

SELENIUM TESTING WITH WEBDRIVER AND GRID 2.0 FOR EXTJS BASED PROJECTS

By Claude Gauthier

TABLE OF CONTENTS

SELENIUM TESTING WITH WEBDRIVER AND GRID 2.0 FOR EXTJS BASED PROJECTS	1
TABLE OF CONTENTS.....	1
BACKGROUND.....	2
ASSUMPTIONS.....	2
PRE-REQUISITES	2
LEARNING DOCUMENTATION	2
REQUIREMENTS.....	3
PROJECT FILE DOWNLOADS	3
INSTALLATION INSTRUCTIONS.....	4
<i>TestNg</i>	4
<i>Chrome Server Driver and IE Server Driver for WebDriver</i>	4
OVERALL STEPS TO RUN THE TESTSETUP PROJECT	4
<i>Expectations</i>	5
ABOUT SELENIUM GRID'S HUB AND NODE.....	5
<i>What is a hub?</i>	5
<i>What is a node?</i>	5
<i>Selenium and the Grid</i>	5
<i>Notes on Hub and Node(s)</i>	5
TESTSETUP SUITE	10
ABOUT TEXTEXTJS42 CLASS.....	11
<i>Background on ExtJS 4.2</i>	11
<i>Why ExtJS 4.2 is hard to test?</i>	11
<i>Code Examples</i>	11
LINKS AND RESOURCES	14
LINKS.....	14
BOOKS	14
CONTACT INFORMATION.....	14

BACKGROUND

The purpose of this document is to setup and execute functional tests using Selenium WebDriver, having the test execute in a controlled fashion, as well as in parallel using Selenium Grid.

Finally, the tests are working against web projects which are written using Sencha's ExtJS 4.2 framework.

ASSUMPTIONS

This documentation is written based on the following assumptions:

- Using a Windows operating system for development of the tests
- Using a Windows operating system for executing the tests
- Using the Selenium Standalone Server version 2.35.0

PRE-REQUISITES

This documentation assumes the following pre-requisites:

- Knowledge of Java
- Knowledge of the Eclipse IDE

LEARNING DOCUMENTATION

The following are suggested material for either reading and/or viewing:

1. Simplified Selenium (www.simplifiedselenium.com)
This documentation serves many purposes such as:
 - a. Provides details instructions on the installation of Eclipse, Java, Selenium and other technologies
 - b. Provides a good background into elements which are important to Selenium testing either as concepts and/or with practical examples and exercises on many topics including: CSS Selectors, XPath, HTML, Ant, etc.
2. Video Tutorials at <http://qtpselenium.com>.
These videos are worth the time to peruse as they offer practical information on the following topics: Eclipse, Java, Reflection API, Log4j, Excel POI, Java Properties, TestNg with Ant, Selenium IDE, Selenium RC, Selenium Web Driver, Grid and much more.

The access to all the videos, you must become a member. The cost for 1 single person for a lifetime access (at the time of writing this documentation) is 250\$ (USD).

Note: QtpSelenium tracks its users' IP address, so, they will ban a user if the access id/password is shared. It is recommended via the website's "contact trainer" link that you negotiate a cost for multiple access, should you choose to use their product video products.

REQUIREMENTS

You will require the following:

- Windows 7 system
- Firefox
- Firebug (via Firefox add-ons) (to be used for inspecting a webpage)
- FirePath (via Firefox add-ons) (to be used to either retrieve xpath/css selectors and or testing them)
- Illuminations for FireBug (via: <http://www.illuminations-for-developers.com/>)
this plug-in has a cost, but it is well worth it as it explores an ExtJS project via its components
- Chrome
- IE
- Java
- Ant
- Eclipse
- TestNg (this unit testing framework is more compatible with Grid 2 than JUnit)
<http://testng.org/doc/download.html>
- Selenium Server Standalone (contains Selenium, WebDriver and Grid 2)
(<https://code.google.com/p/selenium/downloads/list>) version 2.35.0
- Chrome Web Driver (<https://code.google.com/p/selenium/downloads/list>)
- IE Web Web Driver (<https://code.google.com/p/selenium/downloads/list>) (32 and 64 Bits)

PROJECT FILE DOWNLOADS

A project file is available for download, which contains a fully functional Selenium Test Suite with 4 tests, one of them is for an ExtJS 4.2 project which will also be available for download as well as through a URL for testing purposes.

All files are zipped.

- Link to TestSuite project file
<http://www.claudegauthier.net/downloads/TestSetup.zip> (107 mb)
This TestSuite contains everything you need, all the jars are included as well as the drivers, etc.
- Link to ExtJS 4.2 project file
http://www.claudegauthier.net/downloads/extjs42_selenium.zip (27 mb)
This is the full project, the library is included as well as the build
The URL to view the project is: http://www.claudegauthier.net/demos/extjs42_selenium/
- Link to Prioritizer project (used to help set a priority on test case being executed)
<http://www.claudegauthier.net/downloads/Prioritizer.zip> (30 mb)
(the class for this project is merged with the selenium server standalone jar)
- Link to Selenium Grid JSON files and batch files (these help you start the hub and the node(s))
http://www.claudegauthier.net/downloads/batch_jsons_and_others.zip (33 mb)

INSTALLATION INSTRUCTIONS

The “Simplified Selenium” contains the bulk of the instructions when it comes to Java, Firefox, Firebug, Eclipse, Ant, etc.

TESTNG

One thing of importance to mention is that instead of using JUnit for providing test support, we are using TestNg. To install it follow these simple steps.

1. Open Eclipse (assuming version 3.4 and above)
2. Under Help, click Install New Software
3. Click the Add button
4. For Name: type TestNG
5. For Location input this: <http://beust.com/eclipse>
6. Click OK
7. Select the checkbox with the name TestNG and click next
8. follow the instructions, accept everything
9. You will more than likely have to restart Eclipse

CHROME SERVER DRIVER AND IE SERVER DRIVER FOR WEBDRIVER

WebDriver doesn't include the drivers for Chrome and IE in the Selenium Standalone Server jar file as they are considered external drivers.

Although there are included in the TestSetup project, if you need to download them:

<https://code.google.com/p/selenium/downloads/list>

IEDRIVER SERVER WARNINGS

Note: if you are testing on a 64 bits Windows machine and especially with IE10 and find that performance is extremely sluggish, it is recommended that you switch to the 32 bits version of the driver.

Note: on Windows, the IEDriverServer (both 32 and 64 bits) leave their process open even when you close the driver. This process is attached to a conhost process (your command window), so remember to open your Task Manager and clean up from time to time

When you delete the IEDriverServer process, the conhost process associated with it will also close.

OVERALL STEPS TO RUN THE TESTSETUP PROJECT

- The project is configured to run 4 tests in parallel for 3 browsers (Chrome, IE and Firefox)
- Each browser is associated to a node via a port
- an XML file (testing.xml) will be executing via Eclipse (or you can setup an Ant file to start from command prompt, you will use the testing.xml file in your Ant)
- Eclipse will require you to install TestNg (one time, this is a permanent install)
- You will then add the TestSetup Project to Eclipse
- Using the batch files, you will start the hub first, then 3 nodes (nodeIE.bat, nodeFirefox.bat and nodeChrome.bat). They are already configured via their .json file counterpart.
- In Eclipse you will right click on testing.xml and run as TestNg

EXPECTATIONS

- It takes time, probably 2 to 5 minutes, do NOT use your system as you are running tests for Selenium can fail on you
- You will see browsers opening and running tests
- When all is done, Eclipse's console will have an output of the test
- Log4j will also have created a log file in the log folder
- TestNG will have a report ready for you which can be opened from the test-output folder you can use emailable-report.html or index.html (in IE, you must install an SVG plug-in to view the index.html report)

ABOUT SELENIUM GRID'S HUB AND NODE

The following are some notes that I've taken which will help you understand how it works.

The instructions are based on the use of Selenium Server Standalone 2.35.0 jar file. Using other versions DOES NOT guarantee either forward and/or backward compatibility.

Instructions are based on selenium-server-standalone-2.35.0.jar

WHAT IS A HUB?

- The hub is the central location which will be dispatching the tests.
- Tests are dispatched node(s)
- You can have multiple nodes linked to a hub.
- A hub will offer a URL for you to view its console.
- You can start a hub as follows:
- `java -jar selenium-server-standalone-2.35.0.jar -role hub`
- By default, your hub will be available via <http://localhost:4444> (for the console)
- For each node that will connect to the hub the link will be to set to <http://localhost:4444/grid/register>

WHAT IS A NODE?

- The node represents a link to point for executing tests (using hostname, host and port).
- Each node can be configured to have multiple browser instances available
- Each node can have 1 or more instances of browsers (defaults to 3 browsers firefox,chrome,ie)
- Each node can have a maximum of browser instances operating (maxSession)
- Each node's browser can be configured to be opened for a maximum amount of instances (maxInstances)

SELENIUM AND THE GRID

Note: when working with grid, your code must use the RemoteWebDriver in your Java code.

NOTES ON HUB AND NODE(S)

START THE HUB

```
java -jar selenium-server-standalone-2.35.0.jar -role hub
```

ACCESS TO THE HUB'S CONSOLE

by default the port is 4444 and the URL will be `http://localhost:4444`

START A NODE

Start a node (set as many nodes as you want, but each node must have a unique port)

```
java -jar selenium-server-standalone-2.35.0.jar -role node -hub
http://localhost:4444/grid/register -port 5556
```

When you start a node this way, you will have access to 3 browsers (Chrome, IE, Firefox)

LIMIT A NODE TO A SINGLE BROWSER

Have a node work with only 1 browser (for example firefox):

Instruction: append `>> -browser browserName=firefox`

```
java -jar selenium-server-standalone-2.35.0.jar -role node -hub
http://localhost:4444/grid/register -port 5556 -browser browserName=firefox
```

SET MULTIPLE BROWSERS TO A NODE

Have a node work with only multiple browser (for example firefox, chrome and ie):

Instruction: append `>> -browser browserName=firefox -browser browserName=chrome -browser browserName=iexplore`

```
java -jar selenium-server-standalone-2.35.0.jar -role node -hub
http://localhost:4444/grid/register -port 5556 -browser browserName=firefox -browser
browserName=chrome -browser browserName=iexplore
```

SET THE MAXIMUM OF INSTANCES EACH BROWSER CAN BE INSTANTIATED

For example:

- `browserName=firefox,maxInstances=2`
- `browserName=chrome,maxInstances=2`
- `browserName=iexplore,maxInstances=2`

```
java -jar selenium-server-standalone-2.35.0.jar -role node
-hub http://localhost:4444/grid/register -port 5556
-browser browserName=firefox,maxInstances=2
-browser browserName=chrome,maxInstances=2 -browser
browserName=iexplore,maxInstances=2
```

SET THE MAXIMUM AMOUNT OF CONCURRENT BROWSERS THAT A HUB WILL ALLOW FOR NODES TO EXECUTE AT ANY GIVEN TIME

Even though you can have a total of 2 instances per browsers as per the previous example, the amount of concurrent tests is determined globally by -maxSession

For 3 concurrent instances to run, "-maxSession 3" can be appended

This means only 3 instances of browsers can be opened at any given time

```
java -jar selenium-server-standalone-2.35.0.jar -role node -hub
http://localhost:4444/grid/register -port 5556
-browser browserName=firefox,maxInstances=2
-browser browserName=chrome,maxInstances=2 -browser
browserName=iexplore,maxInstances=2
-maxSession 3
```

START A NODE ON ANOTHER PORT (DEFAULT TO 5556)

- port XXXX

```
java -jar selenium-server-standalone-2.35.0.jar -role node -hub
http://localhost:4444/grid/register
-port 5557 -browser browserName=firefox,maxInstances=2 -browser
browserName=chrome,maxInstances=2 -browser browserName=iexplore,maxInstances=2
-maxSession 3
```

EXTERNAL DRIVERS

Chrome and IE require external drivers (assume a drivers folder is in the same directory as your selenium-server-standalone-2.35.0.jar file)

- -Dwebdriver.chrome.driver=c:\selenium\drivers\chromedriver.exe
- -Dwebdriver.ie.driver=c:\selenium\drivers\IEDriverServer32.exe

```
java -Dwebdriver.ie.driver=c:\selenium\drivers\IEDriverServer32.exe -
Dwebdriver.chrome.driver=c:\selenium\drivers\chromedriver.exe -jar selenium-server-
standalone-2.35.0.jar -role node -hub http://localhost:4444/grid/register -port 5556 -
browser browserName=firefox,maxInstances=1 -browser browserName=chrome,maxInstances=1
-browser browserName=iexplore,maxInstances=1 -maxSession 3
```

RECOMMENDED STRATEGY FOR HUB/NODE

With the current version of Selenium Grid/WebDriver(2.35.0), it is best to work with the assumption of 1 node associated to 1 browser.

As you will see in the TestSuite project, the code is setup this way to facilitate control.

Although batch files and configuration files have been provided, the following is an example of 1 hub with 3 nodes

- 1) start your hub
`java -jar selenium-server-standalone-2.35.0.jar -role hub`
- 2) for each node, open a command prompt
- 3) for firefox node on port 5556
`java -jar selenium-server-standalone-2.35.0.jar -role node -hub
http://localhost:4444/grid/register -port 5556 -browser
browserName=firefox,maxInstances=1 -maxSession 3`
- 4) for chrome node on port 5557
`java -Dwebdriver.chrome.driver=c:\selenium\drivers\chromedriver.exe -jar
selenium-server-standalone-2.35.0.jar -role node -hub
http://localhost:4444/grid/register -port 5557 -browser
browserName=chrome,maxInstances=1 -maxSession 3`
- 5) for iexplore node on port 5558
`java -Dwebdriver.ie.driver=c:\selenium\drivers\IEDriverServer32.exe -jar
selenium-server-standalone-2.35.0.jar -role node -hub
http://localhost:4444/grid/register -port 5558 -browser
browserName=ieexplore,maxInstances=1 -maxSession 3`

Nevertheless, Selenium Grid doesn't allow you to control what gets executed in which order, but by using the Testing.xml file, and setting each test in the order we want and by setting a priority within each test class file, we can achieve this.

This was accomplished with the help of the Prioritizer class, which was merged into the selenium-server-standalone-2.35.0.jar file.

PRIORITIZER CLASS

The Prioritizer project is available for inspection, but it is a simple class, that if you wish to use it, and you upgrade to a new version of Selenium, you will have to merge it.

This is how you can merge it, assuming you copy the build's com folder where the selenium jar file is located:

- 1) copy the com folder of the Prioritizer project to the folder containing the selenium server standalone jar file
- 2) ensure you have Java with the SDK installed and that the /bin path of your JDK is in your environment settings under Path
- 3) at the command prompt: `jar uf selenium-server-standalone-2.35.0.jar
com/prioritizer/testcases/TestPrioritizer.class`

SETTING A UNIQUE HOST TO A NODE

`add - host 192.168.10.7`

START A HUB WITH JSON CONFIG

```
java -jar selenium-server-standalone-2.35.0.jar -role hub -hubConfig hub.json
```

```
Sample of hub.json {
  "host": null,
  "port": 4444,
  "newSessionWaitTimeout": -1,
  "servlets": [],
  "prioritizer": null,
  "capabilityMatcher": "org.openqa.grid.internal.utils.DefaultCapabilityMatcher",
  "throwOnCapabilityNotPresent": true,
  "nodePolling": 1000,
  "cleanUpCycle": 5000,
  "timeout": 30000,
  "maxSession": 6
}
```

START A NODE WITH JSON CONFIG

```
java -Dwebdriver.ie.driver=c:\selenium\drivers\IEDriverServer32.exe -jar selenium-server-standalone-2.35.0.jar -role node -nodeConfig nodeIE.json
```

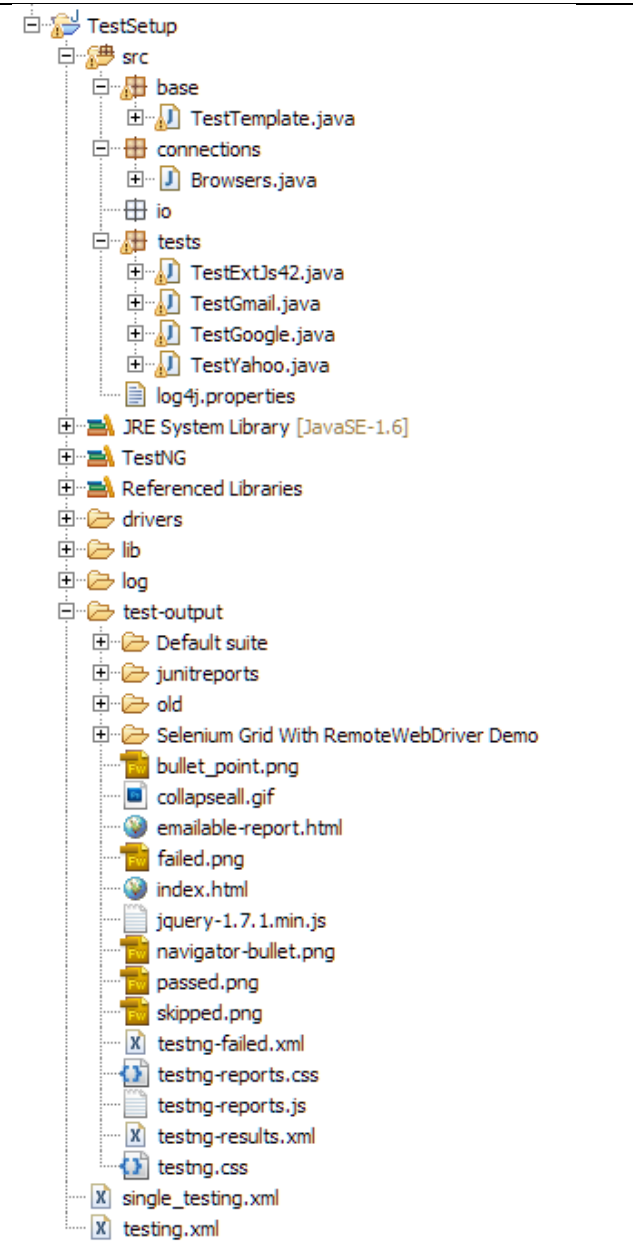
```
Sample of a node configured for IE {
  "capabilities": [
    {
      "browserName": "iexplore",
      "maxInstances": 1
    }
  ],
  "configuration": {
    "nodeTimeout": 120,
    "port": 5558,
    "hubPort": 4444,
    "hubHost": "127.0.0.1",
    "nodePolling": 2000,
    "registerCycle": 10000,
    "register": true,
    "cleanUpCycle": 2000,
    "maxSession": 3
  }
}
```

PRIORITIZER ADDENDUM

- 1) hubConfig needs "prioritizer": "com.prioritizer.testcases.TestPrioritizer"
- 2) DesiredCapabilities must be set with `cp.setCapability("_important", testPriority)`; (testPriority is a value 1, 2, etc.. as string) the higher the String number, the more important it is, thus tests are executed in a descending order.

TESTSETUP SUITE

The TestSetup suite is comprised of the following files:

	<p>base.TestTemplate.java is a class which all tests must be derived from.</p> <p>connections.Browsers.java is a class which is used by base.TestTemplate.java. It basically works with a dataProvider to loop through each supported browser (as listed in getBrowsers()) and then instantiated in a loop in getBrowserDriverInstance()</p> <p>As each class is instantiated, the class' constructor set a testPriority which is consumed in getBrowserDriverInstance();</p> <p>In the tests package, we have 4 test cases Each test inherits from TestTemplate Each test's should have its constructor setting a priority for the test, if it doesn't then the default value is "1".</p> <p>TestTemplate has a "test" method which has to be overridden. The first line of the overridden method should be a call to the super. The signature is (int index, String browser) as determined by the dataProvider and this is passed into the super. After that, each test should be a separate public function.</p> <p>You will notice a log4j.properties. When test are executing, a log file is created and available in the log folder.</p> <p>To execute the test, you should be using testing.xml.</p> <p>It will create a test-output. index.html and emailable-report.html are 2 entry points to view the reports. If opened with IE, you have to install an SVG plugin.</p>
--	---

ABOUT THE TESTEXTJS42 CLASS

Although there are 4 classes in the TestSetup project, the one which is most important is the TestExtJs42.java file.

Selenium is not the best framework to test ExtJS, because ExtJS doesn't build webpages in a way which Selenium is meant to work with.

Better tools are available such as Siesta (<http://www.bryntum.com/products/siesta/>)

But for those of us who must work with Selenium, we must face the difficult challenge of dealing with ExtJS 4.x

BACKGROUND ON EXTJS 4.2

At the moment of writing this document, ExtJS 4.2 is used as the javascript framework used to build the projects which are to be tested using Selenium.

WHY EXTJS 4.2 IS HARD TO TEST?

1. ExtJS 4.x uses a splitDOM engine to generate HTML code to the browser. The generated code is optimized based on the target browser, so, the actual HTML code generated for each browser can be different. In a nutshell, you cannot expect your HTML to be the same between IE and Firefox for example.

This means that you can't rely on XPath with 100% certainty.

2. Sencha's SMVC does NOT rely on ExtJS components having unique IDs and as such, this means that the generated code will not have any predictable unique IDs to work with. So, the best way to deal with this is with the use of WebDriver's JavascriptExecutor. But even that, as you will see below, can be a tricky thing, especially when it comes to handling events. At best, you will try and use the JavascriptExecutor to run command such as Component Queries.

CODE EXAMPLES

- 1) setting the value of a textfield

```
StringBuilder s = new StringBuilder();
s.append("with (Ext.ComponentQuery.query(\"textfield[name='username']\")[0]) {
setValue(arguments[0]); fireEvent('blur'); }");
((JavascriptExecutor) driver).executeScript(s.toString(), username);
```

As you will see from the above sample, setValue() won't work unless you blur from the field.

2) clicking an ExtJS button

In the case of buttons, you will find that events do NOT fire as expect.

First, let me show you how we are able to use a ComponentQuery to find a button and change its text.

```
s = new StringBuilder();
s.append("with (Ext.ComponentQuery.query(\"button[itemId='signInButton']\")[0]) { focus();
setText('new Signing Text'); }"); // this works
((JavascriptExecutor) driver).executeScript(s.toString());
```

Using a similar code structure, we now try to fire the click event on that captured button.

```
s = new StringBuilder();
s.append("with (Ext.ComponentQuery.query(\"button[itemId='signInButton']\")[0]) { focus();
fireEvent('click'); }"); // this doesn't work
((JavascriptExecutor) driver).executeScript(s.toString());
```

The above will not work, and causes the test to crash.

The alternative is to use findElements and/or findElement on a page to old fashion way; Inspecting the code.

But be careful. ExtJS HTML/Dom delivery isn't identical between browsers. At first, try CSS selectors. The reason is simple. 1) Performance. CSS Selectors are faster than XPath. 2) IE and Chrome are very slow on XPath. Avoid it if you can.

The following snippet is how we can get a button to fire under ExtJS, it's not pretty.

```
ArrayList<WebElement> buttons = (ArrayList) driver.findElements(By.cssSelector("a[class^='x-
btn']"));
System.out.println(String.format("***** Found %d buttons ***** ", buttons.size()));
String bText = "";
for(WebElement button : buttons) {
    bText = button.getText();
    if(bText.equals("Sign In")) { // "Sign In" would have to be part of a dictionary
        button.click();
        break;
    }
}
}
```

The reason this was used is that when you inspect the HTML code, you will find that there was nothing directly identifying the ExtJS component.

Here's the code which creates the button in ExtJS

Code in ExtJS for the Button	HTML Generated
<pre> { xtype: 'button', text : "Sign In", action : 'submit', itemId : 'signInButton', margin : '0 33 0 0', formBind : true, disabled : true } </pre>	<pre> Sign In </pre>

When you compare the 2 of them, there is NOTHING that could be used to determine EXACTLY the button we want to use.

Of course, there is something that can be done, but it requires modifying your ExtJS code to add a CSS attribute which would be unique to this element as displayed below.

```

{
    xtype: 'button',
    text : "Sign In",
    action : 'submit',
    cls: 'signInButtonCls',
    itemId : 'signInButton',
    margin : '0 33 0 0',
    formBind : true,
    disabled : true
}

```

LINKS AND RESOURCES

Here are links and resources which can be useful for setting up Selenium and the functional tests.

LINKS

- www.sencha.com: access the API documentation to learn more about ExtJS
- www.qtp.selenium.com: hundreds of hours of videos on pertinent Selenium topics which are worth the investment in time and money. Some of them are free, but the ones you really need are not. When you join them, they have a Google group which trainers, consultants and students get to collaborate and exchange ideas and solutions.
- <http://pages.cs.wisc.edu/~hasti/cs368/javaTutorial/NOTES/Exceptions.html>: Java Exceptions
- <http://selenium.googlecode.com/svn/trunk/docs/api/java/index.html>: Web Driver API doc
- <https://poi.apache.org/>: if you wish to use Excel to keep your parameters and/or your automation, this is the resource.
- <http://poi.apache.org/spreadsheet/quick-guide.html>: for POI, the various recipes to be able to read/write into an Excel Sheet with Java
- <http://logging.apache.org/log4j/1.2/download.html>: the Logger tool
- <http://testng.org/doc/index.html>: TestNg, easy to read
- <http://selenium.googlecode.com/svn/trunk/docs/api/java/index.html>: Selenium API
- <http://docs.seleniumhq.org/download/>: selenium downloads
- <http://thecancerus.com/testing-extjs-application-with-selenium-few-pointers/>: ExtJS/Selenium pointers.

BOOKS

- Mastering ExtJS: currently at version 4.2, by Loiane Groner from PacktPub publisher. This book would get anyone up to speed on every important aspect of ExtJS, assuming no knowledge of coding.
- Simplified Selenium: version 2.0, it is a book as well as a PDF, it is worth the purchase and is a pre-requisite to this documentation.

CONTACT INFORMATION

For further information, I can be reached at claude_r_gauthier@hotmail.com.