Automated Testing in Java and Node JS with Selenium and ExtJS 6.5

schmittjoaopedro ExtJS, HTML, Java, Java Script, Node JS, Tests 29 29+00:00 Julho 29+00:00 2017 12 Minutes

Introduction

In software testing, an automated test is the use of special software to control the execution of tests and compare current results with expected results. Test automation serves to perform repetitive but necessary tasks in a formalized test process, and / or perform some additional activity that can be difficult to execute by a manual process. Automated testing is critical to continuous delivery and continuous testing processes.

Selenium

The Selenium (http://www.seleniumhq.org/) tool perform the automation of browsers. It serves to automate WEB applications for the purpose of testing, but not limited to this, it can also automate repetitive and boring activities. The Selenium tool is supported by the main browsers, and vendors make the effort to keep the tool drivers as a native part of their browsers. In addition, it serves as an intrinsic part of several APIs and Frameworks for browser automation.

Ext JS

Sencha Ext JS is a Java Script framework used to build web applications with intense data usage and cross-platform support. The framework is designed for applications running on desktops, tablets and smartphones. Based on componentization, the framework focuses on the use of Java Script for manipulation of the DOM through the object-oriented paradigm.

Problem

How to perform automated tests of WEB interfaces generated by the ExtJS Framework? The use of Selenium is often trivial, when using fixed structures with elements that have id's or that are very separable by attributes or classes, and in addition, when the generation of html is under the control of the developers. In this scenario the manipulation of the tests can even be performed by the Selenium IDE, a Firefox extension that allows you to "record" the sequence of clicks and interactions with the screen. In the case of Ext JS, this type of tool does not apply because most of the components are created in runtime and this adds more complexity in the flow of the screen construction and in the use of XPaths. Thus, a practical example of how an ExtJS application can be tested using Java code or Java Script using Selenium is shown below.

Requirements

The following items were used to produce this article:

- IntelliJ
- Java 8
- Maven
- Node JS
- Chrome
- Firefox

The source code in Java is available in the Selenium repository with Java (https://github.com/schmittjoaopedro/selenium-extjs-6.5) and in Java Script in the Selenium repository with Java Script (https://github.com/schmittjoaopedro/selenium-extjs-6.5-nodejs). The tests were performed on the pre-made components at the following link: ExtJS Kitchen Sink (http://examples.sencha.com/extjs/6.5.0/examples/kitchensink/#all).

Java Project (Link (https://github.com/schmittjoaopedro/selenium-extjs-6.5))

Starting with the Java project, the following dependencies are required:

- selenium-java
- selenium-chrome-driver
- selenium-htmlunit-driver
- selenium-firefox-driver
- selenium-server

The *webdriver* folder has the necessary drivers to perform the tests, the webdrivers provides an API for interaction with the native functions of browsers.

The code below shows the class responsible for starting the tests in Chrome, this class instantiates the driver and invokes the test methods. The *dragAndDropTest()* method is disabled because in ExtJS unidentified limitations in Chrome prevent it from working, however the Drag and Drop tests normally work with Firefox.

```java
package joao.schmitt;

import joao.schmitt.ext.DragAndDropTest;
import joao.schmitt.ext.EditorGridTest;
import joao.schmitt.ext.FormRegisterTest;
import org.junit.Test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class ChromeTests {

    @Test
    public void formRegisterTest() {
        WebDriver driver = getDriver();
        FormRegisterTest formRegisterTest = new FormRegisterTest(driver);
        formRegisterTest.run();
        driver.quit();
    }

    //@Test
    public void dragAndDropTest() {
        //Chrome incompatibility https://bugs.chromium.org/p/chromedriver
        WebDriver driver = getDriver();
        DragAndDropTest formRegisterTest = new DragAndDropTest(driver);
        formRegisterTest.run();
        driver.quit();
    }

    @Test
    public void editorGridTest() {
        WebDriver driver = getDriver();
        EditorGridTest editorGridTest = new EditorGridTest(driver);
        editorGridTest.run();
        driver.quit();
    }

    private WebDriver getDriver() {
        System.setProperty(
            "webdriver.chrome.driver",
            "webdriver/chromedriver");

        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        return driver;
    }

}
```

The classes invoked DragAndDropTest (https://github.com/schmittjoaopedro/selenium-extjs-6.5/blob/master/src/test/java/joao/schmitt/ext/DragAndDropTest.java), EditorGridTest (https://github.com/schmittjoaopedro/selenium-extjs-6.5/blob/master/src/test/java/joao/schmitt/ext/EditorGridTest.java) and FormRegisterTest (https://github.com/schmittjoaopedro/selenium-extjs-6.5/blob/master/src/test/java/joao/schmitt/ext/FormRegisterTest.java) runs the following test scenarios:

- DragAndDropTest: Drag the “app” component into the “Custom Ext JS” component.

- EditorGridTest: 1) Changes the values of the cells of the line that has the initial value Adder's-Tongue, 2) Removes the Anemone record and 3) Inserts a new record.
- FormRegisterTest: 1) Validate if all fields are initially empty, 2) Validate placeholder texts, 3) Validate required fields and 4) Fill the field values and click in Register button.

The most relevant methods used in the tests will be explained below. Starting with the FormRegister class:

```
//Wait for an element be visible
private void waitForElement(String name) {
    new WebDriverWait(this.webDriver, 600).until(ExpectedConditions.visib
}

//Find an element by label text and check the input name
private void validEmptyElementByLabel(String label, String name) {
    WebElement element = this.webDriver.findElement(By.xpath("//div[label
    Assert.assertEquals(name, element.getAttribute("name"));
}

//Find an element by label text and check the input placeholder
private void validPlaceHolderValues(String label, String value) {
    WebElement element = this.webDriver.findElement(By.xpath("//div[label
    Assert.assertEquals(value, element.getAttribute("placeholder"));
}

//Make onBlur at firstField to valid the alert required message
private void validRequiredFields(String firstLabel, String nextLabel, boo
    this.webDriver.findElement(By.xpath("//div[label/span/span[text()='"
    this.webDriver.findElement(By.xpath("//div[label/span/span[text()='"
    try {
        WebElement element = this.webDriver.findElement(By.xpath("//div[l
    } catch (NoSuchElementException el) {
        Assert.assertTrue(!required);
        return;
    }
    Assert.assertTrue(required);
}

//Input text in the field
private void sendFieldValues(String label, String value) {
    this.webDriver.findElement(By.xpath("//div[label/span/span[text()='"
}

//Set a value in the combobox
private void setComboValue(String label, String value, int size) {
    //Find combobox element
    WebElement comboElement = this.webDriver.findElement(By.xpath("//div[
    String pickerId = comboElement.getAttribute("id") + "-trigger-picker"
    //Click at arrow picker
    this.webDriver.findElement(By.xpath("//div[@id='" + pickerId + "']"))
    String componentId = comboElement.getAttribute("id") + "-picker";
    //Wait for the options
    new WebDriverWait(this.webDriver, 10).until(ExpectedConditions.visibi
    List&amp;amp;amp;amp;lt;WebElement&amp;amp;amp;amp;gt; comboOptions =
    //Check the options list size
    Assert.assertEquals(size, comboOptions.size());
    //Select an option
    this.webDriver.findElement(By.xpath("//li[text()='"+ value +"']")).cl
}
```

```java

//Select the day one of the current month
private void setDateValueOne(String label) {
    //Find combobox element
    WebElement comboElement = this.webDriver.findElement(By.xpath("//div[
    String componentId = comboElement.getAttribute("id") + "-trigger-pick
    //Click at arrow picker
    this.webDriver.findElement(By.xpath("//div[@id='" + componentId + "']
    String xpathDayOne = "//td[@class='x-datepicker-active x-datepicker-c
    //Select the first valid day of the month
    new WebDriverWait(this.webDriver, 10).until(ExpectedConditions.visibi
    this.webDriver.findElement(By.xpath(xpathDayOne)).click();
}

//Check if the button is clickable
private void validButtonState(String label, boolean state) {
    Assert.assertEquals(state, this.webDriver.findElement(By.xpath("//a[s
}

//Button click
private void clickInButton(String label) {
    this.webDriver.findElement(By.xpath("//a[span/span/span[text()='" + l
}
```

And now the methods of the EditorGrid class:

```java
//Wait for a div with an specific text
private void waitForDivWithText(String text) {
    new WebDriverWait(this.webDriver, 600).until(ExpectedConditions.visib
}

//Set a value for a cell with textfield editor
private void setCellValueByLabel(String text, int column, String value) {
    //Find a row cell that the first column contains the value argument
    WebElement cell = this.webDriver.findElement(By.xpath("//tr[td/div[te
    //Start edit
    cell.click();
    //Wait for input element
    new WebDriverWait(this.webDriver, 2).until(ExpectedConditions.visibil
    //Clear the value
    cell.findElement(By.xpath("//input")).clear();
    //Send the new value
    cell.findElement(By.xpath("//input")).sendKeys(value);
    //Finish with Enter
    cell.findElement(By.xpath("//input")).sendKeys(Keys.ENTER);
}

//Set a value for a cell with combobox editor
private void setCellComboValueByLabel(String label, int column, String se
    //Find a row cell that the first column contains the value argument
    WebElement cell = this.webDriver.findElement(By.xpath("//tr[td/div[te
    //Start edit
    cell.click();
    //Wait for input element
    new WebDriverWait(this.webDriver, 2).until(ExpectedConditions.visibil
    //Send the search text
    cell.findElement(By.xpath("//input")).sendKeys(search);
    //Wait for options
    new WebDriverWait(this.webDriver, 2).until(ExpectedConditions.visibil
    //Select an option
    cell.findElement(By.xpath("//li[text()=\"" + value + "\"]")).click();
    //Finish with enter
    cell.findElement(By.xpath("//input")).sendKeys(Keys.ENTER);
}

//Set a value for a cell with numberfield editor
private void setCellSpinnerValueByLabel(String label, int column, int num
    //Find a row cell that the first column contains the value argument
    WebElement cell = this.webDriver.findElement(By.xpath("//tr[td/div[te
    //Start edit
    cell.click();
    //Wait for input element
    new WebDriverWait(this.webDriver, 2).until(ExpectedConditions.visibil
    //Key up a number of times
    for(int i = 0; i < numberClicks; i++) {
        cell.findElement(By.xpath("//input")).sendKeys(Keys.ARROW_UP);
    }
    //Finish with enter
    cell.findElement(By.xpath("//input")).sendKeys(Keys.ENTER);
}

//Click in a specific cell
private void setClickByLabel(String label, int column) {
    this.webDriver.findElement(By.xpath("//tr[td/div[text()=\"" + label +
}

//Scroll to the first record of the grid
private void scrollToTop(String label) {
    WebElement cell = this.webDriver.findElement(By.xpath("//tr[td/div[te
    ((JavascriptExecutor) this.webDriver).executeScript("arguments[0].scr
    try {
        Thread.sleep(500);
    } catch (Exception e) {
        Assert.assertTrue(false);
    }
}
```

Above, only the utilitarian methods created to execute the tests have been presented; the rest of the code is available in the GIT repository; it can be seen that the tests in ExtJS require a good investigation to understand the structure of the elements that makes an intense use of div's and span's. Due to the dynamics in the creation of the elements of the DOM the id's can change during the use of the screen, so a navigation with xpath using texts and attributes of the elements of the DOM is necessary. A problem detected was that Drag and Drop did not work in the Chrome browser and a solution to the problem was not found, but because the methods worked in Firefox it can be assumed that the problem is in the implementation of the driver.

NodeJS Project (Link (https://github.com/schmittjoaopedro/selenium-extjs-6.5-nodejs))

The following is a JavaScript implementation using NodeJS performing the same tests. The following dependencies were used:

- selenium-webdriver
- chromedriver
- geckodriver
- mocha
- chai

Due to the asynchronous characteristic of the selenium tests in NodeJS, all steps of each test are chained through callbacks with Promisse. From a practical point of view, this type of code adds more complexity compared to Java methods. For example, the following shows the main methods for the FormRegisterTest class.

```
//Find an element by label text and check the input name
validEmptyElementByLabel(driver, label, name) {
    return driver.findElement(By.xpath("//div[label/span/span[text()='Use
        .then((element) => element.getAttribute('name'))
        .then((text) => assert.equal(text, 'user'))
}

//Check if the button is clickable
validButtonState(driver, label, state) {
    return driver.findElement(By.xpath(`//a[span/span/span[text()='${labe
       .then((element) => element.getAttribute('class'))
       .then((text) => assert.equal(state, text.indexOf("x-item-disabled"
}

//Find an element by label text and check the input placeholder
validPlaceHolderValues(driver, label, value) {
    return () => driver.findElement(By.xpath(`//div[label/span/span[text(
        .then((element) => element.getAttribute('placeholder'))
        .then((text) => assert.equal(text, value));
}

//Make onBlur at firstField to valid the alert required message
validRequiredFields(driver, firstLabel, nextLabel, required) {
    return driver.findElement(By.xpath(`//div[label/span/span[text()='${f
        .then((element => element.click()))
        .then(() => driver.findElement(By.xpath(`//div[label/span/span[te
        .then((element => element.click()))
        .then(() => driver.findElements(By.xpath(`//div[label/span/span[t
        .then((elements) => assert.equal(required, elements.length === 1)
}

//Input text in the field
sendFieldValues(driver, label, value) {
    return driver.findElement(By.xpath(`//div[label/span/span[text()='${l
        .then((element) => element.sendKeys(value));
}

//Set a value in the combobox
sendComboValue(driver, label, value, size) {
    var state = {};
    return driver.findElement(By.xpath(`//div[label/span/span[text()='${l
        .then((element) => element.getAttribute('id'))
        .then((text) => {
            state = {
                pickerId: `${text}-trigger-picker`,
                componentId: `${text}-picker`
            }
        })
```

```
            .then(() => driver.findElement(By.xpath("//div[@id='" + state.pic
            .then(() => driver.wait(until.elementLocated(By.xpath(`//li[text(
            .then(() => driver.findElements(By.xpath(`//li[@data-boundview='$
            .then((element) => assert.equal(size, element.length))
            .then(() => driver.findElement(By.xpath(`//li[text()='${value}']`
            .then((element) => element.click())
    }

    //Select the day one of the current month
    sendDateValueOne(driver, label) {
        return driver.findElement(By.xpath(`//div[label/span/span[text()='${l
            .then((element) => element.getAttribute('id'))
            .then((text) => `${text}-trigger-picker`)
            .then((text) => driver.findElement(By.xpath(`//div[@id='${text}']
            .then((element) => element.click())
            .then(() => driver.wait(until.elementLocated(By.xpath(`//td[@clas
            .then(() => driver.findElement(By.xpath(`//td[@class='x-datepicke
            .then((element) => element.click());
    }

    //Button click
    clickInButton(driver, label) {
        return driver.findElement(By.xpath(`//a[span/span/span[text()='${labe
           .then((element) => element.click());
    }
```

And now the methods of the EditorGridTest class:

```
//Wait for a div with an specific text
waitForDivWithText(driver, text) {
    return driver.wait(until.elementLocated(By.xpath(`//div[text()="${tex
}

//Set a value for a cell with textfield editor
setCellValueByLabel(driver, label, column, value) {
    return driver.findElement(By.xpath(`//tr[td/div[text()="${label}"]]/t
        .then((element) => element.click())
        .then(() => driver.wait(until.elementLocated(By.xpath(`//tr[td/di
        .then(() => driver.findElement(By.xpath(`//tr[td/div[text()="${la
        .then((element) => element.clear())
        .then(() => driver.findElement(By.xpath(`//tr[td/div[text()="${la
        .then((element) => element.sendKeys(value))
        .then(() => driver.findElement(By.xpath(`//tr[td/div[text()="${la
        .then((element) => element.sendKeys(Key.ENTER));
}

//Set a value for a cell with combobox editor
setCellComboValueByLabel(driver, label, column, search, value) {
    var query = `//tr[td/div[text()="${label}"]]/td[${column}]`;
    return driver.findElement(By.xpath(query))
        .then((element) => element.click())
        .then(() => driver.wait(until.elementLocated(By.xpath(query + '//
        .then(() => driver.findElement(By.xpath(query + '//input')))
        .then((element) => element.sendKeys(search))
        .then(() => driver.wait(until.elementLocated(By.xpath(`//li[text(
        .then(() => driver.findElement(By.xpath(`//li[text()="${value}"]`
        .then((element) => element.click())
        .then(() => driver.findElement(By.xpath(query + '//input')))
        .then((element) => element.sendKeys(Key.ENTER));
}

//Set a value for a cell with numberfield editor
setCellSpinnerValueByLabel(driver, label, column, numberClicks) {
    var query = `//tr[td/div[text()="${label}"]]/td[${column}]`;
    return driver.findElement(By.xpath(query))
        .then((element) => element.click())
        .then(() => driver.wait(until.elementLocated(By.xpath(query + '//
        .then(() => driver.findElement(By.xpath(query + '//input')))
        .then((element) => {
            for(let i = 0; i < numberClicks; i++)
                element.sendKeys(Key.ARROW_UP);
            })
        .then(() => driver.findElement(By.xpath(query + '//input')))
        .then((element) => element.click());
}

//Click in a specific cell
setClickByLabel(driver, label, column) {
    return driver.findElement(By.xpath(`//tr[td/div[text()="${label}"]]/t
        .then((element) => element.click());
}

//Scroll to the first record of the grid
scrollToTop(driver, label) {
    return driver.findElement(By.xpath(`//tr[td/div[text()="${label}"]]`)
        .then((element) => driver.executeScript("arguments[0].scrollIntoV
        .then(driver.sleep(500));
}
```

Conclusions

It can be concluded that although there are recording tools available to perform automated tests, in more complex scenarios these tools do not work well to identify very dynamic screens, making it necessary to use more specific programs. Selenium's advancement through the webdriver provides a very rich API for browser manipulation, even though there is not yet complete documentation of some more specific features for JavaScript.

This article focused on how to perform automated tests for more dynamic scenarios such as ExtJS, it is believed that the generated documentation can serve as a guide in the beginning of projects and in the understanding of some functionalities.

It is also concluded that the Java API for the development of integrated tests is more documented and mature for the creation of integrated tests.

References

https://en.wikipedia.org/wiki/Test_automation (https://en.wikipedia.org/wiki/Test_automation)

http://www.seleniumhq.org/ (http://www.seleniumhq.org/)

https://www.sencha.com/products/extjs/ (https://www.sencha.com/products/extjs/)

https://sites.google.com/a/chromium.org/chromedriver/home (https://sites.google.com/a/chromium.org/chromedriver/home)

https://github.com/mozilla/geckodriver/releases (https://github.com/mozilla/geckodriver/releases)

Com as etiquetas :
automated,
ExtJS,
java,
JavaScript,
nodejs,
script,
Selenium,
Software,
technology,
Test,
webdrive

Publicado por schmittjoaopedro

Graduado como bacharel em Sistemas de Informação pelo Centro Universitário Católica de Santa Catarina campus Jaraguá do Sul. Formado no Ensino Médio pelo Senai com Técnico em Redes de Computadores Articulado. Atualmente desenvolvedor JEE/Web em Sistemas de Engenharia na WEG. Pesquisador no período de faculdade em Informática pela Católica de Santa Catarina. Contato 47 - 99615 2305 E-mail: schmittjoaopedro@gmail.com Web page: https://joaoschmitt.wordpress.com/ Linkedin: https://www.linkedin.com/in/joao-pedro-schmitt-60847470/ Curriculum lattes: http://lattes.cnpq.br/9304236291664423 Twitter: @JooPedroSchmitt Ver todos os artigos de schmittjoaopedro

4 thoughts on “Automated Testing in Java and Node JS with Selenium and ExtJS 6.5”

hulian.com@gmail.com diz:
31 31+00:00 Agosto 31+00:00 2017 às 15:05 | Editar
Hi, is there any method to test drag-n-drop in an ExtJS grid using Selenium?
We want drag one row in an index to another index in the same grid using Selenium. We found the sample DragAndDropTest.java, but that does not work for grid row. We want some samples, but any help is appreciated.

Responder

schmittjoaopedro diz:
2 02+00:00 Setembro 02+00:00 2017 às 15:45 | Editar
Hello, you can test the same method just by changing the targets. In my experience I had problems with my drag-and-drop tests in other browsers, I was able to make it work only in Mozilla Firefox.

Responder

hulian.com@gmail.com diz:
5 05+00:00 Setembro 05+00:00 2017 às 14:51 | Editar
Yes, we changed the targets to the rows we want to drag, both from-row and to-row are html elements, and both can fetch by Selenium. But the drag-and-drop operation does not work.
The OS we used is Ubuntu 16.04, the browser we used is Chrome 59. The test code is something like following:

```
public void test() {
// fromRow is a html element
WebElement fromRow = driver.findElement(By.xpath(“path for from row”));
// toRow is a html element
WebElement toRow = driver.findElement(By.xpath(“path for to row”));
Actions actions = new Actions(driver);
actions.dragAndDrop(fromRow, toRow).perform();
}
```

schmittjoaopedro diz:
5 05+00:00 Setembro 05+00:00 2017 às 22:36 | Editar
Give a look at this link: https://bugs.chromium.org/p/chromedriver/issues/detail?id=841

Responder