");
- driver.navigate (). refresh ();

Examples

Navigieren () [C #]

Es ist möglich, direkt im Browser zu navigieren, beispielsweise mit den standardmäßigen Symbolleistenbefehlen, die in allen Browsern verfügbar sind:



Sie können ein Navigationsobjekt erstellen, indem Sie im Treiber `Navigate()` aufrufen:

```
IWebDriver driver
INavigation navigation = driver.Navigate();
```

Mit einem Navigationsobjekt können Sie zahlreiche Aktionen ausführen, um den Browser im Web zu navigieren:

```
//like pressing the back button
navigation.Back();
//like pressing the forward button on a browser
navigation.Forward();
//navigate to a new url in the current window
navigation.GoToUrl("www.stackoverflow.com");
//Like pressing the reload button
navigation.Refresh();
```

Navigieren () [Java]

Navigieren Sie zu einer beliebigen URL:

```
driver.navigate().to("http://www.example.com");
```

Um rückwärts zu gehen:

```
driver.navigate().back();
```

Vorwärts bewegen:

```
driver.navigate().forward();
```

Um die Seite zu aktualisieren:

```
driver.navigate().refresh();
```

Browsermethoden in WebDriver

WebDriver, Die zum Testen zu verwendende Hauptschnittstelle, die einen idealisierten Webbrowser darstellt. Die Methoden in dieser Klasse lassen sich in drei Kategorien unterteilen:

- Kontrolle über den Browser selbst
- Auswahl von WebElements
- Hilfsmittel zur Fehlersuche

Schlüsselmethoden sind get (String), mit dem eine neue Webseite geladen wird, und die verschiedenen Methoden ähnlich wie findElement (By), mit denen WebElements gesucht werden. In diesem Beitrag werden wir Browser-Kontrollmethoden lernen. erhalten

```
void get(java.lang.String url)
```

Laden Sie eine neue Webseite in das aktuelle Browserfenster. Dies geschieht mithilfe einer HTTP-GET-Operation, und die Methode wird blockiert, bis der Ladevorgang abgeschlossen ist. Es ist am besten zu warten, bis diese Zeitüberschreitung abgelaufen ist, da sich die zugrunde liegende Seite ändert, während der Test die Ergebnisse zukünftiger Aufrufe für diese Schnittstelle ausführt, dies gegen die frisch geladene Seite. **Verwendungszweck**

```
//Initialising driver
 WebDriver driver = new FirefoxDriver();

 //setting timeout for page load
 driver.manage().timeouts().pageLoadTimeout(20, TimeUnit.SECONDS);

 //Call Url in get method
 driver.get("https://www.google.com");
 //or
 driver.get("https://seleniumhq.org");
```

getCurrentUrl

```
java.lang.String getCurrentUrl()
```

Rufen Sie eine Zeichenfolge ab, die die aktuelle URL darstellt, die der Browser anzeigt. Es gibt die URL der aktuell im Browser geladenen Seite zurück.

Verwendungszweck

```
//Getting current url loaded in browser & comparing with expected url
 String pageURL = driver.getCurrentUrl();
 Assert.assertEquals(pageURL, "https://www.google.com");
```

getTitle

```
java.lang.String getTitle()
```

Sie gibt den Titel der aktuellen Seite zurück, wobei führende und nachgestellte Leerzeichen entfernt werden, oder null, wenn noch keiner festgelegt wurde.

Verwendungszweck

```
//Getting current page title loaded in browser & comparing with expected title
 String pageTitle = driver.getTitle();
 Assert.assertEquals(pageTitle, "Google");

getPageSource

java.lang.String getPageSource()
```

Holen Sie sich die Quelle der zuletzt geladenen Seite. Wenn die Seite nach dem Laden geändert wurde (z. B. durch Javascript), kann nicht garantiert werden, dass der zurückgegebene Text der der geänderten Seite entspricht.

Verwendungszweck

```
//get the current page source
 String pageSource = driver.getPageSource();
```

schließen

```
void close()
```

Schließen Sie das aktuelle Fenster und beenden Sie den Browser, wenn es das letzte geöffnete Fenster ist. Wenn mit dieser Treiberinstanz mehr als ein Fenster geöffnet ist, schließt diese Methode das Fenster, das den aktuellen Fokus hat.

Verwendungszweck

```
//Close the current window
     driver.close();
```

Verlassen

```
void quit()
```

Beendet diesen Treiber und schließt jedes zugehörige Fenster. Nach dem Aufruf dieser Methode können wir keine andere Methode verwenden, die dieselbe Treiberinstanz verwendet.

Verwendungszweck

```
//Quit the current driver session / close all windows associated with driver
     driver.quit();
```

Dies sind alles sehr nützliche Methoden, die in Selenium 2.0 verfügbar sind, um den Browser nach Bedarf zu steuern.

Navigation online lesen: https://riptutorial.com/de/selenium-webdriver/topic/7272/navigation

Kapitel 17: Navigieren Sie zwischen mehreren Frames

Einführung

In Webseiten enthalten die Frame-Anzahl, Selen-Ansicht, Frame ist seperate Fenster, so dass auf den in Frame vorhandenen Inhalt zugegriffen werden kann, um in Frame zu wechseln. Oft benötigen wir eine Webstruktur, in der wir einen Frame mit Frame haben, um innerhalb von Frame-Fenstern zu navigieren. Die Selenium-Methode bietet die Methode swithTo ().

Examples

Rahmenbeispiel

```
<iframe "id="iframe_Login1">

    <iframe "id="iframe_Login2">

        <iframe "id="iframe_Login3">

        </iframe>

    </iframe>

</iframe>
```

Um in Selen in den Frame zu wechseln, verwenden Sie die Methode swithTo () und frame ().

driver.switchTo (). frame (iframe_Login1); driver.switchTo (). frame (iframe_Login2); driver.switchTo (). frame (iframe_Login3);

Um zurückzuschalten, können wir parentFrame () und defaultContest () verwenden.

parentFrame (): Ändert den Fokus auf den übergeordneten Kontext. Wenn der aktuelle Kontext der Browserkontext der obersten Ebene ist, bleibt der Kontext unverändert.

```
driver.switchTo().parentFrame();
```

defaultContent (): Wählt entweder den ersten Frame auf der Seite oder das Hauptdokument aus, wenn eine Seite iframes enthält.

```
 driver.switchTo().defaultContent();
```

Navigieren Sie zwischen mehreren Frames online lesen: https://riptutorial.com/de/selenium-webdriver/topic/9803/navigieren-sie-zwischen-mehreren-frames

Kapitel 18: Roboter in Selenium

Syntax

- Verzögerung (in ms)
- keyPress (int keycode)
- keyRelease (int keycode)
- mouseMove (int x, int y)
- mousePress (int-Tasten)
- mouseRelease (int-Tasten)
- mouseWheel (int wheelAmt)

Parameter

Parameter	Einzelheiten
Frau	Zeit zum Schlafen in Millisekunden
Schlüsselcode	Konstante, um die angegebene Taste zu drücken, um beispielsweise `A` Code zu drücken, ist `VK_A` . Weitere Informationen finden Sie hier: https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyEvent.html
x, y	Bildschirm koordiniert
Tasten	Die Button-Maske; eine Kombination aus einer oder mehreren Maustastenmasken
wheelAmt	Anzahl der Kerben, um das Mausrad zu bewegen, negativer Wert, um vom positiven Wert des Benutzers aufwärts / abwärts zu gehen, um den Benutzer nach unten zu bewegen

Bemerkungen

Dieser Abschnitt enthält Details zur Implementierung der Roboter-API mit dem Selenium Webdriver. Die Robot-Klasse wird verwendet, um native Systemeingaben zu generieren, wenn Selen dazu nicht in der Lage ist, z. B. durch Drücken der rechten Maustaste, F1-Taste usw.

Examples

Keypress-Ereignis mit der Roboter-API (JAVA)

```
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;
```

```
public class KeyBoardExample {
    public static void main(String[] args) {
        try {
            Robot robot = new Robot();
            robot.delay(3000);
            robot.keyPress(KeyEvent.VK_Q); //VK_Q for Q
        } catch (AWTException e) {
            e.printStackTrace();
        }
    }
}
```

Mit Selen

Manchmal müssen wir eine beliebige Taste drücken, um das Tastendruckereignis in der Webanwendung zu testen. Für eine Instanz zum Testen der EINGABETASTE im Anmeldeformular können Sie mit Selenium WebDriver Folgendes schreiben

```
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.Test;

public class LoginTest {

    @Test
    public void testEnterKey() throws InterruptedException
    {
        WebDriver driver=new FirefoxDriver();
        Robot robot=null;
        driver.get("test-url");
        driver.manage().window().maximize();
        driver.findElement(By.xpath("xpath-expression")).click();
        driver.findElement(By.xpath("xpath-expression")).sendKeys("username");
        driver.findElement(By.xpath("xpath-expression")).sendKeys("password");
        try {
            robot=new Robot();
        } catch (AWTException e) {
            e.printStackTrace();
        }
        //Keyboard Activity Using Robot Class
        robot.keyPress(KeyEvent.VK_ENTER);
    }
}
```

Mausereignis mit Roboter-API (JAVA)

Mausbewegung:

```
import java.awt.Robot;

public class MouseClass {
 public static void main(String[] args) throws Exception {
```

```
     Robot robot = new Robot();

     // SET THE MOUSE X Y POSITION
     robot.mouseMove(300, 550);
     }
}
```

Linke / rechte Maustaste drücken:

```
import java.awt.Robot;
import java.awt.event.InputEvent;

public class MouseEvent {
 public static void main(String[] args) throws Exception {
     Robot robot = new Robot();

     // LEFT CLICK
     robot.mousePress(InputEvent.BUTTON1_MASK);
     robot.mouseRelease(InputEvent.BUTTON1_MASK);

     // RIGHT CLICK
     robot.mousePress(InputEvent.BUTTON3_MASK);
     robot.mouseRelease(InputEvent.BUTTON3_MASK);
     }
}
```

Klicken und scrollen Sie das Rad:

```
import java.awt.Robot;
import java.awt.event.InputEvent;

public class MouseClass {
 public static void main(String[] args) throws Exception {
     Robot robot = new Robot();

     // MIDDLE WHEEL CLICK
     robot.mousePress(InputEvent.BUTTON3_DOWN_MASK);
     robot.mouseRelease(InputEvent.BUTTON3_DOWN_MASK);

     // SCROLL THE MOUSE WHEEL
     robot.mouseWheel(-100);
     }
}
```

Roboter in Selenium online lesen: https://riptutorial.com/de/selenium-webdriver/topic/4877/roboter-in-selenium

Kapitel 19: Screenshots aufnehmen

Einführung

Screenshots erstellen und in einem bestimmten Pfad speichern

Syntax

- Datei src = ((TakesScreenshot) Treiber) .getScreenshotAs (OutputType.FILE);
- FileUtils.copyFile (src, neue Datei ("D: \ screenshot.png"));

Examples

JAVA

Code zum Erstellen und Speichern eines Screenshots:

```
public class Sample
{
    public static void main (String[] args)
    {
        *//Initialize Browser*
        System.setProperty("webdriver.gecko.driver", "**E:\\path\\to\\geckodriver.exe**");
        WebDriver driver = new FirefoxDriver();
        driver.manage().window().maximize();
        driver.get("https://www.google.com/");

        //Take Screesnshot
        File src = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
        try {
            //Save Screenshot in destination file
            FileUtils.copyFile(src, new File("D:\\screenshot.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Nimmt Screenshot:

```
File src = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
```

Speichert den Screenshot von der Quelle bis zum Ziel:

```
FileUtils.copyFile(src, new File("D:\\screenshot.png"));
```

Screenshots aufnehmen online lesen: https://riptutorial.com/de/selenium-webdriver/topic/10094/screenshots-aufnehmen

Kapitel 20: Scrollen

Einführung

In diesem Thema werden verschiedene Ansätze für das Scrollen mit `selenium`

Examples

Scrollen mit Python

1. Scrollen Sie mit `Actions` Zielelement (Schaltfläche "BROWSE TEMPLATES" unten auf der Seite)

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains

driver = webdriver.Chrome()
driver.get('http://www.w3schools.com/')
target = driver.find_element_by_link_text('BROWSE TEMPLATES')
actions = ActionChains(driver)
actions.move_to_element(target)
actions.perform()
```

2. Scrollen Sie mit `JavaScript` Zielelement (Schaltfläche "BROWSE TEMPLATES" unten auf der Seite)

```
from selenium import webdriver

driver = webdriver.Chrome()
driver.get('http://www.w3schools.com/')
target = driver.find_element_by_link_text('BROWSE TEMPLATES')
driver.execute_script('arguments[0].scrollIntoView(true);', target)
```

3. Scrollen zum Zielelement (Schaltfläche "BROWSE TEMPLATES" am unteren Seitenrand) mit integrierter Methode

```
from selenium import webdriver

    driver = webdriver.Chrome()
    driver.get('http://www.w3schools.com/')
    target = driver.find_element_by_link_text('BROWSE TEMPLATES')
    target.location_once_scrolled_into_view
```

Beachten Sie, dass `location_once_scrolled_into_view` nach dem Scrollen auch `x` , `y` Koordinaten des Elements zurückgibt

4. Mit den `Keys` zum Ende der Seite `Keys`

```
from selenium import webdriver
```

```
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome()
driver.get('http://www.w3schools.com/')
driver.find_element_by_tag_name('body').send_keys(Keys.END) # Use send_keys(Keys.HOME) to
scroll up to the top of page
```

Beachten Sie, dass `send_keys(Keys.DOWN)` / `send_keys(Keys.UP)` und `send_keys(Keys.PAGE_DOWN)` / `send_keys(Keys.PAGE_UP)` auch zum Scrollen verwendet werden können

Unterschiedliches Scrollen mit Java mit verschiedenen Möglichkeiten

> Die folgende Lösung kann auch in anderen unterstützten Programmiersprachen mit einigen Syntaxänderungen verwendet werden

---

1. So scrollen Sie nach **unten** / Seite / Bereich / Abteilung auf der Webseite, während eine benutzerdefinierte Bildlaufleiste vorhanden ist (kein Browser-Bildlauf). Klicken Sie hier, um zu sehen, ob die Bildlaufleiste ein unabhängiges Element hat.

   Im folgenden Code übergeben Sie Ihr Bildlaufleistenelement und benötigen Bildlaufpunkte.

```
    public static boolean scroll_Page(WebElement webelement, int scrollPoints)
    {
    try
    {
        System.out.println("---------------- Started - scroll_Page ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        // drag downwards
        int numberOfPixelsToDragTheScrollbarDown = 10;
        for (int i = 10; i < scrollPoints; i = i + numberOfPixelsToDragTheScrollbarDown)
        {
            dragger.moveToElement(webelement).clickAndHold().moveByOffset(0,
numberOfPixelsToDragTheScrollbarDown).release(webelement).build().perform();
        }
        Thread.sleep(500);
        System.out.println("---------------- Ending - scroll_Page ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- scroll is unsucessfully done in scroll_Page -----
-----------");
        e.printStackTrace();
        return false;
    }
  }
```

---

2. So scrollen Sie nach **oben** / Seite / Bereich / Abteilung auf der Webseite, während eine benutzerdefinierte Bildlaufleiste vorhanden ist (kein Browser-Bildlauf). Klicken Sie hier, um zu sehen, ob die Bildlaufleiste ein unabhängiges Element hat.

Im folgenden Code übergeben Sie Ihr Bildlaufleistenelement und benötigen Bildlaufpunkte.

```
public static boolean scroll_Page_Up(WebElement webelement, int scrollPoints)
{
    try
    {
        System.out.println("---------------- Started - scroll_Page_Up ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);
        // drag upwards
        int numberOfPixelsToDragTheScrollbarUp = -10;
        for (int i = scrollPoints; i > 10; i = i + numberOfPixelsToDragTheScrollbarUp)
        {
            dragger.moveToElement(webelement).clickAndHold().moveByOffset(0,
numberOfPixelsToDragTheScrollbarUp).release(webelement).build().perform();
        }
        System.out.println("---------------- Ending - scroll_Page_Up ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- scroll is unsucessfully done in scroll_Page_Up---
-------------");
        e.printStackTrace();
        return false;
    }
}
```

---

3. Führen Sie einen Bildlauf nach unten durch, wenn Sie einen Bildlauf mit **mehreren Browsern durchführen** (In-Built-Browser) und Sie mit der **Bild-Ab-Taste** nach unten scrollen möchten. Klicken Sie hier für eine Demo

   Im folgenden Code übergeben Sie Ihr Bildlaufelement wie `<div>` und benötigen eine Abwärtspfeiltaste.

```
 public static boolean pageDown_New(WebElement webeScrollArea, int iLoopCount)
 {
    try
    {
        System.out.println("---------------- Started - pageDown_New ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        for (int i = 0; i <= iLoopCount; i++)
        {

dragger.moveToElement(webeScrollArea).click().sendKeys(Keys.PAGE_DOWN).build().perform();
        }
        System.out.println"---------------- Ending - pageDown_New ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- Not able to do page down ----------------");
        return false;
    }
```

```
}
```

4. Führen Sie einen Bildlauf nach oben durch, wenn Sie einen Bildlauf mit **mehreren Browsern durchführen** (In-Built-Browser) und Sie mit der Bild-Auf- **Taste** nach oben blättern möchten. Klicken Sie hier für eine Demo

   Im folgenden Code übergeben Sie Ihr Bildlaufelement wie `<div>` und benötigen einen Aufwärtspfeil.

```
public static boolean pageUp_New(WebElement webeScrollArea, int iLoopCount)
{
    try
    {
        System.out.println("---------------- Started - pageUp_New ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        for (int i = 0; i <= iLoopCount; i++)
        {

dragger.moveToElement(webeScrollArea).click().sendKeys(Keys.PAGE_UP).build().perform();
        }
        System.out.println("---------------- Ending - pageUp_New ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- Not able to do page up ----------------");
        return false;
    }
}
```

5. Führen Sie einen Bildlauf nach unten durch, wenn Sie einen Bildlauf mit **mehreren Browsern durchführen** (In-Built-Browser) und mit der **Pfeiltaste Nur Abwärts** nach unten scrollen möchten. Klicken Sie hier für eine Demo

   Im folgenden Code übergeben Sie Ihr Bildlaufelement wie `<div>` und benötigen die Abwärtstaste.

```
public static boolean scrollDown_Keys(WebElement webeScrollArea, int iLoopCount)
{
    try
    {
        System.out.println("---------------- Started - scrollDown_Keys ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        for (int i = 0; i <= iLoopCount; i++)
        {

dragger.moveToElement(webeScrollArea).click().sendKeys(Keys.DOWN).build().perform();
        }
        System.out.println("---------------- Ending - scrollDown_Keys ----------------");
        return true;
```

```
    }
    catch (Exception e)
    {
        System.out.println("---------------- Not able to do scroll down with keys-------------
---");
        return false;
    }
}
```

---

6. So scrollen Sie nach oben, wenn Sie mit **mehreren Browsern blättern** (In-Built-Browser) und mit **Nur Pfeil** nach oben blättern möchten. Klicken Sie hier für eine Demo

   Im folgenden Code übergeben Sie Ihr Bildlaufelement wie `<div>` und benötigen den Aufwärtsschlüssel.

```
public static boolean scrollUp_Keys(WebElement webeScrollArea, int iLoopCount)
{
    try
    {
        System.out.println("---------------- Started - scrollUp_Keys ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        for (int i = 0; i <= iLoopCount; i++)
        {
            dragger.moveToElement(webeScrollArea).click().sendKeys(Keys.UP).build().perform();
        }
        System.out.println("---------------- Ending - scrollUp_Keys ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- Not able to do scroll up with keys---------------
-");
        return false;
    }
}
```

---

7. Führen Sie einen Bildlauf nach oben / unten durch, wenn Sie im **Browser einen Bildlauf durchführen** (In-Built-Browser) und mit einem **festen Punkt** nach oben / unten blättern möchten. Klicken Sie hier für eine Demo

   Im folgenden Code übergeben Sie Ihren Bildlaufpunkt. Positiv bedeutet nach unten und negativ bedeutet nach oben.

```
public static boolean scroll_without_WebE(int scrollPoint)
{
    JavascriptExecutor jse;
    try
    {
        System.out.println("---------------- Started - scroll_without_WebE ----------------");

        driver = ExecutionSetup.getDriver();
        jse = (JavascriptExecutor) driver;
        jse.executeScript("window.scrollBy(0," + scrollPoint + ")", "");
```

```
        System.out.println("---------------- Ending - scroll_without_WebE ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- scroll is unsucessful in scroll_without_WebE ----
------------");
        e.printStackTrace();
        return false;
    }
}
```

---

8. Führen Sie einen Bildlauf nach oben / unten durch, wenn Sie im **Browser einen Bildlauf durchführen** (In-Built-Browser) und Sie nach oben / unten auf **Zum Element im sichtbaren Bereich oder dynamischen Bildlauf** blättern möchten. Klicken Sie hier für eine Demo

   Im folgenden Code übergeben Sie Ihr Element.

```
public static boolean scroll_to_WebE(WebElement webe)
{
    try
    {
        System.out.println("---------------- Started - scroll_to_WebE ----------------");

        driver = ExecutionSetup.getDriver();
        ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView();", webe);

        System.out.println("---------------- Ending - scroll_to_WebE ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- scroll is unsucessful in scroll_to_WebE ---------
-------");
        e.printStackTrace();
        return false;
    }
}
```

Hinweis: Bitte überprüfen Sie Ihren Fall und verwenden Sie Methoden. Wenn irgendein Fall fehlt, lass es mich wissen.

Scrollen online lesen: https://riptutorial.com/de/selenium-webdriver/topic/9063/scrollen

Kapitel 21: Seitenobjektmodell

Einführung

Eine wichtige Rolle bei der Automatisierung von Websites und Webanwendungen besteht darin, Elemente auf dem Bildschirm zu identifizieren und mit ihnen zu interagieren. Elemente werden in Selen mithilfe von Locators und der `By` Klasse gefunden. Diese Locators und Interaktionen werden als bewährte Methode in Page Objects eingefügt, um doppelten Code zu vermeiden und die Wartung zu vereinfachen. Es kapselt `WebElements` und nimmt an, Verhalten und `WebElements` auf der Seite (oder einem Teil einer Seite in einer Web-App) zu enthalten.

Bemerkungen

Das Seitenobjektmodell ist ein Muster, in dem wir objektorientierte Klassen schreiben, die als Schnittstelle zu einer bestimmten Ansicht einer Webseite dienen. Wir verwenden die Methoden dieser Seitenklasse, um die erforderliche Aktion auszuführen. Vor einigen Jahren manipulierten wir den HTML-Code der Webseite direkt in Testklassen, was sehr schwierig zu pflegen war und zusammen mit den Änderungen der Benutzeroberfläche sehr spröde war.

Wenn Sie Ihren Code jedoch so organisieren, dass das Seitenobjektmuster so angeordnet ist, steht Ihnen eine anwendungsspezifische API zur Verfügung, mit der Sie die Seitenelemente bearbeiten können, ohne sich im HTML-Bereich zu bewegen. Die grundlegende Daumenlehne besagt, dass Ihr Seitenobjekt alles enthalten sollte, was ein Mensch auf dieser Webseite tun kann. Um beispielsweise auf das Textfeld auf einer Webseite zuzugreifen, sollten Sie dort eine Methode verwenden, um den Text abzurufen und nach allen Änderungen den String zurückzugeben.

Einige wichtige Punkte, die Sie beim Entwerfen der Seitenobjekte beachten sollten:

1. Das Seitenobjekt sollte in der Regel nicht nur für Seiten erstellt werden, Sie sollten es jedoch lieber für bedeutende Elemente der Seite erstellen. Beispielsweise sollte eine Seite mit mehreren Registerkarten zum Anzeigen verschiedener Diagramme Ihrer Wissenschaftler die gleiche Anzahl von Seiten haben wie die Anzahl der Registerkarten.

2. Wenn Sie von einer Ansicht zur anderen navigieren, sollte die Instanz der Seitenklassen zurückgegeben werden.

3. Die Dienstprogrammmethoden, die nur für eine bestimmte Ansicht oder Webseite vorhanden sein müssen, sollten nur zu dieser Seitenklasse gehören.

4. Die Assertion-Methoden sollten nicht von Seitenklassen beachtet werden. Sie können Methoden haben, um boolean zurückzugeben, aber sie dort nicht zu überprüfen. Um beispielsweise den vollständigen Namen des Benutzers zu überprüfen, können Sie eine Methode zum Abrufen eines booleschen Werts verwenden:

```
public boolean hasDisplayedUserFullName (String userFullName) {
    return driver.findElement(By.xpath("xpathExpressionUsingFullName")).isDisplayed();
```

```
}
```

5. Wenn Ihre Webseite auf iframe basiert, sollten Sie auch Seitenklassen für iframes verwenden.

Vorteile des Page Object Pattern:

1. Saubere Trennung zwischen Testcode und Seitencode
2. Im Falle einer Änderung der Benutzeroberfläche der Webseite müssen Sie Ihren Code nicht an mehreren Stellen ändern. Ändern Sie nur in Seitenklassen.
3. Keine Streuelemente.
4. Macht den Code verständlicher
5. Einfache Wartung

Examples

Einführung (mit Java)

Ein Beispiel für den Anmeldetest basierend auf dem Seitenobjektmuster:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

/**
 * Class which models the view of Sign-In page
 */
public class SignInPage {

    @FindBy(id="username")
    private usernameInput;

    @FindBy(id="password")
    private passwordInput;

    @FindBy(id="signin")
    private signInButton;

    private WebDriver driver;

    public SignInPage(WebDriver driver) {
        this.driver = driver;
    }

    /**
     * Method to perform login
     */
    public HomePage performLogin(String username, String password) {
        usernameInput.sendKeys(username);
        passwordInput.sendKeys(password);
        signInButton.click();
        return PageFactory.initElements(driver, HomePage.class);
    }
```

```
}


import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
/**
 * Class which models the view of home page
 */
public class HomePage {
    @FindBy(id="logout")
    private logoutLink;

    private WebDriver driver;

    public HomePage(WebDriver driver) {
        this.driver = driver;
    }

    /**
     * Method to log out
     */
    public SignInPage logout() {
        logoutLink.click();
        wait.ForPageToLoad();
        return PageFactory.initElements(driver, SignInPage.class);
    }
}

/**
 * Login test class
 */
public class LoginTest {
    public void testLogin() {
        SignInPage signInPage = new SignInPage(driver);
        HomePage homePage = signInPage.login(username, password);
        signInPage = homePage.logout();
    }
}
```

C

Seitenobjekte sollten Verhalten enthalten, Informationen zu Zusicherungen zurückgeben und möglicherweise eine Methode für die Seitenbereitschaftsmethode bei der Initialisierung. Selen unterstützt Seitenobjekte mit Annotationen. In C # ist es wie folgt:

```
using OpenQA.Selenium;
using OpenQA.Selenium.Support.PageObjects;
using OpenQA.Selenium.Support.UI;
using System;
using System.Collections.Generic;

public class WikipediaHomePage
{
    private IWebDriver driver;
    private int timeout = 10;
    private By pageLoadedElement = By.ClassName("central-featured-logo");
```

```
    [FindsBy(How = How.Id, Using = "searchInput")]
    [CacheLookup]
    private IWebElement searchInput;

    [FindsBy(How = How.CssSelector, Using = ".pure-button.pure-button-primary-progressive")]
    [CacheLookup]
    private IWebElement searchButton;

    public ResultsPage Search(string query)
    {
        searchInput.SendKeys(query);
        searchButton.Click();
    }

    public WikipediaHomePage VerifyPageLoaded()
    {
        new WebDriverWait(driver, TimeSpan.FromSeconds(timeout)).Until<bool>((drv) =>  return
drv.ExpectedConditions.ElementExists(pageLoadedElement));

        return this;
    }
}
```

Anmerkungen:

- `CacheLookup` speichert das Element im Cache und speichert bei jedem Aufruf ein neues Element zurück. Dies verbessert die Leistung, eignet sich jedoch nicht für dynamisch wechselnde Elemente.
- `searchButton` hat 2 Klassennamen und keine ID. Aus diesem Grund kann ich weder Klassennamen noch IDs verwenden.
- Ich habe bestätigt, dass meine Locators das von mir gewünschte Element mit den Developer Tools (für Chrome) zurückgeben. In anderen Browsern können Sie FireBug oder ähnliches verwenden.
- `Search()` Methode gibt ein anderes `ResultsPage` (`ResultsPage`) zurück, wenn Sie mit dem Suchklick auf eine andere Seite umleiten.

Best Practices-Seite Objektmodell

- Erstellen Sie separate Dateien für Kopf- und Fußzeilen (da sie für alle Seiten üblich sind und es nicht sinnvoll ist, sie zu einem Teil einer Seite zu machen)
- Gemeinsame Elemente (wie Suchen / Zurück / Weiter usw.) in separaten Dateien aufbewahren (Du sollst jegliche Art von Duplizierungen entfernen und die Trennung logisch halten)
- Für Driver ist es eine gute Idee, eine separate Driver-Klasse zu erstellen und Driver als statisch zu halten, damit auf alle Seiten zugegriffen werden kann! (Ich habe alle meine Webseiten DriverClass erweitern)
- Die in PageObjects verwendeten Funktionen sind in kleinstmögliche Blöcke unterteilt, wobei die Häufigkeit und die Art und Weise, in der sie aufgerufen werden, berücksichtigt wird (die Art und Weise, wie sie für die Anmeldung verwendet werden. Die Anmeldung kann zwar in die Funktionen enterUsername und enterPassword unterteilt werden, sie behält sie jedoch bei da die Login-Funktion logischer ist, da in den meisten Fällen die Login-Funktion

aufgerufen wird, anstatt separate Aufrufe von enterUsername und enterPassword-Funktionen durchzuführen.

- Die Verwendung von PageObjects selbst trennt das Testskript von den elementLocators
- Nützliche Funktionen in separaten utils-Ordnern (wie DateUtil, ExcelUtils usw.)
- Konfigurationen in separaten conf-Ordnern haben (z. B. Festlegen der Umgebung, in der die Tests ausgeführt werden sollen, Konfigurieren der Ausgabe- und Eingabeordner)
- Bei Fehlschlagen screenCapture einbinden
- Haben Sie eine statische Wartevariable in der DriverClass mit einer impliziten Wartezeit wie Sie getan haben. Versuchen Sie immer, bedingte Wartezeiten zu haben, anstatt statische Wartezeiten wie: wait.until (ExpectedConditions). Dadurch wird sichergestellt, dass das Warten die Ausführung nicht unnötig verlangsamt.

Seitenobjektmodell online lesen: https://riptutorial.com/de/selenium-webdriver/topic/4853/seitenobjektmodell

Kapitel 22: Selen-Gitter

Examples

Knotenkonfiguration

Die Konfiguration des Selen-Grid-Knotens befindet sich auf dem Knoten selbst und enthält die Informationen zur Netzwerkkonfiguration und zu den Knotenfunktionen. Die Konfiguration kann auf verschiedene Arten angewendet werden:

- Standardkonfiguration
- JSON-Konfiguration
- Befehlszeile Konfiguration

JSON-Konfiguration

Die Knotenkonfiguration in der JSON-Datei ist in zwei Abschnitte unterteilt:

- Fähigkeiten
- Aufbau

Funktionen definieren Bereiche wie die unterstützten Browsertypen und -versionen, Speicherorte der Browser-Binärdateien und die Anzahl der maximalen Instanzen jedes Browsertyps.

Die Konfiguration befasst sich mit Einstellungen wie Hub- und Knotenadressen und Ports.

Nachfolgend finden Sie ein Beispiel für eine JSON-Konfigurationsdatei:

```
{
  "capabilities": [
    {
      "browserName": "firefox",
      "acceptSslCerts": true,
      "javascriptEnabled": true,
      "takesScreenshot": false,
      "firefox_profile": "",
      "browser-version": "27",
      "platform": "WINDOWS",
      "maxInstances": 5,
      "firefox_binary": "",
      "cleanSession": true
    },
    {
      "browserName": "chrome",
      "maxInstances": 5,
      "platform": "WINDOWS",
      "webdriver.chrome.driver": "C:/Program Files (x86)/Google/Chrome/Application/chrome.exe"
    },
    {
      "browserName": "internet explorer",
      "maxInstances": 1,
      "platform": "WINDOWS",
```

```
      "webdriver.ie.driver": "C:/Program Files (x86)/Internet Explorer/iexplore.exe"
    }
  ],
  "configuration": {
    "_comment" : "Configuration for Node",
    "cleanUpCycle": 2000,
    "timeout": 30000,
    "proxy": "org.openqa.grid.selenium.proxy.WebDriverRemoteProxy",
    "port": 5555,
    "host": ip,
    "register": true,
    "hubPort": 4444,
    "maxSessions": 5
  }
}
```

So erstellen Sie einen Knoten

Um einen Knoten zu erstellen, benötigen Sie zunächst einen Hub. Wenn Sie keinen Hub haben, können Sie ihn folgendermaßen erstellen:

```
java -jar selenium-server-standalone-<version>.jar -role hub
```

Dann können Sie einen Knoten erstellen:

```
java -jar selenium-server-standalone-<version>.jar -role node -hub
http://localhost:4444/grid/register // default port is 4444
```

Weitere Infos hier: https://github.com/SeleniumHQ/selenium/wiki/Grid2

Selen-Gitter online lesen: https://riptutorial.com/de/selenium-webdriver/topic/1359/selen-gitter

Kapitel 23: Selen-Gitterkonfiguration

Einführung

Selenium Grid ist ein Framework zum Ausführen von Tests, die über eine Reihe von Testgeräten verteilt werden. Es wird zum Testen von Webanwendungen verwendet. Es ist möglich, Tests in verschiedenen gängigen Programmiersprachen zu schreiben, einschließlich C #, Groovy, Java, Perl, PHP, Python und Ruby. Die Tests können mit einer Reihe von Webbrowsern auf Plattformen wie Windows, Linux und OS X ausgeführt werden.

Es handelt sich um Open-Source-Software, die unter der Apache 2.0-Lizenz veröffentlicht wird: Webentwickler können sie kostenlos herunterladen und verwenden.

Syntax

- Für die Ausführung der JAR-Datei gilt die folgende Syntax für jede JAR-Datei
- `java -jar <jar-file-full-name>.jar -<your parameters if any>`

Parameter

Parameter	Einzelheiten
Rolle	Ist das, was dem Selen mitteilt, welches `hub` oder `node` es war
Hafen	Hiermit legen Sie fest, welchen Port der `hub` oder `node` abhören soll.
Nabe	Dieser Parameter wird im `node` , um die Hub-URL anzugeben
browserName	Es wurde in `node` , um den Browsernamen wie Firefox, Chrome oder Internet Explorer anzugeben
maxInstanzen	Hier wird die Instanz des Browsers angegeben, z. 5 bedeutet, dass es 5 Instanzen des Browsers gibt, von denen der angegebene Benutzer vorhanden ist.
nodeConfig	Eine Json-Konfigurationsdatei für den Knoten. Hier können Sie die Rolle, den Port usw. angeben
hubConfig	Eine Json-Konfigurationsdatei für den Knoten. Hier können Sie die Rolle, den Port, die maximale Anzahl von Instanzen usw. angeben

Examples

Java-Code für Selenium Grid

```
String hubUrl = "http://localhost:4444/wd/hub"
DesiredCapabilities capability = DesiredCapabilities.firefox(); //or which browser you want
RemoteWebDriver driver = new RemoteWebDriver(hubUrl, capability);
```

Erstellen eines Selenium Grid-Hubs und -Knotens

Erstellen eines Hubs

Eine schnelle Konfiguration für ein Hub- und Knoten-Setup im Selen-Gitter. Weitere Informationen finden Sie unter: Raster 2 Dokumente

Bedarf

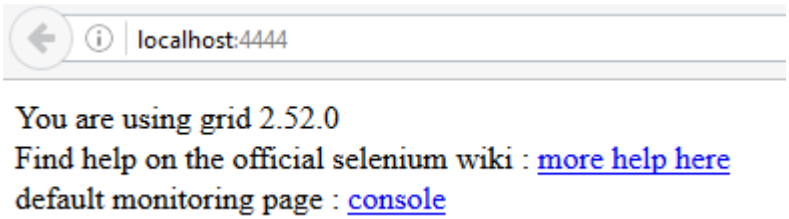
Um einen Grid-Hub einzurichten, benötigen Sie das Fließende:

- Selen-Server-Standalone-.jar

Hub erstellen

Um einen Hub zu erstellen, müssen Sie den Selenserver ausführen.

1. Laden Sie Selenium-server-standalone-.jar herunter
2. Öffnen Sie Ihr Terminal und navigieren Sie zu dem Ordner, in dem sich Selenium-server-standalone-.jar befindet
3. Führen Sie den folgenden Befehl aus:
    1. Für die Standardkonfiguration `java -jar selenium-server-standalone-<Version>.jar -role hub`
    2. Für die Json-Konfiguration `java -jar selenium-server-standalone-<Version>.jar -role hub -hubConfig hubConfig.json`
4. Öffnen Sie http: // localhost: 4444 / Sie werden eine folgende Nachricht sehen



Klicken Sie auf `console` -> `View config` for, um die Konfiguration für die Hub-Details anzuzeigen.

Einen Knoten erstellen

Bedarf

Um einen Grid-Hub einzurichten, benötigen Sie das Fließende:

- Selen-Server-Standalone-.jar
- Webtreiber
  - Chrome-Treiber
  - FireFox-Treiber
  - Microsoft Edge-Treiber
- Browser
  - Chrom
  - Feuerfuchs
  - Microsoft Edge (Windows 10)

Erstellen des Knotens

Jetzt Knoten für den Hub erstellen

1. Laden Sie Selenium-server-standalone-.jar herunter
2. Laden Sie die Browser herunter, in denen Sie testen möchten
3. Laden Sie die Treiber für die Browser herunter, in denen Sie testen möchten
4. Öffnen Sie ein neues Terminal und navigieren Sie zum Speicherort der JAR-Datei des Selen-Servers
5. Führen Sie den folgenden Befehl aus:
   1. für die Standardkonfiguration `java -jar selenium-server-standalone-<VERSION NUMBER>.jar -role node`
   2. Für die Json-Konfiguration `java -jar selenium-server-standalone-<Version>.jar -role node -nodeConfig nodeConfig.json`
6. Gehen Sie nun zu http: // localhost: 4444 / grid / console , um die Knotendetails anzuzeigen

Konfiguration über Json

Eine Beispielkonfiguration für einen Hub:

```
java -jar selenium-server-standalone-<version>.jar -role hub -hubConfig hubConfig.json
```

```
{
    "_comment" : "Configuration for Hub - hubConfig.json",
    "host": ip,
    "maxSessions": 5,
    "port": 4444,
    "cleanupCycle": 5000,
    "timeout": 300000,
    "newSessionWaitTimeout": -1,
    "servlets": [],
    "prioritizer": null,
    "capabilityMatcher": "org.openqa.grid.internal.utils.DefaultCapabilityMatcher",
    "throwOnCapabilityNotPresent": true,
    "nodePolling": 180000,
    "platform": "WINDOWS"
}
```

Eine Beispielkonfiguration für einen Knoten

```
java -jar selenium-server-standalone-<version>.jar -role node -nodeConfig nodeConfig.json
```

```
{
  "capabilities":
  [
    {
      "browserName": "opera",
      "platform": "WINDOWS",
      "maxInstances": 5,
      "seleniumProtocol": "WebDriver",
      "webdriver.opera.driver": "C:/Selenium/drivers/operadriver.exe",
      "binary":"C:/Program Files/Opera/44.0.2510.1159/opera.exe"
    },
    {
      "browserName": "chrome",
      "platform": "WINDOWS",
      "maxInstances": 5,
      "seleniumProtocol": "WebDriver",
      "webdriver.chrome.driver": "C:/Selenium/drivers/chromedriver.exe",
      "binary":"C:/Program Files/Google/Chrome/Application/chrome.exe"
    },
    {
      "browserName": "firefox",
      "platform": "WINDOWS",
      "maxInstances": 5,
      "seleniumProtocol": "WebDriver",
      "webdriver.gecko.driver": "C:/Selenium/drivers/geckodriver.exe",
      "binary":"C:/Program Files/Mozilla Firefox/firefox.exe"
    }
  ],
  "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",
  "maxSession": 5,
  "port": 5555,
  "register": true,
  "registerCycle": 5000,
  "hub": "http://localhost:4444",
  "nodeStatusCheckTimeout": 5000,
  "nodePolling": 5000,
  "role": "node",
  "unregisterIfStillDownAfter": 60000,
  "downPollingLimit": 2,
  "debug": false,
  "servlets" : [],
  "withoutServlets": [],
  "custom": {}
}
```

Selen-Gitterkonfiguration online lesen: https://riptutorial.com/de/selenium-webdriver/topic/2504/selen-gitterkonfiguration

Kapitel 24: Selenium e2e-Setup

Einführung

Dieses Thema behandelt die End-to-End-Einrichtung von Selenium, dh Selenium Webdriver + TestNG + Maven + Jenkins.

Informationen zum Hinzufügen von Berichten finden Sie im Thema HTML-Berichte

Examples

TestNG-Setup

TestNG ist Ihr aktualisiertes Testframework für junit. Wir werden **testng.xml verwenden** , um **Testsuiten** aufzurufen. Dies ist hilfreich, wenn wir CI vorab verwenden möchten.

testng.xml

Erstellen Sie im Stammordner Ihres Projekts eine XML-Datei mit dem Namen testng.xml. Beachten Sie, dass der Name auch anders sein kann, aber der Einfachheit halber wird er überall als "testng" verwendet.

Nachfolgend finden Sie den einfachen Code für die Datei testng.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Smoke"> //name of the suite
    <test name="Test1"> //name of the test
        <classes>
            <class name="test.SearchTest">
            <methods>
                <include name="searchTest"/>
            </methods>
            </class>
        </classes>
    </test>
</suite>
```

Maven-Setup

TBD. So richten Sie pom.xml für den Aufruf von testng.xml ein

Jenkins Setup

TBD. Behandelt das Jenkins-Setup für das Abrufen von Code von git / bitbucket usw.

Selenium e2e-Setup online lesen: https://riptutorial.com/de/selenium-

webdriver/topic/10724/selenium-e2e-setup

Kapitel 25: Selen-Webtreiber mit Python, Ruby und Javascript sowie CI-Tool

Einführung

Dies ist eine Möglichkeit, Selentests mit CircleCI durchzuführen

Examples

CircleCI-Integration mit Selenium Python und Unittest2

Circle.yml

```
machine:
  python:
    # Python version to use - Selenium requires python 3.0 and above
    version: pypy-3.6.0
dependencies:
    pre:
        # Install pip packages
        - pip install selenium
        - pip install unittest
test:
  override:
    # Bash command to run main.py
    - python main.py
```

main.py

```
import unittest2

# Load and run all tests in testsuite matching regex provided
loader = unittest2.TestLoader()
# Finds all the tests in the same directory that have a filename that ends in test.py
testcases = loader.discover('.', pattern="*test.py")
test_runner = unittest2.runner.TextTestRunner()
# Checks that all tests ran
success = test_runner.run(testcases).wasSuccessful()
```

example_test.py

```
class example_test(unittest.TestCase):
    def test_something(self):
        # Make a new webdriver instance
        self.driver = webdriver.Chrome()
        # Goes to www.goooogle.com
        self.driver.get("https://www.google.com")
```

Selen-Webtreiber mit Python, Ruby und Javascript sowie CI-Tool online lesen:

https://riptutorial.com/de/selenium-webdriver/topic/10091/selen-webtreiber-mit-python--ruby-und-javascript-sowie-ci-tool

Kapitel 26: Verwenden von @ FindBy-Annotationen in Java

Syntax

- CLASS_NAME: @FindBy (Klassenname = "Klassenname")
- CSS: @FindBy (css = "css")
- ID: @FindBy (id = "id")
- ID_OR_NAME: @FindBy (Wie = Wie.ID_OR_NAME, Verwendung von "idOrName")
- LINK_TEXT: @FindBy (linkText = "text")
- NAME: @FindBy (name = "name")
- PARTIAL_LINK_TEXT: @FindBy (partialLinkText = "text")
- TAG_NAME: @FindBy (tagName = "tagname")
- XPATH: @FindBy (xpath = "xpath")

Bemerkungen

Beachten Sie, dass es zwei Möglichkeiten gibt, die Anmerkung zu verwenden. Beispiele:

```
@FindBy(id = "id")
```

und

```
@FindBy(how = How.ID, using  ="id")
```

sind gleich und beide suchen nach Element anhand ihrer ID. Im Falle von `ID_OR_NAME` Sie nur verwenden

```
@FindBy(how = How.ID_OR_NAME, using ="idOrName")
```

`PageFactory.initElements()` muss nach der `@FindBy` Instantiierung verwendet werden, um Elemente zu finden, die mit der Annotation `@FindBy` markiert `@FindBy` .

Examples

Grundlegendes Beispiel

Nehmen wir an, wir verwenden das Page-Objektmodell . Page Objektklasse:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
```

```
public class LoginPage {

    @FindBy(id="loginInput") //method used to find WebElement, in that case Id
    WebElement loginTextbox;

    @FindBy(id="passwordInput")
    WebElement passwordTextBox;

    //xpath example:
    @FindBy(xpath="//form[@id='loginForm']/button(contains(., 'Login')")
    WebElement loginButton;

    public void login(String username, String password){
        // login method prepared according to Page Object Pattern
        loginTextbox.sendKeys(username);
        passwordTextBox.sendKeys(password);
        loginButton.click();
        // because WebElements were declared with @FindBy, we can use them without
        // driver.find() method
    }
}
```

Und test klasse:

```
class Tests{
    public static void main(String[] args) {
        WebDriver driver = new FirefoxDriver();
        LoginPage loginPage = new LoginPage();

        //PageFactory is used to find elements with @FindBy specified
        PageFactory.initElements(driver, loginPage);
        loginPage.login("user", "pass");
    }
}
```

Es gibt einige Methoden, um WebElements mit @FindBy zu finden.

Verwenden von @ FindBy-Annotationen in Java online lesen: https://riptutorial.com/de/selenium-webdriver/topic/6777/verwenden-von---findby-annotationen-in-java

Kapitel 27: Verwenden von Selenium Webdriver mit Java

Einführung

Selenium Webdriver ist ein Web-Automatisierungs-Framework, mit dem Sie Ihre Webanwendung mit verschiedenen Webbrowsern testen können. Im Gegensatz zu Selenium IDE können Sie mit webdriver eigene Testfälle in einer Programmiersprache Ihrer Wahl entwickeln. Es unterstützt Java, .NET, PHP, Python, Perl, Ruby.

Examples

Browserfenster mit spezifischer URL mit Selenium Webdriver in Java öffnen

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

class test_webdriver{
    public static void main(String[] args) {
       WebDriver driver = new FirefoxDriver();
       driver.get("http://stackoverflow.com/");
       driver.close();
    }
}
```

Öffnen eines Browserfensters mit der to () - Methode

Öffnen eines Browsers mit der to () - Methode.

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
class navigateWithTo{
    public static void main(String[] args) {
       WebDriver driver = new FirefoxDriver();
       driver.navigate().to("http://www.example.com");
       driver.close();
    }
}
```

Verwenden von Selenium Webdriver mit Java online lesen: https://riptutorial.com/de/selenium-webdriver/topic/9158/verwenden-von-selenium-webdriver-mit-java

Kapitel 28: Warten

Examples

Arten des Wartens in Selenium WebDriver

Beim Ausführen einer Webanwendung muss die Ladezeit berücksichtigt werden. Wenn Ihr Code versucht, auf ein Element zuzugreifen, das noch nicht geladen ist, gibt WebDriver eine Ausnahme aus und Ihr Skript wird angehalten.

Es gibt drei Arten von Wartezeiten -

- **Implizite Wartezeiten**
- **Explizite Wartezeiten**
- **Fließende Wartezeiten**

Implizite Wartezeiten werden verwendet, um die Wartezeit im gesamten Programm festzulegen, während explizite Wartezeiten nur für bestimmte Teile verwendet werden.

Implizite Wartezeit

Eine implizite Wartezeit ist, WebDriver anzuweisen, das DOM für eine bestimmte Zeit abzufragen, wenn versucht wird, ein Element oder Elemente zu finden, wenn diese nicht sofort verfügbar sind. Implizite Wartezeiten sind im Grunde Ihre Methode, um WebDriver die Latenzzeit mitzuteilen, die Sie sehen möchten, wenn das angegebene Webelement nicht vorhanden ist, nach dem WebDriver sucht. Die Standardeinstellung ist 0. Nach der Festlegung wird die implizite Wartezeit auf die Lebensdauer der WebDriver-Objektinstanz festgelegt. Implizites Warten wird im Instantiierungsteil des Codes mit dem folgenden Snippet deklariert.

Beispiel in **Java** :

```
driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
// You need to import the following class - import java.util.concurrent.TimeUnit;
```

Beispiel in **C #** :

```
driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(15));
```

In diesem Fall teilen Sie WebDriver mit, dass es 15 Sekunden warten sollte, wenn das angegebene Element nicht auf der Benutzeroberfläche (DOM) verfügbar ist.

Explizites Warten

Es kann vorkommen, dass ein Element mehr Zeit zum Laden benötigt. Das implizite Warten auf solche Fälle ist nicht sinnvoll, da der Browser für jedes Element die gleiche Zeit unnötig wartet und die Automatisierungszeit verlängert. Das explizite Warten hilft hier, indem es das implizite Warten auf bestimmte Elemente umgeht.

Explizite Wartezeiten sind intelligente Wartezeiten, die auf ein bestimmtes Webelement beschränkt sind. Mit expliziten Wartezeiten teilen Sie WebDriver im Wesentlichen so lange mit, wie lange es dauert, um X-Einheiten zu warten, bevor es aufgibt.

Explizite Wartezeiten werden mit den Klassen WebDriverWait und ExpectedConditions ausgeführt. Im folgenden Beispiel werden wir bis zu 10 Sekunden warten, bis ein Element, dessen ID Benutzername lautet, sichtbar werden, bevor wir mit dem nächsten Befehl fortfahren. Hier sind die Schritte.

Beispiel in **Java** :

```
//Import these two packages:
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

//Declare a WebDriverWait variable. In this example, we will use myWaitVar as the name of the
variable.
WebDriverWait myWaitVar = new WebDriverWait(driver, 30);

//Use myWaitVar with ExpectedConditions on portions where you need the explicit wait to occur.
In this case, we will use explicit wait on the username input before we type the text tutorial
onto it.
myWaitVar.until(ExpectedConditions.visibilityOfElementLocated(By.id(“username”)));
driver.findElement(By.id(“username”)).sendKeys(“tutorial”);
```

Die ExpectedConditions-Klasse hat einige vordefinierte allgemeine Bedingungen, um auf ein Element zu warten. Klicken Sie hier , um eine Liste dieser Bedingungen in der Java-Bindung anzuzeigen.

Beispiel in **C #** :

```
using OpenQA.Selenium;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.PhantomJS;

// You can use any other WebDriver you want, such as ChromeDriver.
using (var driver = new PhantomJSDriver())
{
    driver.Navigate().GoToUrl("http://somedomain/url_that_delays_loading");

    // We aren't going to use it more than once, so no need to declare this a variable.
    new WebDriverWait(driver, TimeSpan.FromSeconds(10))
        .Until(ExpectedConditions.ElementIsVisible(By.Id("element-id")));

    // After the element is detected by the previous Wait,
    // it will display the element's text
    Console.WriteLine(driver.FindElement(By.Id("element-id")).Text);
}
```

In diesem Beispiel wartet das System 10 Sekunden, bis das Element sichtbar ist. Wenn das Element nach dem Timeout nicht sichtbar ist, gibt der WebDriver eine `WebDriverTimeoutException` .

Bitte beachten Sie: Wenn das Element vor dem Timeout von 10 Sekunden sichtbar ist, fährt das System sofort mit der weiteren Verarbeitung fort.

Fließend warten

Im Gegensatz zum impliziten und expliziten Warten werden beim fließenden Warten zwei Parameter verwendet. Timeout-Wert und Abrufhäufigkeit. Nehmen wir an, wir haben einen Timeout-Wert von 30 Sekunden und eine Abfragefrequenz von 2 Sekunden. WebDriver sucht alle 2 Sekunden bis zum Timeout-Wert (30 Sekunden) nach einem Element. Nachdem der Timeout-Wert ohne Ergebnis überschritten wurde, wird eine Ausnahme ausgelöst. Nachfolgend finden Sie einen Beispielcode, der die Implementierung des fließenden Wartens veranschaulicht.

Beispiel in **Java** :

```
Wait wait = new FluentWait(driver).withTimeout(30, SECONDS).pollingEvery(2,
SECONDS).ignoring(NoSuchElementException.class);


WebElement testElement = wait.until(new Function() {
    public WebElement apply(WebDriver driver) {
        return driver.findElement(By.id("testId"));
    }
});
```

Ein weiterer Vorteil des fließenden Wartens ist, dass wir bestimmte Arten von Ausnahmen (z. B. NoSuchElementExceptions) während des Wartens ignorieren können. Aufgrund all dieser Vorkehrungen ist fließendes Warten in AJAX-Anwendungen sowie in Szenarien, in denen die Elementladezeit häufig schwankt, hilfreich. Der strategische Einsatz von fließendem Warten verbessert die Automatisierungsbemühungen erheblich.

Verschiedene Arten expliziter Wartebedingungen

Beim expliziten Warten erwarten Sie das Eintreten einer Bedingung. Sie möchten beispielsweise warten, bis ein Element anklickbar ist.

Hier sind einige häufige Probleme dargestellt.

Bitte beachten Sie: In all diesen Beispielen können Sie ein beliebiges `By` als Locator verwenden, beispielsweise `classname` , `xpath` , `link text` , `tag name` oder `cssSelector`

Warten Sie, bis das Element sichtbar ist

Wenn zum Beispiel das Laden Ihrer Website einige Zeit dauert, können Sie warten, bis die Seite

vollständig geladen ist und Ihr Element für den WebDriver sichtbar ist.

C #

```
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait.Until(ExpectedConditions.ElementIsVisible(By.Id("element-id")));
```

Java

```
WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("element-id")));
```

Warten Sie, bis das Element nicht mehr sichtbar ist

Wie zuvor, aber umgekehrt.

C #

```
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait.Until(ExpectedConditions.InvisibilityOfElementLocated(By.Id("element-id")));
```

Java

```
WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.invisibilityOfElementLocated(By.id("element-id")));
```

Warten Sie, bis im angegebenen Element Text vorhanden ist

C #

```
IWebElement element = driver.FindElement(By.Id("element-id"));

WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait.Until(ExpectedConditions.TextToBePresentInElement(element, "text"));
```

Java

```
WebElement element = driver.findElement(By.id("element-id"));

WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.textToBePresentInElement(element, "text"));
```

Wenn Sie zu dem angegebenen Link gehen, sehen Sie dort alle Wartebedingungen.

Der Unterschied zwischen der Verwendung dieser Wartebedingungen liegt in ihren Eingabeparametern.

Das bedeutet, dass Sie das WebElement übergeben müssen, wenn der Eingabeparameter WebElement lautet. Sie müssen den Element-Locator übergeben, wenn der By-Locator als Eingabeparameter verwendet wird.

Wählen Sie mit Bedacht, welche Art von Wartezeit Sie verwenden möchten.

Warten auf den Abschluss der Ajax-Anfragen

C

```
using OpenQA.Selenium
using OpenQA.Selenium.Chrome;
using System.Threading;

namespace WebDriver Tests
{
    class WebDriverWaits
    {
        static void Main()
        {
            IWebDriver driver = new ChromeDriver(@"C:\WebDriver");

            driver.Navigate().GoToUrl("page with ajax requests");
            CheckPageIsLoaded(driver);

            // Now the page is fully loaded, you can continue with further tests.
        }

        private void CheckPageIsLoaded(IWebDriver driver)
        {
            while (true)
            {
                bool ajaxIsComplete = (bool)(driver as
IJavaScriptExecutor).ExecuteScript("return jQuery.active == 0");
                if (ajaxIsComplete)
                    return;
                Thread.Sleep(100);
            }
        }
    }
}
```

Dieses Beispiel ist nützlich für Seiten, auf denen Ajax-Anforderungen gestellt werden. Hier verwenden wir den `IJavaScriptExecutor` , um unseren eigenen JavaScript-Code auszuführen. Da es sich innerhalb einer `while` Schleife befindet, wird es solange ausgeführt, bis `ajaxIsComplete == true` und die return-Anweisung ausgeführt wird.

Wir prüfen, ob alle Ajax-Anforderungen abgeschlossen sind, indem `jQuery.active` bestätigen, dass `jQuery.active` gleich `0` . Dies funktioniert, weil jedes Mal, wenn eine neue Ajax-Anforderung erfolgt, `jQuery.active` inkrementiert wird und jedes Mal, wenn eine Anforderung ergänzt wird, diese dekrementiert wird. Daraus können wir ableiten, dass bei `jQuery.active == 0` alle Ajax-Anforderungen abgeschlossen sein müssen.

Fließend warten

Fließendes Warten ist eine Superklasse des **expliziten** Wartens (`WebDriverWait`), die besser konfigurierbar ist, da sie ein Argument für die Wait-Funktion akzeptieren kann. Ich werde das **implizite** Warten weitergeben, da dies die beste Vorgehensweise ist , um es zu vermeiden.

Verwendung (Java):

```
Wait wait = new FluentWait<>(this.driver)
        .withTimeout(driverTimeoutSeconds, TimeUnit.SECONDS)
        .pollingEvery(500, TimeUnit.MILLISECONDS)
        .ignoring(StaleElementReferenceException.class)
        .ignoring(NoSuchElementException.class)
        .ignoring(ElementNotVisibleException.class);

WebElement foo = wait.until(ExpectedConditions.presenceOfElementLocated(By.yourBy));

// or use your own predicate:
WebElement foo = wait.until(new Function() {
  public WebElement apply(WebDriver driver) {
    return element.getText().length() > 0;
  }
});
```

Wenn Sie das explizite Warten mit seinen Standardwerten verwenden, ist es einfach ein `FluentWait<WebDriver>` mit den `DEFAULT_SLEEP_TIMEOUT = 500;` : `DEFAULT_SLEEP_TIMEOUT = 500;` und `NotFoundException` ignorieren.

Fließend warten

Jede FluentWait-Instanz definiert die maximale Wartezeit auf eine Bedingung sowie die Häufigkeit, mit der die Bedingung geprüft werden soll. Darüber hinaus kann der Benutzer das Warten so konfigurieren, dass bestimmte Arten von Ausnahmen während des Wartens ignoriert werden, z. B. NoSuchElementExceptions bei der Suche nach einem Element auf der Seite. Es ist dem Treiber zugeordnet.

```
Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)
.withTimeout(30, SECONDS) //actuall wait for the element to pe present
.pollingEvery(5, SECONDS) //selenium will keep looking for the element after every 5seconds
.ignoring(NoSuchElementException.class); //while ignoring this condition
wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("username"));
```

Warten online lesen: https://riptutorial.com/de/selenium-webdriver/topic/4435/warten

Kapitel 29: Zuhörer

Examples

JUnit

Wenn Sie JUnit zur Ausführung verwenden, können Sie die `TestWatcher` Klasse erweitern:

```
public class TestRules extends TestWatcher {

    @Override
    protected void failed(Throwable e, Description description) {
        // This will be called whenever a test fails.
    }
```

Sie können es also in Ihrer Testklasse einfach aufrufen:

```
public class testClass{

@Rule
public TestRules testRules = new TestRules();

@Test
public void doTestSomething() throws Exception{
    // If the test fails for any reason, it will be caught be testrules.
}
```

EventFiringWebDriver

EventFiringWebDriver verwenden . Sie können WebDriverEventListener daran anschließen und Methoden überschreiben, dh die onException-Methode

```
EventFiringWebDriver driver = new EventFiringWebDriver(new FirefoxDriver());
WebDriverEventListener listener = new AbstractWebDriverEventListener() {
    @Override
    public void onException(Throwable t, WebDriver driver) {
        // Take action
    }
};
driver.register(listener);
```

Zuhörer online lesen: https://riptutorial.com/de/selenium-webdriver/topic/8226/zuhorer

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit dem Selen-Webtreiber	Abhilash Gupta, Alice, Community, Eugene S, iamdanchiv, Jakub Lokša, Josh, Kishor, Michal, Mohit Tater, Pan, Priyanshu Shekhar, rg702, Santoshsarma, Tomislav Nakic-Alfirevic, vikingben
2	Aktionen (Emulieren komplexer Benutzergesten)	Josh, Kenil Fadia, Liam, Priyanshu Shekhar, Tom Mc
3	Auffinden von Webelementen	alecxe, daOnlyBG, Jakub Lokša, Josh, Liam, Łukasz Piaszczyk, Moshisho, NarendraR, noor, Priya, Sakshi Singla, Siva
4	Ausnahmen in Selenium-WebDriver	Brydenr
5	Basic Selenium Webdriver-Programm	Jakub Lokša, Josh, Liam, Priya, Sudha Velan, Thomas, vikingben
6	Behandeln Sie eine Warnung	Andersson, Aurasphere, Priya, SlightlyKosumi
7	Einstellen / Abrufen der Browserfenstergröße	Abhilash Gupta
8	Fehlerbehandlung bei der Automatisierung mit Selen	Andersson
9	Frames wechseln	Andersson, dreamwork801, Jakub Lokša, Java_deep, Jim Ashworth, Karthik Taduvai, Kyle Fairns, Liam, lloyd, Priyanshu Shekhar, SlightlyKosumi
10	Headless-Browser	Abhilash Gupta, Jakub Lokša, Liam, r_D, Tomislav Nakic-Alfirevic
11	HTML-Berichte	Ashish Deshmukh
12	Interaktion mit dem Webelement	Jakub Lokša, Liam, Moshisho, Siva, Sudha Velan

13	Interaktion mit den Browserfenstern	Andersson, Josh, Sakshi Singla
14	Javascript in der Seite ausführen	Brydenr, Liam
15	Klasse auswählen	Gaurav Lad, Liam
16	Navigation	Andersson, Liam, Santoshsarma, viralpatel
17	Navigieren Sie zwischen mehreren Frames	Pavan T, Raghvendra
18	Roboter in Selenium	Priyanshu Shekhar
19	Screenshots aufnehmen	Abhilash Gupta, Sanchit
20	Scrollen	Andersson, Sagar007
21	Seitenobjektmodell	JeffC, Josh, Moshisho, Priyanshu Shekhar, Sakshi Singla
22	Selen-Gitter	Eugene S, Y-B Cause
23	Selen-Gitterkonfiguration	mnoronha, Prasanna Selvaraj, selva, Thomas
24	Selenium e2e-Setup	Ashish Deshmukh
25	Selen-Webtreiber mit Python, Ruby und Javascript sowie CI-Tool	Brydenr
26	Verwenden von @ FindBy-Annotationen in Java	Alex Wittig, Łukasz Piaszczyk
27	Verwenden von Selenium Webdriver mit Java	r_D, the_coder
28	Warten	Jakub Lokša, Josh, Kenil Fadia, Liam, Moshisho, noor, Sajal Singh, Saurav
29	Zuhörer	Erki M.