Apache OFBiz® Selenium-WebDriver

Version unspecified

Table of Contents

With this project it’s possible to write and run Apache OFBiz User Interface Tests, unit tests or Business Use Cases Tests (or demo).

There are multiple language or technique to write / record selenium, the choice for this project is to use java, even for Business Analyst.
The main reason for this choice is

1. all "autocompletion" mechanism and javadoc reading when writing code;
2. having a full IT language.

Obviously, usage of java should be very simple for all non IT people, Business Analyst or Product Owner or …

1. Selenium tests Type and Goal

It’s possible to realize multiple Test Type with selenium-webdriver, there are in the project some example for each type.

- User Interface Unit Test, call in this documentation Unit Test
- Business Use Case test, call in this documentation, Scenario Test
- Performance Test, they are very closer than Scenario Test but the Goal is different

For each type there are specifics constrains and Goals/rules.

1.1. Unit Test

Goals

Each test must test only one, functionality, clearly defined
Each test should test all actions / messages for the functionality in one context.
Running Unit Test should be not too long, to be able to run often and so detect new bugs rapidly.
Data used for these tests are dedicated for them, load with ext-tests and it’s not necessary they area similar to real business data.

Constrains

Error correction, when error is detected by a Unit Test, should be swift, because test boundary is very clear.
A Unit Test is not build to be used by a scenario test, so, if part of code could be used by Scenario Test, dedicated method should be created for that.
All methods, like simple action, call by scenario test, should be used by unit test. For Unit Test, it’s not necessary to include a lot of Log Massages, it’s the assert message which should explain clearly where is the problem when test failed.
Unit test are short, so showInfoPanel is not necessary, only Log Messages.

1.2. Scenario Test

Goals

Chain an action set corresponding to a Business use case, in a single ofbiz application or on multiple ofbiz Application (Back Product, Order, Front eCommerce, …).
Test should be limited to only one Use Case, it’s better to do 3 Scenario test than 1 mixing the 3. A Use Case can be simple, like Product validation or be related to a tutorial like product feature creation (back) and usage on the front.
A scenario test can be used as a tutorial, for that it’s necessary to use some showInfoPanel explain, on a user point of view, each step.
Scenario Test Data should be similar to real business data and dedicated to the test, some base data, existing before (like demo data for example) could be checked or updated by a pre-scenario dedicated which is run just before running Scenario test.

Constrains

A Scenario Test is build by using a lot of single action method used by unit tests.

As a Scenario test, test only one Use Case, only information about this use case is tested.
Most of the log message are done by showInfoPanel to explain the current step in the course of the use case.
Messages should be explicit for action and data used. Be clear between showInfoPanel for functional usage and log message for "functional & techniques".

1.3. Performance Test

Goals

Goal is not to test if it works, but if it works at a correct speed with multiple users.
It must be possible to run it multiple time in one environment.
It should not be too short, to be able to run multiple instance of the test with the first instance continuing to run when the last start.
Test should be limited to only one use case, but to be more long, with multiple data.

Constrains

Most of time performance test is base on a scenario test, or use methods coming from them.
Constrains are more on the technical infrastructure, to be able to run multiple instance in the same time.

2. Installation

OfbSwd project is on its own, but it need that the ofbiz which will be tested is on, the ofbiz can be local or on an other infrastructure.

Most of time developers (technical or functional) works in the same time on application development and associated tests, so most of time Apache OFBiz and OfbSwd are install on same computer alongside one another.
This installation chapter is base on this configuration.

```
your development directory/
 ├──── ofbiz-framework/       - Base directory for your Apache OFBiz installation
 ├──── OfbSwd/                - Base directory for Ofbiz Selenium WebDriver
```

OfbSwd project contain all libs you need in the "standard" situation, but you will need to download some specifics drivers depending of the web browser and its release you want to test.

The "standard" situation is browser Firefox and Chrome (or Chromium) at the last release, so you need to have one of them installed on your computer.

2.1. selenium.js in OFBiz

It's not necessary to modify OFBiz to be able launch a selenium because selenium is only a web browser automate, **but** if you want to use showInfoPanel() method which print on the current web page a message, you need to add a javascript (selenium.js) in all ofbiz pages (where showInfoPanel() can be call).

For that you need do :

1. copy selenium.js in OFBiz sub-directory themes/common-theme/webapp/common/js/plugins/
   selenium.js is located in OfbSwd project in sub-directory ofbizFiles/themes/common-theme/webapp/common/js/plugins/
2. modify OFBiz file CommonScreens.xml which is in sub-directory themes/common-theme/widget in screen GlobaActions, add one more line
   <set field="layoutSettings.javaScripts[+0]" value="/common/js/plugins/selenium.js" global="true"/>

```
--- themes/common-theme/widget/CommonScreens.xml    (revision 1855606)
+++ themes/common-theme/widget/CommonScreens.xml    (copy of travail)
@@ -135,6 +135,7 @@
                 <set field="layoutSettings.javaScripts[+0]"
value="/common/js/util/setUserTimeZone.js"/>
                 <set field="layoutSettings.javaScripts[+0]"
value="/common/js/plugins/moment-timezone/moment-timezone-with-data.min.js"
global="true"/>
                 <set field="layoutSettings.javaScripts[+0]"
value="/common/js/plugins/moment-timezone/moment-with-locales.min.js"
global="true"/>
+                <set field="layoutSettings.javaScripts[+0]"
value="/common/js/plugins/selenium.js" global="true"/>
             </actions>
             <widgets />
         </section>
```

2.2. Run jenkins job

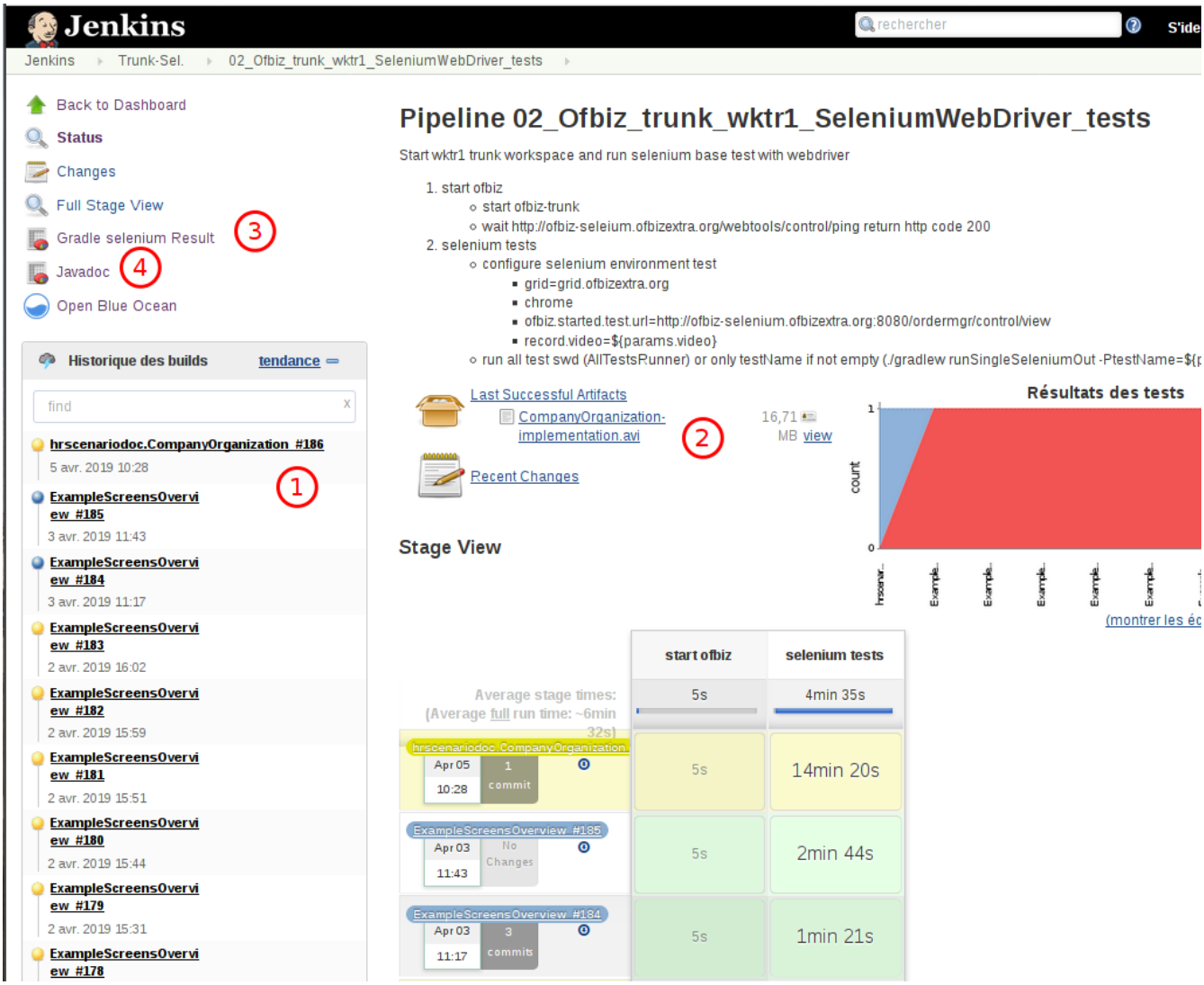
In ofbizextra project Jenkins is used as task automation and scheduling, so this chapter explain "selenium jobs" in a jenkins context, but of course all will be very similar with other task automate tools.

First of all, https://jenkins.ofbizextra.org/ can be used to look at selenium run result like video.
Secondly, it can be used to check if a installation type or a process is still running. For example, daily the selenium scenario for HR component is running on the Apache OFBiz Trunk demo, if it was run without problem and on your installation it does’nt run, the problem is on your configuration not in OFBiz or OfbSwd ☺.

For SeleniumWD, there are 3 job types

- job to install / prepare an OFBiz environment (which will be use for test)
- job to run one (or multiple) selenium test, theses job have multiple parameters to choose context of launch (video or not, showInfoPanel or not, which ofbiz environment, which test, build number, ….)
- job to launch job running selenium to be able to plan it regularly, example: run every week this and this scenario test.

Most of time for Job running a Selenium test the job Id is change with SeleniumTestName.



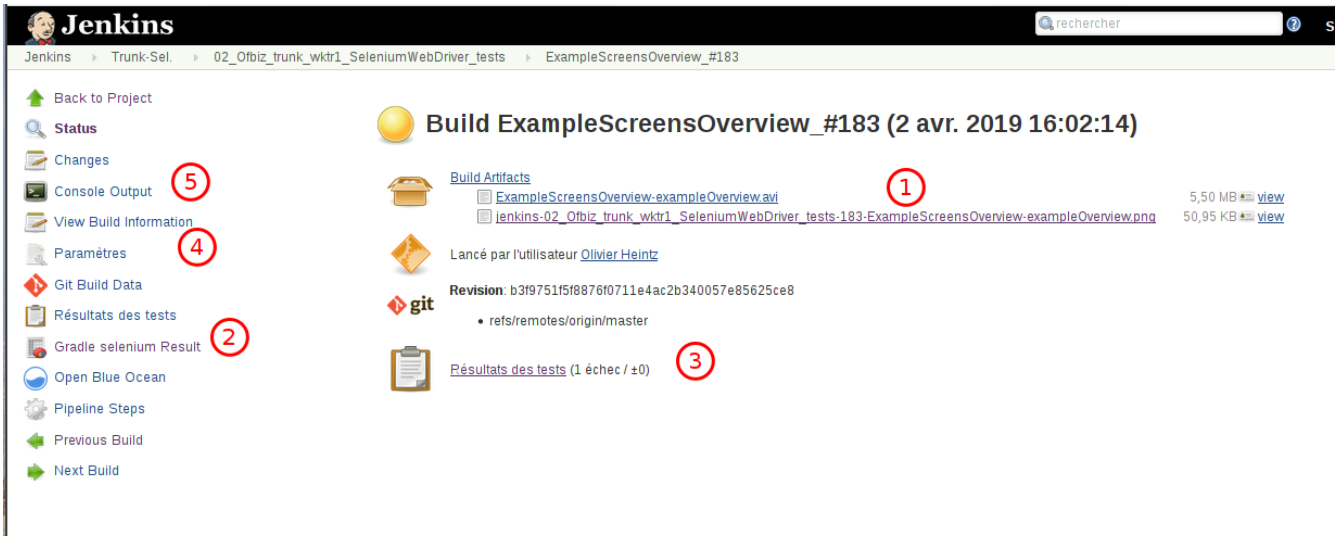
① list of last run, with selenium testName behind the jobId

② the files generated by the last run, here just the video

③ Selenium Test result of the last run

④ Javadoc for OfbSwd project

<u>After clicking on one job in Build History</u>



① the files generated by the build, most of time, at least one avi and one png(screenShot at the end)

② the detail of the Selenium test result (gradle presentation)

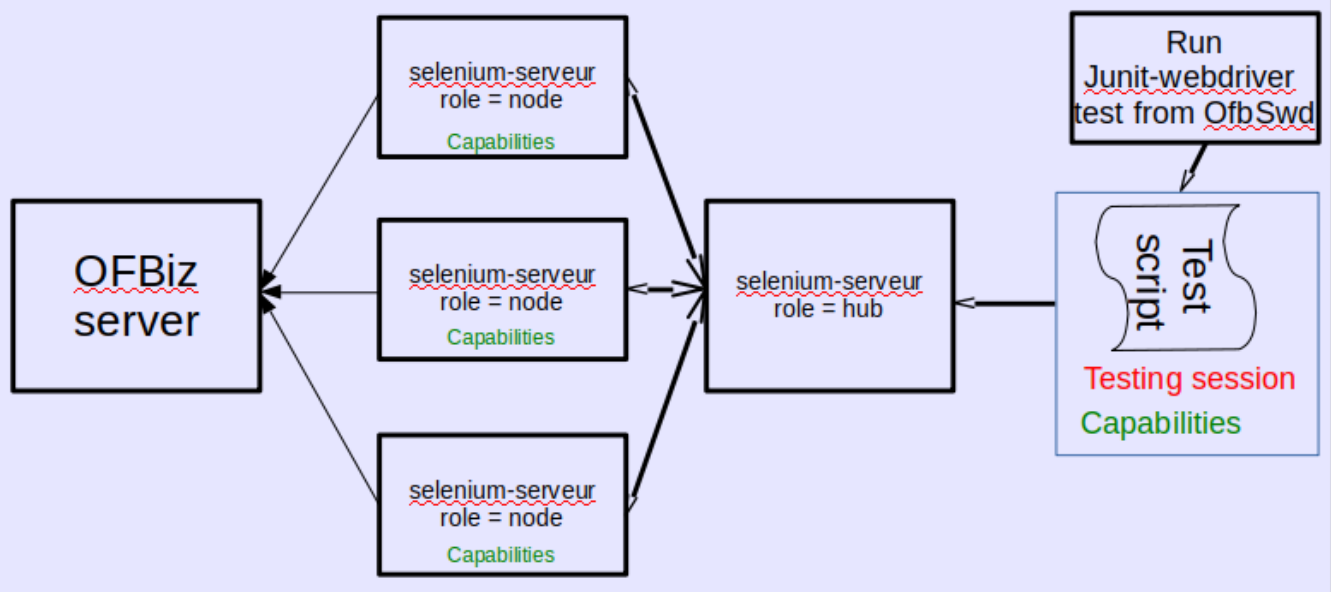
③ more simple presentation for the Selenium test result, the jenkins standard presentation

④ to view parameters used by this build

⑤ to view the "unix" command log for this build

2.3. Run a test on a local computer

Multiple configuration are possible, this chapter explain the default used solution : grid.



So, it's necessary to run 4 process
so 4 terminal when you run interactively the process on a single computer :

1. ofbiz to be able to test it
2. hub : tools/hub.sh
3. node : (minimum 1) tools/node.sh
4. OfbSwd to run the selenium test : ./gradlew runSingleSelenium -PtestName=YOUR-TEST

When the test is run, the file selenium.properties is read to have details about technical environment (Capabilities) of the test, the three main are :

- ofbizBaseUrl , can be local or remote
- Browser
- Browser release.

Most of time, when OFBiz is running locally, it's necessary to have `acceptInsecureCerts=yes` because there is no valid certificate.

Before running node, check if parameters of file tools/node-conf-V3.json is aligned with your environment (mainly browser release). When hub and node are **not** running on the same

computer, the uri should be adjust.
For a node there are 2 place for hub uri adjustment : in node.sh `-hub http://192.168.xxx` and in node-conf-V3.json `"hub": "http://192.168.xxx"`

2.3.1. launch the selenium

In the 4th terminal, launch the selenium test with gradle cde

```
./gradlew runSingleSelenium -PtestName=TheSeleniumTestYouWant
```

If you need to launch multiple tests, use a dedicated Suite class like AllTestsRunner.java file in src/main/java/org/ofbizextra/ofbswd/test/ directory and put in it the list of tests wanted.

On the terminal, at the end of the test, there is only a simple message to say if test failed or not
If test is OK there is something like

```
<==========----> 75% EXECUTING [36s]
> :runSingleSelenium > 0 tests completed
> :runSingleSelenium > Executing test
org.ofbizextra.ofbswd.test.ExampleScreensOverview
```

To have more details go to file:///yourPathToTheProject/OfbSwd/build/reports/tests/runSingleSelenium/index.html
and files generated are in file:///yourPathToTheProject/OfbSwd/build/test-output/

2.4. Browser Driver

Driver for chrome and firefox are in the directory tools/lib, but if they are not compatible with yours release browser you want them for an other release, you can download some other release :

- https://sites.google.com/a/chromium.org/chromedriver/downloads
- https://github.com/mozilla/geckodriver/releases for firefox

and for all other browser, go to the selenium reference webpage https://docs.seleniumhq.org/download/ in the chapter *Third Party Drivers, Bindings, and Plugins*

On linux debian environment, there is a package chrome-driver which is synchronize with package chrome to be sure to have the correct driver (on /usr/bin) for your chromium browser.

3. Write Selenium WD

3.1. Record Actions

With Selenium, most of time, paper or documentation speak about recording a suite of user actions and re-play it.
It's one of the possibility but it's useful only at the beginning of use Selenium.

For example, to be able to make a full detail User Interface test, there are a lot of point to check if all is correct and there are not a lot of navigation to do, so it's better to start by a short record session and use the result to write the full code wanted.

Other example, to do a scenario test, data is very important and should be easily change without changing code.
When writing a scenario test, Business Analyst is focus on which action to do at each step and don't want to have to choose some technical point (how to select this button ?)

With these two example, it's only show that recording web actions tools are sometime used but not very often, so this documentation present rapidly 2 tools and if more documentation about it, it's better to go directly to their documentation.

Recording tools exist for multiple browser and are add-on for them, Chrome, Firefox, ..., below Firefox is using.

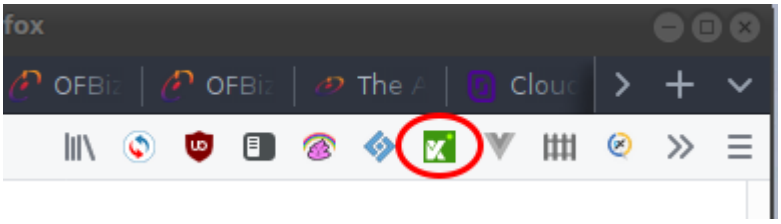
The first well know tools is selenium-ide add-on https://addons.mozilla.org/firefox/user/13759230/. Currently the selenium-ide community decide to a complete re-writing of it to be able to use all features in the new webDriver selenium library.
So, the current version of selenium-ide is no more usable for our context, because export actions recorded is not yet available.

The second tools is katalon-recorder, https://addons.mozilla.org/firefox/user/12712544/ it could import file generated by selenium-ide and have multiple similarity with selenium-ide.

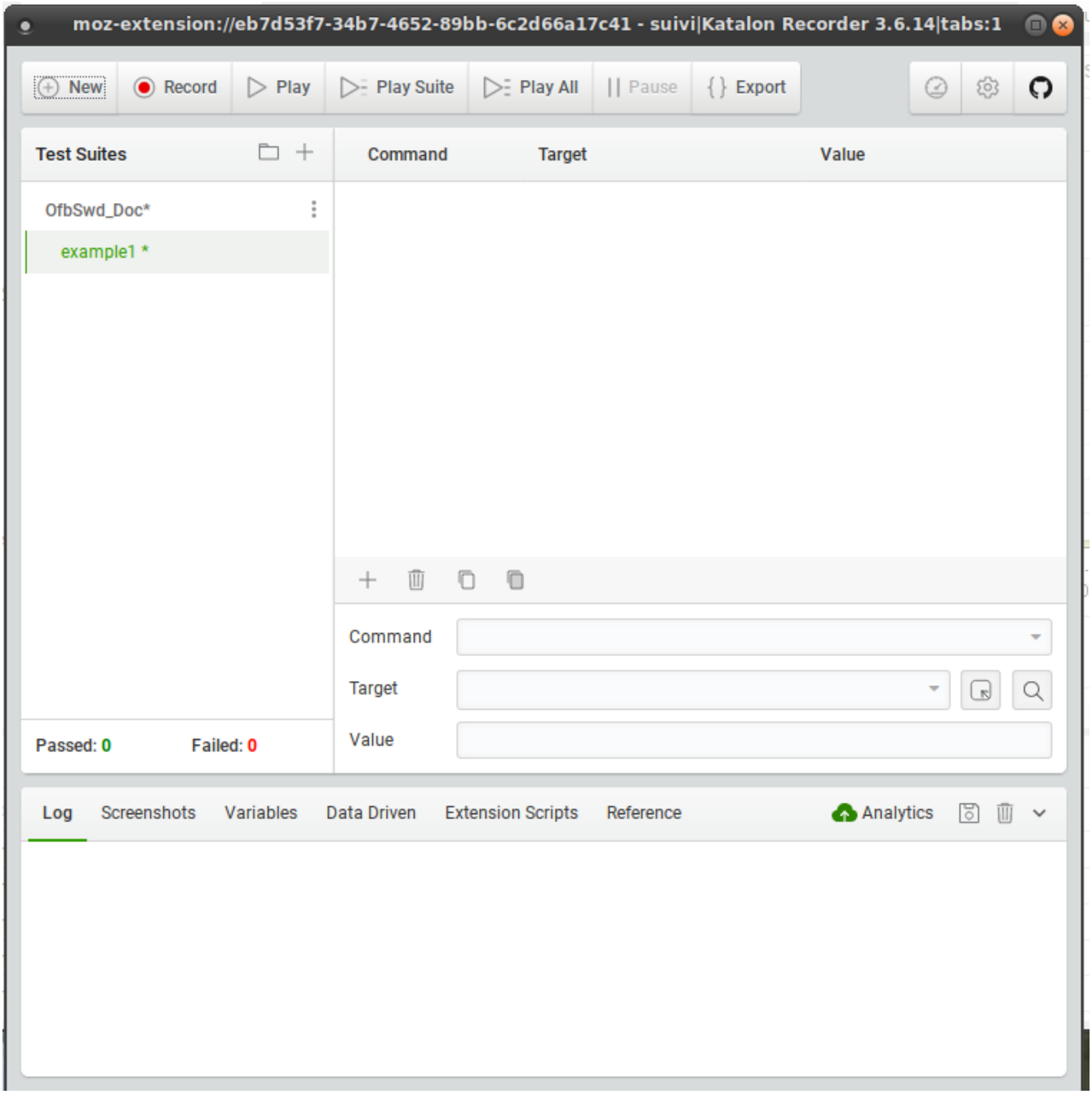
It's possible to use the two tools in the same time if needed.

3.1.1. With Katalon-recorder

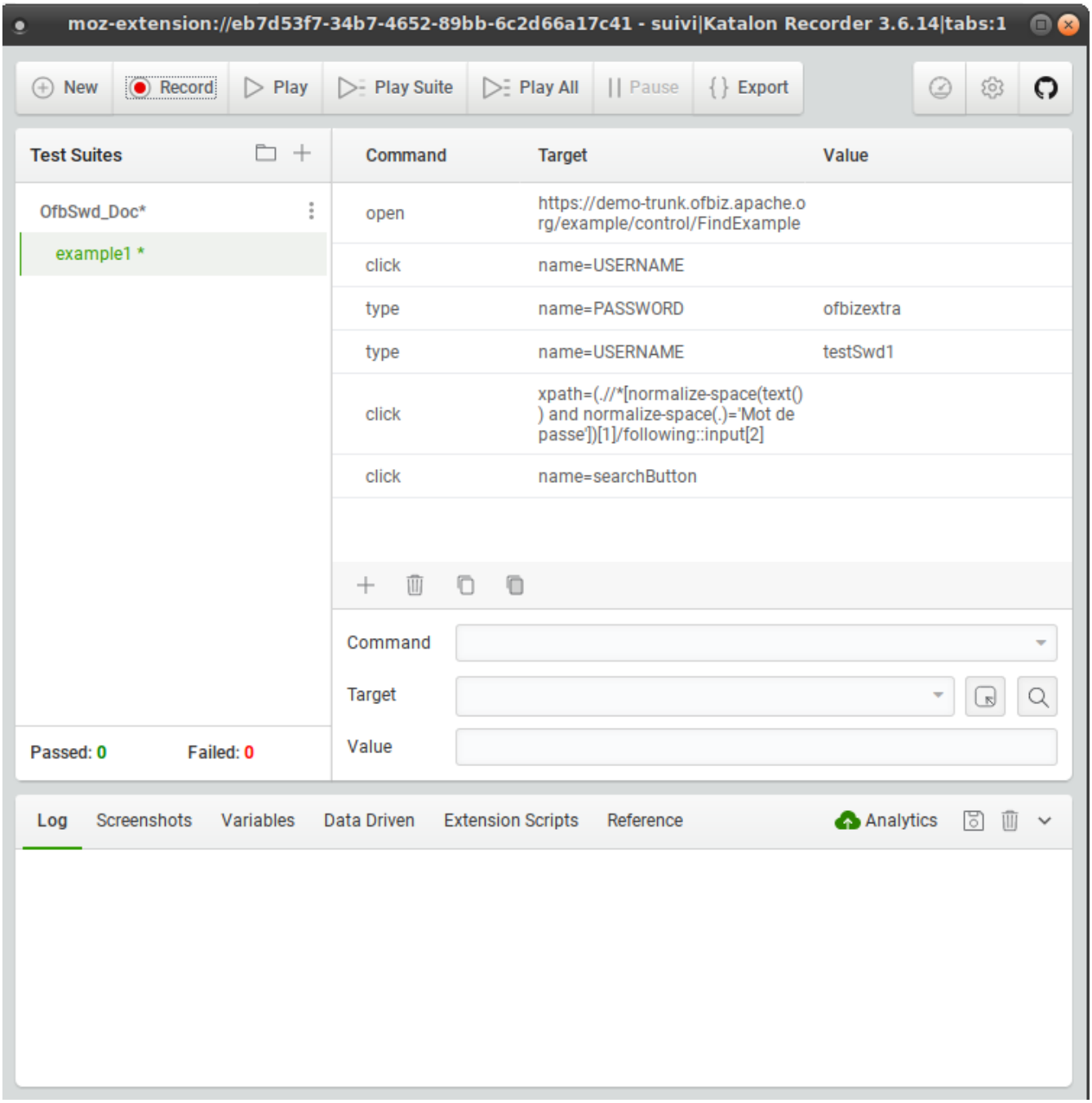
After add-on installation, there is a new icon in browser



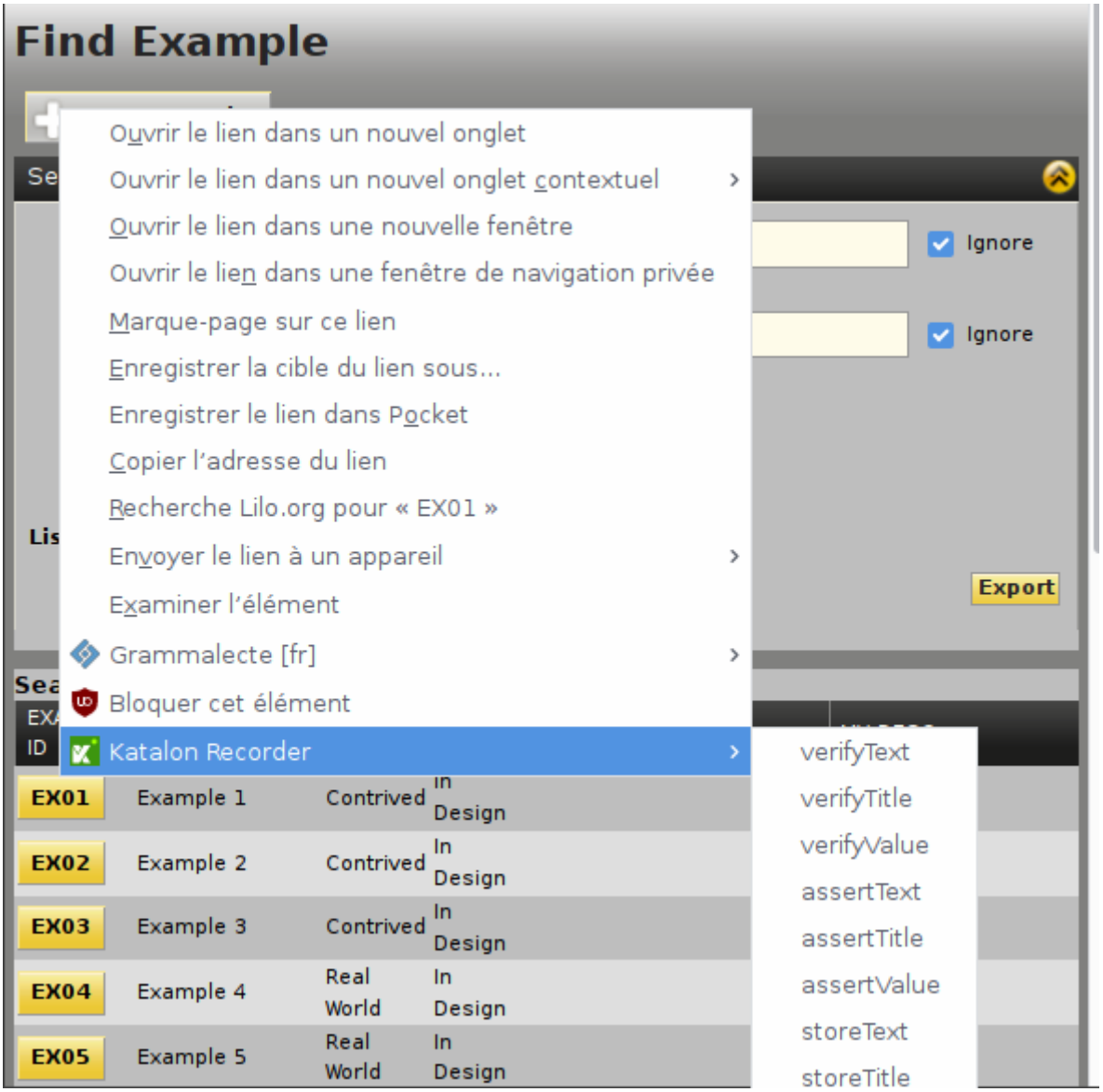
Open a tab with OFBiz, open katalon-recorder (by clicking on icon), choose SuiteName and TestCase Name



Now, click on record button, refresh browser tab and do some actions (login, search, ...)
the recorded actions will appear in the Katalon windows



It’s possible to add some check or store with right-click on a web element and choose in katalon contextual menu
Example : on EX01 link



Of course, there are all what is needed to modify the recorded actions and adjust to have what you want, and test it with the play button.

As we work with java, the next step is to export to java(WebDriver + JUnit) and copy code portion it's needed

```
@Test
  public void testExample1() throws Exception {
    driver.get("https://demo-trunk.ofbiz.apache.org/example/control/FindExample");
    driver.findElement(By.name("USERNAME")).click();
    driver.findElement(By.name("PASSWORD")).clear();
    driver.findElement(By.name("PASSWORD")).sendKeys("ofbizextra");
    driver.findElement(By.name("USERNAME")).clear();
    driver.findElement(By.name("USERNAME")).sendKeys("testSwd1");
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Password'])[1]/following::input[2]")).click();
    driver.findElement(By.name("searchButton")).click();
  }
```

For detail about Java syntax see next chapter !

With this tools, it's more easy to find how to locate a specifics webElement, in the katalon windows in field target, there are multiple choice. It's useful to learn the different method !
As it's possible change manually the target field content, the magnifier button is perfect to check if the syntax is correct to select the webElement wanted.

So, this tools is useful to record AND to help finding/checking good syntax for xPath or CssSelector or ...

3.2. Java files

3.2.1. General points

In the OfbSwd Project there are some existing tests so some java files which are good to use them as example of how to do.

There are two test writer type, technical for Unit Test or usage method and Business Analyst for scenario test. It's too difficult to write two different documentations, so if you are a Business Analyst and some part seem too technical, read the more faster and don't worry, you will not used them ☺ .

When a test failed sometime and is ok otherwise, most of the time the problem is a **Waiting defect** :
When a test is running, it's faster than with a human. For example, after a click, program immediately do the next action without waiting the complete screen refresh and so, if server is slow, the webElement for next action is not yet present.
The best practice is to put a waitElementPresent before using the element.

Most of time, the waitElementPresent is automatically done by the internal OfbSwd method (see in next chapter, SimpleTestSuite), so you have not to manage it, but don't forget the previous point when there is a "random" error.

The second point is a feature of Junit test which is not good in selenium context. Junit test engine is used to run selenium test.
When there are multiple testCase in one java testSuite file, all are run when the testSuite is run but there is no way to say in which order the test will be executed.
If a test is compose by :

- creation (one method)
- modification (one method)
- searching (one method)
- ... (one method)

it's not possible to say creation has to be done before modification. In this case, the only solution is to have only one test which call the different method.

link

seleniumhq javadoc : https://seleniumhq.github.io/selenium/docs/api/java/index.html

3.2.2. Main OfbSwd Class

All selenium java file are in the directory OfbSwd/src/main/java/org/ofbizextra/ofbswd/test/ and subdirectories

SimpleTestSuite.java

All TestSuite Java file inherit from this class, it contain all the running context loading and all the internal "framework" method.

In this file there are some method with name similar to seleniumhq method, it's advisable to use them in place of seleniumhq because they included some waitElementPresent/clickable and manage javascript wait process loader-wrapper and ...
The goal is to have, in test, a simple syntax, but behind, all actions needed to have reliable test.

At the same level there are all tests testSuite class.

utils subdirectory

Contain the helpers class which help developers with ready to use method for classic needs in selenium tests.

use subdirectory

Should contain all class which can be call for test, unit or scenario.

The best practice is to have one class by OFBiz component, and method name should be similar to possible action done by a user.

3.2.3. Test and data

Data used for the tests should be externalize in data dedicated files (directory tests-data).

Test Data are manage with xml files using a scenario reference. When a test is running, the data scenario name is read in the selenium properties : line < dataScenario=default >

In Test Data file, in scenario, test-suite name and test-case name must be exactly the same that in the tests file.

in TestSuite java

As all Testsuite class inherit from SimpleTestSuite class, some variables/objects are initialized to facilitate reading data.
testCaseData is available to have access to all data-obj include in data file under test-case name.

In best practice, use method should have data-object as parameters except when they need less than 2 parameters data.

for indexed data, access is done with getString()

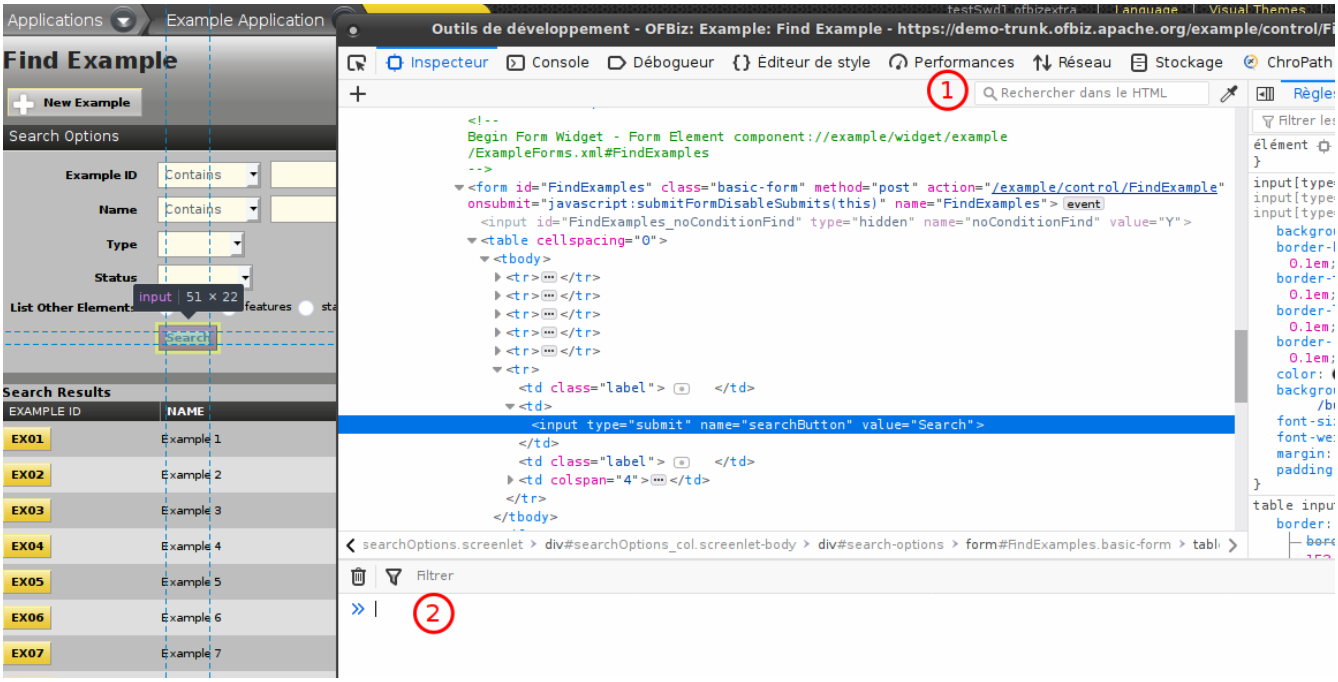
3.2.4. HowTo find WebElement

In all unit test method or use method, finding or using webElement is at each step, and there are multiple manner to do it.
Preference order to use is

1. id because normally it should be unique on a web page
2. name because it's very clear
3. cssSelector because browser engine is very efficient on it
4. xPath it's same as cssSelector but maybe less efficient

For cssSelector and xPath, shortly as possible to be readable.

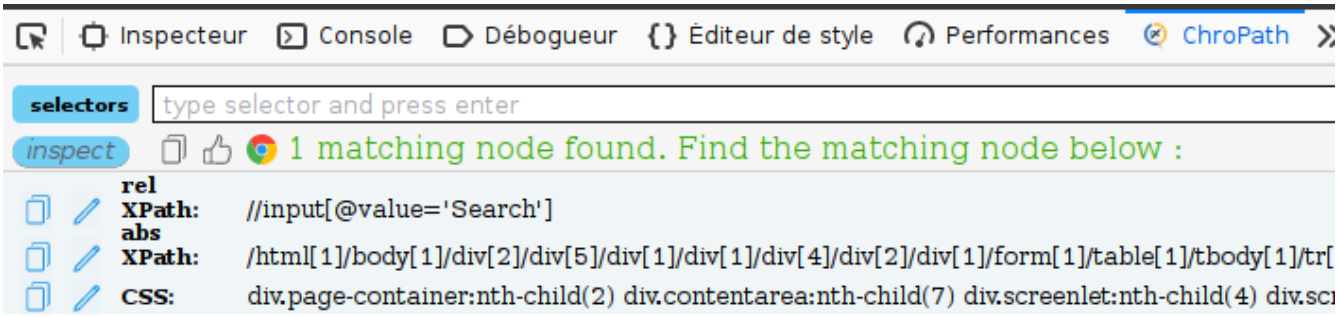
In chrome or Firefox, if a right click is done on a WebPage element and "inspect an element" is selected the web console is open with the element selected.



① it's possible to directly enter a cssSelector to check if it select the correct element (or if it select multiple elements). Be careful, not all cssSelector syntax is manage

② it's possible to enter directly a cssSelector `$("cssSElector_expression")` to test it and show element
or a xPath `$x("xpath_expression")`

Some browser addon exist to show xpath or cssSelector after select a webElement, one of them is chroPath for Firefox https://addons.mozilla.org/firefox/user/14010447/




```
<input type="submit" name="searchButton" value="Search" xpath="1">
```

3.2.5. General Advice

Writing selenium use time, but corrections & evolutions is a more large part of the Selenium life time.
Each time UI framework or UI change, process change, demo data or data change, selenium should be adapted. If developer do it in the same time he know were to change, but when the starting point is a selenium failed, often it's not the new dev which is a problem but the selenium which should be adapted.
So, selenium documentation and comment are very important to give access to everyone to be able to change and make evolution in it.

Comment

To do good Selenium, do it step by step, and add more comment than in a standard program.

```
// This is a comment on one line
```

```
/* This is
   a comment on multiple lines
 */
```

simple ' better than \"

When a copy paste is done about a xPath or cssSelector all «"» are protected by «\» and it works but it's not readable. Prefer to change them by single «'» it will be better

After copy paste

```
assertTrue("Test1",driver.findElements(By.cssSelector("img[title=\"Test1\"]")).size()>
0);
```

more readable :

```
assertTrue("Test1",driver.findElements(By.cssSelector("img[title='Test1']")).size()>0)
;
```

3.3. CSS Selector

Css selector engine in browser seem to be, most of time, more efficient than xPath, so it's better to use it ☺ .

Some OfbSwd build in method, to simplifying writing selenium, use a string parameter as a cssSelector using in a By.cssSelector() For example :

```
testSuite.click(cssSelector);
```

Each time you need to test or find a cssSelector (or xPath) you can use recorder tools or internal browser tools or addon

3.3.1. ID selector

syntax :

```
"<tag-name>#id"
```

examples:

- select input with id "username"

  ```
  By.cssSelector("input#username")
  ```

- select div with id "editarea"

  ```
  By.cssSelector("div#editarea")
  ```

3.3.2. CLASS selector

syntax :

```
"<tag-name>.<className>[.<className>]"
```

examples:

- select div with class portletInfoDetails-link

```
By.cssSelector("div.portletInfoDetails-link")
```

- select link "a" with class "expanded"

```
By.cssSelector("a.expanded")
```

- select table with class "basic-table" and "hover-bar"

```
By.cssSelector("table.basic-table.hover-bar")
```

3.3.3. ATTRIBUTES selector

syntax :

```
"<tag-name><[att-name=att-value]>[<[att-name=att-value]>]"
```

examples:

- select button with "Store" as value

```
By.cssSelector("button[value='Store']")
```

- select image with "Create" as title

```
By.cssSelector("img[title='Create']")
```

- select input with "userName" as name and "identifier" as placeholder

```
By.cssSelector("input[name='userName'][placeholder='identifier']")
```

3.3.4. Text visible selector

With cssSelector it's not possible to select a webElement by it's contain text.
For selenium the solution is to use By.linkText or By.partialLinkText when it's for <a> tag.
In all other cases xPath is the solution.

examples:

- select link with "My Fruit" as visible text

```
By.linkText("My Fruit")
```

- select link with text containing word "fruit"

```
By.partialLinkText("Fruit")
```

3.3.5. More complex syntax

- startsWith "^"

```
By.cssSelector("button[value^='Recor']")
By.cssSelector("div[id^='edi']")
By.cssSelector("input[name^='userna']")
By.cssSelector("a[class^='currentEl']")
```

- contains "*"

```
By.cssSelector("button[value*='tore']")
By.cssSelector("div[id*='itr']")
By.cssSelector("input[name*='erna']")
By.cssSelector("a[class*='rrentElem']")
```

- endsWith "$"

```
By.cssSelector("button[value$='orded']")
By.cssSelector("div[id$='rea']")
By.cssSelector("input[name$='rname']")
By.cssSelector("a[class$='rrentElement']")
```

3.3.6. advanced selector

- direct child ">"

```
By.cssSelector("div#main>a")
```

- child, direct or not, never mind the level, " "

```
By.cssSelector("div#main a")
```

- next element at the same level "+"

```
By.cssSelector("div#main + div")
```

- first child "first-child" (no check on type)

```
By.cssSelector("div#main *:first-child")
```

- first p in child "first-of-type"

```
By.cssSelector("div#main p:first-of-type")
```

- first child p "first-child"

```
By.cssSelector("div#main > p:first-child")
```

- third p "nth-child(3)"

```
By.cssSelector("div#main > p:nth-child(3)")
```

3.4. XPath

Often, xPath is proposed by recorder tools or by WebElement localizer, when you don't find how to use id or name, use the more shorter xPath you can.

Table 1. Code to use

Goal	Code example
Starting from div with a id and give a path in the DOM	By.xpath("//div[@id='scrlt_ExampleItems']/div/ul/li/a")
Starting from div with a id and select the next div with class ...	By.xpath("//div[@id='scrlt_ShowExample']/div[@class='screenlet-title-bar']")
Starting from a form with a id and select the first td below which contain 'test description' maybe with space before or after	By.xpath("//form[@id='ShowExample']//td[normalize-space(.)='test description']")
Starting from a form with a id and select the first td below which is 'test description'	By.xpath("//form[@id='ShowExample']//td[.='test description']")

3.5. Tips and Tricks

3.5.1. Running multiple time same test

A test is linked to the base data and its test data associated, so, without specifics rules / actions it will failed if it's run a second time in same environment.

It’s not a good thing because for analyze or testing corrections or hypothesis it’s usual to need to run multiple time the test and re-load all data is sometime long.
So, in best practice for writing test and associated data, there is data indexing (for main data). Indexing is defining string data as indexed and when running test, a buildNumber (job parameters) is used to suffix the data. With indexing, most of time, it’s possible to run multiple time the same test on one environment. BuildNumber job parameters default value is usually 1, but it’s possible to change it when job is run manually.

With indexed data, when you analyze data by using application, it’s often necessary to read test log to have the correct value.

3.6. Code Example

Table 2. Code example

Goal	Code example
select by id	By.id("xxxxx")
select by name	By.name("exampleName")
select by link text	By.linkText("Search")
select by class name	By.className("nav-next")
Waiting for a id	driverWait.until(ExpectedConditions.presenceOfElementLocated(By.id("xxxxx")));
waiting id is no more visible or disappear from DOM	driverWait.until(ExpectedConditions.invisibilityOfElementLocated(By.id("xxxxxx")));
waiting a webElement (found before) disappear from DOM (if it’s rebuild it’s not the same)	WebElement we = driver.findElement(By.name("submitButton")); source.click(By.name("submitButton")); **driverWait.until(ExpectedConditions.stalenessOf(we));**
waiting a text appear (it can contain more)	driverWait.until(ExpectedConditions.textToBePresentInElement(By.cssSelector("li.nav-displaying"), "Displaying 21 - 25 of 25"));
Print a message and wait 2 seconds	LogSelenium.showInfoPanel("features will be modify", 2);
waiting for visibility of a element	driverWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("EditFacility")));

- Click on icon (with title) and wait portlet (13.07) title (or part of it) appear, (title is a link to help)

```
driver.findElement(By.cssSelector("img[title='Items']")).click();
driverWait.until(ExpectedConditions.textToBePresentInElement(By.xpath("//div[@id='scrlt_ExampleItems']/div/ul/li/a"), "List Example Items EX14 :"));
```

- Click on icon (with title) and wait portlet (13.07) title (or part of it) appear, (title is NOT a link to help)

```
driver.findElement(By.cssSelector("img[title='Items']")).click();
driverWait.until(ExpectedConditions.textToBePresentInElement(By.xpath("//div[@id='s
crlt_ExampleItems']/div/ul/li"), "List Example Items EX14 :"));
```

- Click on icon (with title) and wait form Form submit button appear (so all form is appeared)

```
driver.findElement(By.cssSelector("img[title='Add']")).click();
driverWait.until(ExpectedConditions.presenceOfElementLocated(By.name("submitButton"
)));
```

- Click on back button (link) and wait it's no more visible (it or other in page)

```
driver.findElement(By.linkText("Back")).click();
driverWait.until(ExpectedConditions.invisibilityOfElementLocated(By.linkText("Back"
)));
```

- test if a webElement contain a text (exactly or contain)

```
assertEquals("test selenium",
driver.findElement(By.xpath("//tr[@id='ExampleItems_row_0']/td[2]")).getText());
```

or

```
assertTrue(driver.findElement(By.xpath("//tr[@id='ExampleItems_row_0']/td[2]")).get
Text().contains("test selenium"));
```

- test if form list contain 5 line (the first one is 0)

```
assertTrue(driver.findElements(By.id("ListExamples_row_4")).size()!=0);
assertTrue(driver.findElements(By.id("ListExamples_row_5")).size()==0);
```

- test if icon delete id displayed

```
assertTrue(driver.findElement(By.xpath("//div[@id='ExampleItems_deleteLink_0_div']/
/img[@title='Delete']")).isDisplayed());
```

- wait for alert dialog box appear and validate after checking message

```
driverWait.until(ExpectedConditions.alertIsPresent());
Alert alert = driver.switchTo().alert();
assertTrue(alert.getText().matches("^You confirm [\\s\\S]$"));
alert.accept();
```

or with OfbSwd build in method

```
testSuite.clickWithAlert(By.cssSelector("img[title='Delete']", "You confirm");
```

- OfbSwd build in method to Test if WebElement is present

```
  if(testSuite.isElementPresent(By.id("ListFacilities_facilityId_0_div"))) {...
```

- wait for visibility of

```
driverWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("EditFacility_
facilityId")));
```

- wait for NONE visibility of

```
driverWait.until(ExpectedConditions.invisibilityOfElementLocated(By.xpath("(//div[@
id='Lookup_refId_div']/a")));
```

- wait for presence of a webElement

```
driverWait.until(ExpectedConditions.presenceOfElementLocated(By.id(fieldName)));
```

3.7. Video recording

The jar grid-service-provider is very similar with jar selenium-server-standalone but it add some implicit parameters to be able to record on video test currently running. It record the overall screen, where browser is launch.

At the end of the test, video will be in directory build/outputs with the name className-testMethodName.avi

With SimpleTestSuite, method which all testing methods inherit, start recording video depending on parameters in selenium.properties record.video=yes.

The method LogSelenium.showInfoPanel call a javascript (in ofbiz) which print a message on the browser. This is useful to explain to the people watching test running what is the next step (or result of the previous one).

When there is parameters infoPanelEnabled=yes (in selenium.properties) messages are appear in screen. Each message need a short pause for user be able to read it, the parameter infoPanelEnabled is desactivated, most of the time, on production automatic integration test to have UI test shorter.
With parameter logPanelMessage=yes messages are included in the log output.

4. Selenium with OFBiz

TODO this chapter should be oriented only on OFBiz dedicated methods

All tools or method describe in this chapter are available in the OfbSwd project.

5. Best Practices

5.1. Summary

The main mandatory rules are, ordered by priority

1. A Unit Test must be independent from the others
2. New function should be associated to its tests
   - if there are new services, it must exist, at least, one junit test by service.
   - if there are User interface (portlet, screen, menu), it must be exist a selenium unit test
   - the new function should be used by a selenium scenario test
3. No data included in the tests, all data must be read in data files
   - all method which need a set of data should have one (or more) data-obj as parameters, not a list of simple parameters.
     The goal is that method is for a business object, not a set of field.
     - data-obj parameters name should be the OFBiz object name.
   - Each time a data-obj's field must be unique, it's necessary to use the indexed type. Like this the build-number (in selenium.properties) will be use as suffix when this field will be read.
4. Each method usable in selenium scenario, should have a javadoc, readable by a "Business Analyst" (or functional developer) with enough information to use it.
5. When there is a data-obj as parameters, used fields of data-obj should be detailed in javadoc, OR a data-obj template should exist in data template files.
6. Unit test should be run multiple time on the same environment, same for Scenario test on dynamics data (Orders, Shipments, WorkForce, ...).
   Most of time it's not possible on Scenario data for static base date (HR organization, Product, ...), but we should try.
7. For all action methods, put a showInfoPanel at the beginning to explain the future action, on a tutorial perspective.

be careful at the beginning of test, ofbiz page must load to have showInfoPanel working.

Mandatory java rules :

1. standard package naming rules : package, class, method and variable
2. javadoc
3. no warning
4. no tabs.

Testing Environment :

- It’s good to test, at least, in 2 different browser (or more), good practice is to work with one and Integration Test (run by jenkins) use an other. Currently, on ofbizextra platform, most of time Chromium is used by Jenkins, so in development phase Firefox should be used ;-)
- Currently, tests and demonstration are done on a "standard basic" PC, so screen format using is 1368x768 definition
  (just little lesser 1365x765). The screen definition wanted for test is in selenium.properties and when test running selenium ask browser to use this definition if it’s possible.

5.2. Golden Rules

1. It’s always possible to enhance a test, i’s always to test more, but it’s better to have a short test that no test or test not usable because not finish !
2. new development = new test
3. Scenario test is more important than unit test
4. Bug correction or Enhancement development = new simple test or enhancement of current test.
5. Javadoc is the testing document on a Business Analyst perspective. With javadoc I should understand what a doing each method and which method I can call for doing one action.
6. Creation →Search → List → Modification → Desactivation → Activation → Remove

Exemple :

```
/**
 * Payment method testing
 * <ul>
 *   <li>Create a payment mode. ({@link #CreateBankAccount CreateBankAccount})</li>
 *   <li>Modify a payment mode. ({@link #ModifBankAccount ModifBankAccount})</li>
 *   <li>//TODO : desactivate payment mode</li>
 *   <li>remove payment mode. ({@link #DeleteBankAccount DeleteBankAccount})</li>
 * <ul>
 *
 * @param partyId
 *
 * @throws Exception
 */
```

5.3. Inform each time Work In progress or TODO

Use the javadoc syntax to point each part not finish or with workaround with //**TODO** and if it’s about a part of code which should be removed, waiting validation, add a date of the comment and, if necessary, date when code "must" be removed (ex: one month after code is commented)

```
//TODO thing description to do after other thing
```

5.4. Tests organization

About test Class, it's necessary to clearly separated Unit Test Class from Using/Actions Class, or in other word, Class used and write by and for technical developers from methods for Business Analyst (functional developers) to help us to write scenario tests.

First one are for testing each part of user interface : button, Add / Remove, search criteria. Testing elements are chosen by the developers.

Second one are to simulate a "normal" executing a use case, so, building a sequence of actions, with some test but not a lot because if an action failed without test to stop, most of time next step need the result of it and so will failed and stop the scenario.
Example : if test create a product to use it in an order, even if there is no check product creation succeed, order creation will failed.
In scenario test, log message can be use to quickly see where is the starting point of why Test failed.

The ideal testing structure is : each entity (as defined in User manual Glossary) should be tested with a minimum of 5 methods and have 4 methods available for using it by scenario :

- testEntity() : do the Unit tests for the entity with calling for the four other methods.
- createEntite(...) : do the entity creation (with a entity data-obj as parameters), return created id (if possible testing if no error message).
- modifEntite(...) : do modification (if possible test if if modification is done).
- deleteEntite(...) : do suppression (if possible testing if no error message).
- searchEntite(...) : do a search and if necessary select one entity to go to the entity sub-application.
  (if possible testing if sub-application is open at the end).

The method testEntity call the four other methods and some other to be unit test, so having a lot of small test.

If it's possible for a entity to activate or desactivate the object or a related object, 2 other methods should be add :

- activateEntite(...) : do activation.
- deactivateEntite(...) : do desactivation.
- Historic management is, most of time, directly manage in the two methods.

Currently there are so many UI selenium tests, so it's better to start by scenario tests to be able to quickly test a large part of ofbiz functionality.
With unit test we have stronger tests, but for only one entity.
When writing a scenario test, using/action methods are created and it will be easy to use when when writing Unit Tests in futur.

5.5. showInfoPanel() and log()

5.5.1. showInfoPanel()

LogSelenium.showInfoPanel(...) : Method to generate message for user which watch test running or video recorded during the test. Most of time (depending of parameters in selenium.properties) this message appear too in log.

Two main use case for this method :

1. use the test video as a tutorial, the watcher does'nt know the application, so message should be clear and explicit about next action.
2. help Business Analyst which search why test failed, so message should be related to javadoc usage method. When message are correct find when and why test failed is fast.

Some Action methods start by a showInfoPanel to be coherent with previous use case, but the "technical" should not have showInfoPanel because to many message on screen become un-readable. Log message is better for technical message.

If possible, give, in the message, data used by the next action, it's useful to check in the environment after test some other information.

```
> LogSelenium.showInfoPanel(module, source, "Print relationship for (" + partyId +
")", 1);
```

It's possible to give how many second the message will be print on screen, this parameters should be used only in specifics context.
If this parameters (the second) is not given, the default value parameterized in selenium.properties will be used, so it's more useful for user which want run a selenium with a more longer (or more shorter) time to read message to be able to change it.

5.5.2. log()

LogSelenium.log(...) or LogSelenium.info(...) : method to generate message in the log to the developers / debuggers of the test, technical or functional or both ;-) .

To have readable log, it's important to put value added message, that's mean which help to understand the flow of actions and where the test are. If possible put some data used or read by the test.
It's not necessary to put to many message, if you need a very detailed log use LogSelenium.verbose() and activate in selenium.properties verbose=yes.
In log, each message is printed after method name and line where message have been called, so it's not necessary to add these information in the message.

```
LogSelenium.log(module,"id=new_list-name_" + offerId);
```

Result

```
  2016-07-11 11:28:034 front.EcommerceOrder.addList(EcommerceOrder.java:310) ::
id=new_list-name_OB10803
```

In some case showInfoPanel can disturbed the test, for example, if a drop-down is open before message appear it close it. In these cases, it's possible to use log.

Sometime log is read by Business Analyst, when it search informations on a test which failed, so messages should be coherent with javadoc, for example showing some internal information mention in it.

5.6. ID should be indexed

To easily debug an error in test, it's often need to run same test multiple times. To be done quickly it's better if all run on the same Database instance (not doing a re-load between two launch), so all ID (or main data) for Business Object (order, Product, Supplier, ...) should be unique at each run.

To solve this issue, it's possible to "indexed" some field in data-obj, it's mean :

1. indexed data are build (during run) with data read from data-file and a suffix which is the buildNumber (read from selenium.properties). Example : mail address for a front user will be seleniumUser1@o-toit.com the first run and seleniumUser2@o-toit.com the second one.
2. In the data file (scenario / test-suite / test-case / data-obj) at the field level in a data-obj, type should be indexed and not string.
3. No more action is needed it's reading data from data file which index the data, so only a .getString("field") to do.

```
  <data-obj name="login">
      <indexed name="username" value="selenium"/> <!-- indexed via a suffix :
selenium1, selenium2, ... -->
      <string name="password" value="ofbiz"/>
  </data-obj>
```

In log message, it's usefull to give field value which will be used because it will be with the index and it's better to be able to check after the test some information on the OFBiz instance tested.
Some time it's useful to indexed some data (not needed to be unique) just to easily understand which data is from which run, because after multiple run there are multiple similar Product or Customer or ... so description or name with a index is clear.

6. Error analysis

When a Selenium test failed, how to do to be fast to clearly find why it failed, problem in test or in test data or in ofbiz code (bug or just modification not manage in test) or in ofbiz data.
If selenium Test is correctly written and this process (describe in this documentation) followed, most of time, analysis is fast.

preamble : When a test failed, most of time it's a modification or problem in OFBiz application or in data. Don't start to search in the test !

If error is strange because no modifications in code or data since last run without error and all seems correct, try to re-run the test a second time to check if error is still there the second time. Browser or network are the reason of these strange errors

6.1. First Analysis Level

6.1.1. Being like a Business Analyst :

1. identify the test name which failed
2. look at the screenshot (when it failed) for this test
3. read the error message, clear (for you) or technical (and maybe not completely clear ;-))
4. read the stack trace about the methods call at the error time
5. read the javadoc of the test
6. read the javadoc for all method on the stack, starting from the last and working backward, to find in which one you locate the error
7. go to the test log to collect data during test (specifically the indexed one)
8. open the data file used by the test
9. go to the website tested and try to reproduce the fail situation
10. if needed more detail about context, read the log "details"
11. if you could'nt reproduced the error or does'nt understand why it appear, explain to a technical developers your analysis.

Each of these point are detailed below

6.1.2. Read Error Message

For the jenkins jobs, in Test Result (standard or gradle (in left panel)), select the test which failed, Error Message is the 1,2 (or 3) first line under <<Message d'erreur>>

Message is

- **clear** The message is generated by the test and so it should be explicit when you have the context, if it's not the case, don't forget to change it when the problem is found and solved !

- **Technical** Most of the time message is coming from an exception, so technique, a web element not found or not display element. You could read it and understand it, if needed ask someone to explain it.

6.1.3. Read Stack Trace methods call

In the Execution stack (or Pile d'exécution), the stack is something like

pile d'exécution

```
...
normal, platform: LINUX, platformName: LINUX, rotatable: false, setWindowRect: true,
takesHeapSnapshot: true, takesScreenshot: true, unexpectedAlertBehaviour: ,
unhandledPromptBehavior: , version: 69.0.3497.92, webStorageEnabled: true,
webdriver.remote.sessionid: f5bcdc89cad05106f743fbc2ea7...}
Session ID: f5bcdc89cad05106f743fbc2ea771fb0
    at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
    at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.ja
va:62)
    at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccesso
rImpl.java:45)
    at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
    at org.openqa.selenium.remote.ErrorHandler.createThrowable(ErrorHandler.java:214)
    at
org.openqa.selenium.remote.ErrorHandler.throwIfResponseFailed(ErrorHandler.java:166)
    at
org.openqa.selenium.remote.http.JsonHttpResponseCodec.reconstructValue(JsonHttpRespons
eCodec.java:40)
    at
org.openqa.selenium.remote.http.AbstractHttpResponseCodec.decode(AbstractHttpResponseC
odec.java:80)
    at
org.openqa.selenium.remote.http.AbstractHttpResponseCodec.decode(AbstractHttpResponseC
odec.java:44)
    at
org.openqa.selenium.remote.HttpCommandExecutor.execute(HttpCommandExecutor.java:158)
    at org.openqa.selenium.remote.RemoteWebDriver.execute(RemoteWebDriver.java:548)
    at org.openqa.selenium.remote.RemoteWebElement.execute(RemoteWebElement.java:276)
    at org.openqa.selenium.remote.RemoteWebElement.click(RemoteWebElement.java:83)
    at org.ofbizextra.ofbswd.test.SimpleTestSuite.click(SimpleTestSuite.java:540) ①
    at
org.ofbizextra.ofbswd.test.ExampleVuejsTestSuite.simpleTest(ExampleVuejsTestSuite.java
:34)  ①
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    ...
```

With on top the last method which generated error and down method which call the previous one (show in previous line).

① Usually it's needed to look only to org.ofbizextra.xxx method (xxx as the component of the failed test)

In this example, it's possible to say that simpleTest call a click which generated error (on a functional point of view)
on a technical point of view, simpleTest is in file ExampleVuejsTestSuite.java and the click call is in line 34 and click method is in file SimpleTestSuite.java and error is generated at line 540 by calling ....

6.1.4. Read Javadoc

Javadoc can be read here
You can go directly to the desired method javadoc by using Index button in navigation bar (menu on top) and selected it.

If javadoc not work, it's necessary to run a job with the javadoc option selected

In a test, when some data are used (offerId, login, ...), usually it's mentioned in the javadoc by giving data-obj (cf next chapter) used with notation like xxxxx.yyyy with xxxx the test-case and yyyy data-obj.

6.1.5. Look Test Log

In the main screen of the error message and the stack trace, under the stack Trace there is "Sortie Standard" section. In the "Gradle Selenium Result" panel there is a button "Standard Output" to go directly inside.
This is the log about the test with messages from test developers.
When there are multiple tests on the same java class file, it's necessary to find the part of log which concern the test you analyzed.

To found starting log about the test you search for, search with test name

```
: **** Start test : FileName.TestName
```

This log should be understandable, including the "Business Analysts". Each time the log is used to navigate in the test and analyze data and results, developers should, in the same time, enhance it with all elements missing.

6.1.6. Data File Used by Test

All data used to enter or check data during tests are (or should be) in external file. Most of time there is one file by package, it's only a convention not technically mandatory. Data files are XML and in each file, there are 4 levels :

1. scenario ⇐ scenario name is used (in selenium.properties) when selenium is run to choose the set of data to used
2. test-suite ⇐ test-suite name must be exactly the same as java testSuite class name
3. test-case ⇐ test-case name must be exactly the same as java testCase method name
4. data-obj (which could be contain some data-obj) ⇐ is a functional (or business) object

5. data (string, indexed, date, integer, double)

```
  <?xml version="1.0" encoding="UTF-8"?>
  <testdata xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="http://repository.ofbizextra.org/dtd/testcasedata.xsd">
      <version>1.0</version>
      <scenario name="job-jenkin"> <!-- a set of data used by selenium jobs -->
          <test-suite name="TestSuiteName">
              <test-case name="testMethod">
                  <data-obj name="objet"> <!-- a business object : a order, a user,
...  -->
                      <string name="maChaine" value="the value"/>
                  </data-obj>
              </test-case>
          <test-suite>
      </scenario>
  </testdata>
```

Data file are in the directory test-data in the project OfbSwd.

A lot of ID are generated by OFBiz (partyId, OrderId, ...), so to retrieve generated id during a test and be able to visualize them in application (if simple searching criteria is not possible), it's necessary to read test log, generally generated Id used after it creation is printed in the log.

6.1.7. Goto Website tested

Multiple testing environment are used, at least 1 per OFBiz release :

- 13.07 : https://ofbiz13-07-selenium.ofbizextra.org/example/control/main
- trunk on ofbizextra : https://ofbiz-selenium.ofbizextra.org/example/control/main
- trunk on Apache OFBiz demo : https://demo-trunk.ofbiz.apache.org/example/control/main