# **PAF Selenium Framework Overview**

Contents

[**PAF Selenium Framework Overview** 1](#_Toc367196915)

[Document version 1](#_Toc367196916)

# Document version

* 1.00: 20130917 Alter.hu. Document created.

In this document I will give you the detail guideline to build up the Selenium framework and how to run the script.

* Prerequisite resources we need

1, Download the Selenium Server standalone file from this URL: [https://code.google.com/p/selenium/downloads/detail?name=selenium-server-standalone-2.35.0.jar&can=2&q=, if](https://code.google.com/p/selenium/downloads/detail?name=selenium-server-standalone-2.35.0.jar&can=2&q=,%20if) you are the Automation developer, it’s better to download the Selenium source code here: <https://code.google.com/p/selenium/downloads/detail?name=selenium-java-2.35.0.zip&can=2&q>=

2, Download the IE and chrome driver (which is used for supporting running script in IE or chrome, Selenium can run in Firefox Seamlessly, so we no need the Firefox driver here):

IE Driver (32 bit IE):<https://code.google.com/p/selenium/downloads/detail?name=IEDriverServer_Win32_2.35.3.zip&can=2&q>=

IE Driver (64 bit IE): <https://code.google.com/p/selenium/downloads/detail?name=IEDriverServer_x64_2.35.3.zip&can=2&q>=

Chrome Driver: <https://code.google.com/p/chromedriver/downloads/detail?name=chromedriver_win32_2.3.zip&can=2&q>=

3,Downalod the Ant JAR file ,as we wrapped the Selenium with Ant now ,which is used for the test case or test suite feature in Selenium .you can download from here: <http://testng.org/testng-6.8.zip>

4, Download the log4j file, which we used for outputting the script report, it’s very helpful to see the script execution flow and to debug the script. you can download it from here : <http://www.apache.org/dyn/closer.cgi/logging/log4j/1.2.17/log4j-1.2.17.zip> and <http://www.apache.org/dyn/closer.cgi/logging/log4j/companions/extras/1.1/apache-log4j-extras-1.1.zip>

5, Download the JMail jar file, which is used for sending the email. You can download it from here: <http://java.net/projects/javamail/downloads/download/javax.mail.jar>

6, Download the java excel API file, which is used for using the excel file to manager our test data: <http://sourceforge.net/projects/jexcelapi/files/jexcelapi/2.6.12/jexcelapi_2_6_12.zip/download>

7, Download the jsoup jar file, which is used for parsing the html content, as currently we used the template html content to generate a Selenium execution report. You can download it from here: <http://jsoup.org/packages/jsoup-1.7.2.jar>

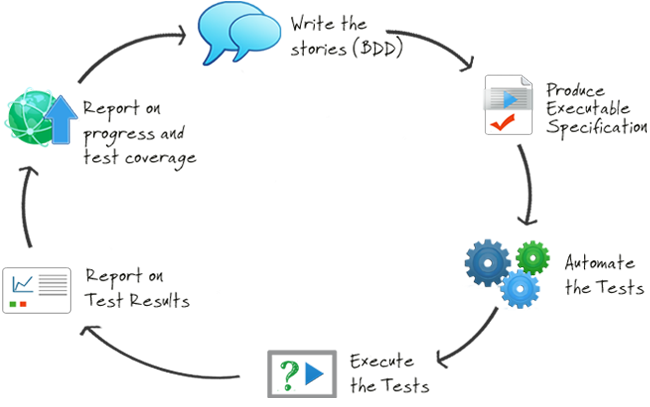
8, Download the jfreechart jar file, which is used for generating our automation overview execution graph: <http://sourceforge.net/projects/jfreechart/files/latest/download?source=files>

9, Download the AutoIt supported file, which is used for testing Windows Application, as we know, Selenium is used for Web Automation testing. You can download it from here. <http://www.autoitscript.com/cgi-bin/getfile.pl?autoit3/autoit-v3.zip> ,

Jacob jar and DLL file from here: <http://sourceforge.net/projects/jacob-project/files/latest/download?source=files> .

10, Download the MySQL or oracle JDBC driver file. It’s optional.

* **PAF Framework structure Overview**



1. Source code catalog

This is the Selenium framework core code, there are three directories:

* 1. com.hp.pop

Which is the page object level, currently we used the Selenium Page Factory to initiate the page element .so it’s easy to collection and manager the UI object in page. Just like the object repository in QTP. You can find this pattern detail from here: <https://code.google.com/p/selenium/wiki/PageObjects>

<http://www.slideshare.net/dantebriones/using-the-page-object-pattern>

<http://www.thoughtworks.com/insights/blog/writing-twist-tests-page-object-pattern>

In automated web testing, a **Page Object** is a class or object that represents a web page in your application. A Page Object hides the technical details about how you interact with a web page behind a more readable and business-focused facade. This has two main advantages:

* The tests are much more readable
* Page access logic is centralized in on place, making maintenance much easier

When it comes to navigation with Page Objects, one popular strategy is to have methods that navigate to a new page return a Page Object for this page. For example, the clickSubmitBtn method listed about could be rewritten as follows:

// click the submit button

**public** **void** clickSubmitBtn() {

SeleniumCore.*highLight*(driver, submitbtn);

SeleniumCore.*clickElement*(driver, submitbtn);

*logger*.info("Click the Submit button to log in the home page");

}

If an operation does navigate to a new page, the Page Object simply returns the new page object like blow:

**public** Home\_Page logAsValidUser(String username, String inputpassword) {

String pagename=**this**.getClass().getName();

*logger*.info("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"+pagename+"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

**if** (isLoginPage()) {

*logger*.info("User now is logging the PAF,input username and password......");

typeEmail(username);

typePassword(inputpassword);

clickSubmitBtn();

}

//String teststr=(String) SeleniumCore.executeJS(driver, "return 'test is a noass';");

// String ps=(String) SeleniumCore.executeJS(driver, "var a=document.getElementById('overridelink');return a;");

//String ts=(String) SeleniumCore.executeJS(driver, "return document.getElementById('overridelink');");

*logger*.info("Had logged in the home page successfully......");

**return** PageFactory.*initElements*(driver, Home\_Page.**class**);

}

In this section we had created a base class called PageObject and all the other page object extends this class, so it can access the variable defined in base class easier.

* 1. ***com.hp.tests***

This directory stored all the test cases we used in project, we wrapped every simple business flow as a simple testNG class, and organized all these test case into testNG case suite. Then we can run a complete end to end process;

Every test needs to extend the BaseDriver class, so it can use the common feature defined in BaseDriver class;

* 1. ***com.hp.utility***

This directory stored many common reused functions, like using the AutoIt API, using database connection, using Excel parse API, using File operation, using time utility .g.tc

* 1. AutoItXUtils.java: the core function for using AutoIt framework;
  2. BaseDriver.java: the core Selenium test function ,every Selenium test case need to extend this function ,it will setup our Selenium server and capture the error screenshot when we met error ,also it will do a recovery if meeting the run error exception at the run time;
  3. DatabaseUtils.java: this is used for connecting the database, currently now wrapped it for MySQL database API;
  4. ExcelUtils.java: this is the core excel operation API, which we can use to access, read or write excel as we want;
  5. FileUtils.java: this is the core file operation API, we can use it to read, modify or write content in file;
  6. FirefoxProfileFile.java: this is the Firefox configuration file, we can use it to configure Firefox to bypass the certificate error or download the file automatically without pop dialog;
  7. HostUtils.java: this is the host operation API, we can use it to get the host name, host IP address or the OS version .etc.
  8. JsoupUtils.java: this is the core API used to parse the html document, we will use it to compile the PAF report in future;
  9. RecoveryScenario.java: this is the recovery file, you can define any function you need here to recover when you run the script, and it’s very useful;
  10. RetryFail.java, RetryListener.java: this is the function used to listen the testNG execution status, if the test failed, we will retry to run the test with the specified times;
  11. SeleniumCore.java: this is the Selenium core Wrapped API, you can access its function to click or refresh or pop the data in page ,here we recommend you not use the Selenium internal API directly ,using these functions will be useful to debug the code;
  12. TimeUtils.java: this is about the time utility API, you can use it to count the time between a time interval, or you can get the formatted timestamp as you want;

1. **Associated libraries catalog**

The libraries we used here need to add to the class path in our project.

1. **Associated resource catalog**

AutoItX3.dll: this is the AutoIt API file, you need to register this DLL file in your execution host, run the command in windows like blow: regsvr32 AutoItX3.dll;

Automation\_Overview\_Template.htm: this is the template report file, which used for the email report we will generate, the report will be like blow:

chromedriver.exe: this is the chrome driver we had downloaded before, which is used for the chrome browser testing;

IEDriverServer.exe: this is the IE driver we had downloaded before, which is used for the IE browser testing;

logo.png: this is the company logo file, which we used this for the email content can contain the company logo;

Remember\_certificate\_exception-1.0.0-fx.xpi: this used to bypass the security certificate error in Firefox, as now we had developed code to bypass the exception error, so you need to care this file;

report\_template.htm: this is the daily email report template file, every day the execution report is based on this template file content;

RunSeleniumServer\_Hub\_Step1.bat, RunSeleniumServer\_Node\_Step2.bat: this is the most important executable file we need to run in the execution host. Firstly you need to run the RunSeleniumServer\_Hub\_Step1.bat file and then run the RunSeleniumServer\_Node\_Step2.bat file.

selenium\_error.txt: this is just the selenium error collection file, in this file I collection some errors we met in selenium develop. And give some solution for them.

TestData - Editor.xls, TestData.xls: this is the data driver file ,all the testing data we stored in this excel file ,firstly you can modify or update the test data in TestData - Editor.xls as you want anytime ,and then send CTRL+Q hotkey in excel ,it will copy all the contents in TestData - Editor.xls into TestData.xls. This can prevent the excel conflict when opening the excel at the same time;

1. **device data catalog**

This directory had stored all the SN/PN files we need to test in the PAF project.

testdevice.xlsx; in this file stored the SN/PN data;

1. **Ant and TestNG file**

build\_localhost.xml: this is a basic Ant Script which we used in Jenkins jobs to run the Ant Script, including these operation:

* Clear the execution host environment and keep the testing environment purely.
* Compile all the java source code to class file ;
* Start the Selenium server in the host ,this had not been used now;
* Run the testNG suite;
* Stop the Selenium Server ,this is also not been used now yet;

testng\_localhost.xml: this is the test suite we need to run, including the testNG tests;

* **How to run the script**

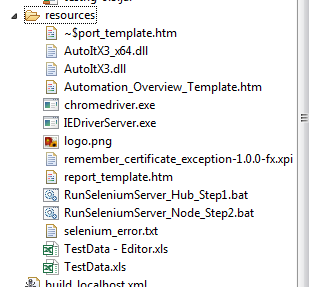
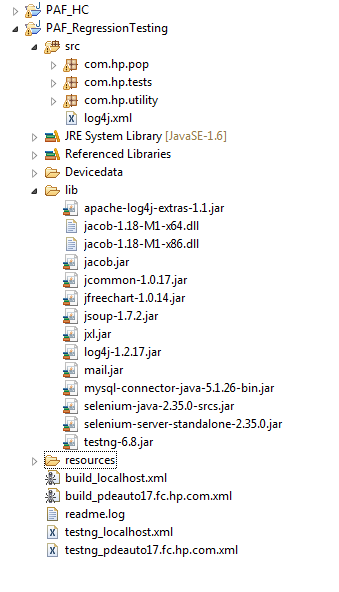
Run script in local host

Firstly, you need to start the Selenium Server in your host .then you just start the Jenkins job, it will run the test as you pointed here;

1. Run Script in remote host

Firstly, you also need to start the Selenium Server in the execution host, then you just start the Jenkins job, it will run the test as you pointed here;

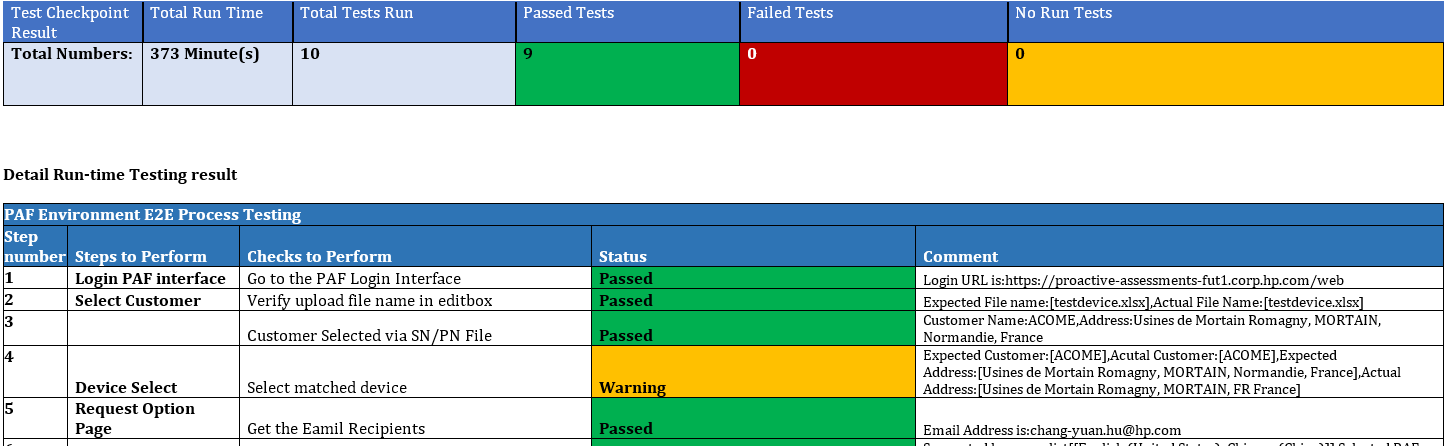
***3 The eclipse whole project structure like blow***



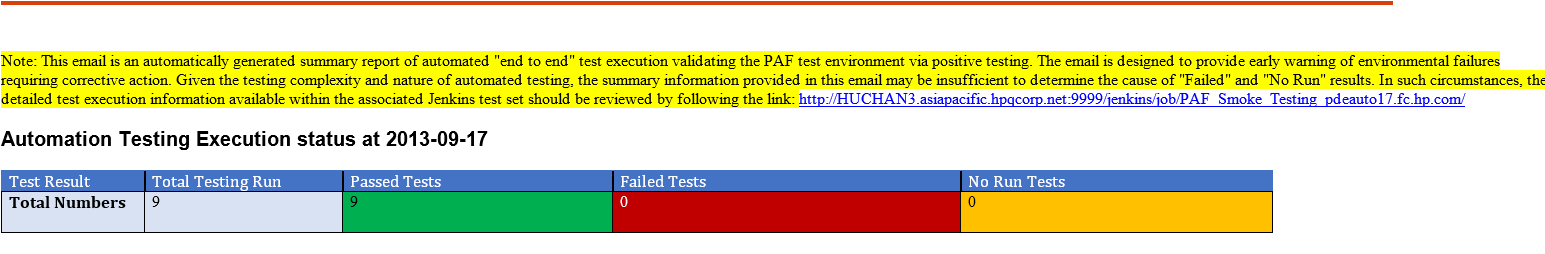
* **Automation execution Email report**

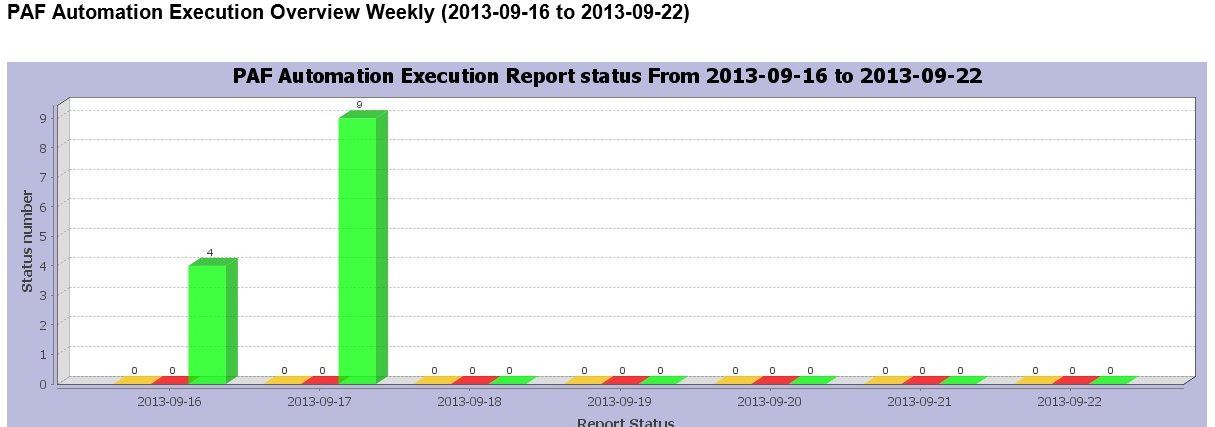
Now every time when we end up the automation execution, we will generate two reports, one is for the today’s execution report, another is for the PAF project overview execution report, contain many analysis graphs, like weekly report, monthly report and total project report.

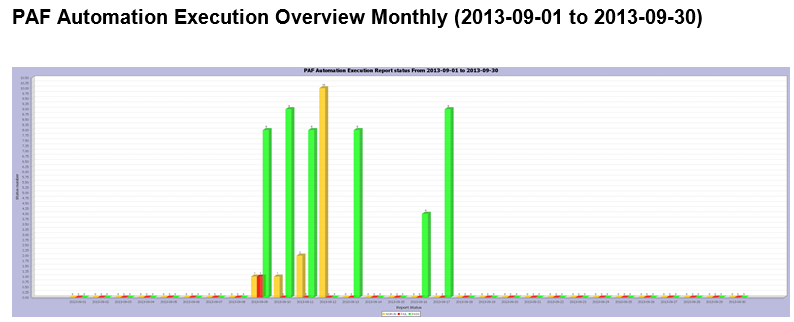
1. ***Everyday execution report be like blow:***



1. ***PAF whole project execution overview report be like blow:***







* **Selenium Tips**

1. In the testng.xml you need to change the “enabled” property in every test to make sure which test you need to run in this suite.
2. If the test failed ,you want to stop the test, you can use the dependonmethod for the TestNG tests, but if you want to continue the test ,you should better use another testNG test;
3. Using xpath or cssselector as possible as we can ;
4. It’s better to delete all the unnecessary folders in PAF eclipse workspace even though we had deleted all these in our ant script;’

* **Reference links**

<http://www.thucydides.info/>

<http://www.wakaleo.com/blog/selenium-2-webdriver-quick-tips-page-object-navigation-strategies>

<https://groups.google.com/forum/#!topic/testng-users/_F9geWyxUsI>

<http://hedleyproctor.com/2011/05/tutorial-writing-xpath-selectors-for-selenium-tests/>