Frameworks for Browser Automation Testing

Luís A. Bastião Silva

University of Aveiro
DETI / IEETA
bastiao@ua.pt

universidade de aveiro
deti departamento de electrónica, telecomunicações e informática
ieeta instituto de engenharia electrónica e telemática de aveiro

Outline

- Motivation
- Comparison between frameworks
- Nightwatch.js
  - How it works?
  - How to write tests?
  - How to run the tests?
- How to use in your project?
- Demo

What is Browse Automating Tests?

You can use it also to automate your daily tasks, if you are able to do it..

- Launch and automate your browser
- Fill and submit forms
- Click and follow links
- Verify if the output is according of what you define.

Motivation

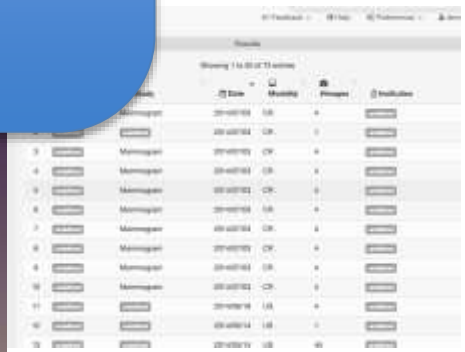
.. or in other words, why should I use Browse Automated Testing?

- Often web developers changes in the interface, change the code, we fix bugs and add new featur
- Some
- Brows y
- Test v

Software developers are humans and they made mistakes. Developers can type:

"id="buttob_searc" where should be "id="button_search" – and it breaks the functionality.

Browse automated tests can detect it!



Comparison between different frameworks

Name	Nightwatch.js	Selenium	Browsera	dalekjs	htmlunit	watin	sahipro	venusjs
OSS	Yes	Yes	Paid	Yes	Yes	Yes	Paid	Yes
Activity	High	High	Support	Medium	Low	Low	Support	Medium
WebDriver	Yes	Yes	Yes	Yes	No	No, IE only	Yes	Yes
Jenkins*	Medium	Medium	Automated tests, reports	Medium	Unknown	Unknown	Automated tests, reports	Medium
Output	Very good, JUnit	Very good, JUnit	Very good, UI Navigation	Good	Good	Unknown	Very good, UI Navigation	Unknown
Difficulty to write	Easy	Medium (syntax is not good)	Easy	Easy	Verbose	Easy	Easy---	Easy
Deployment	Medium	Medium	Easy	Medium	Medium	Easy	Easy	Medium
Observations	Selenium	-	-	Docs not good	-	-	-	Developed by LinkedIn

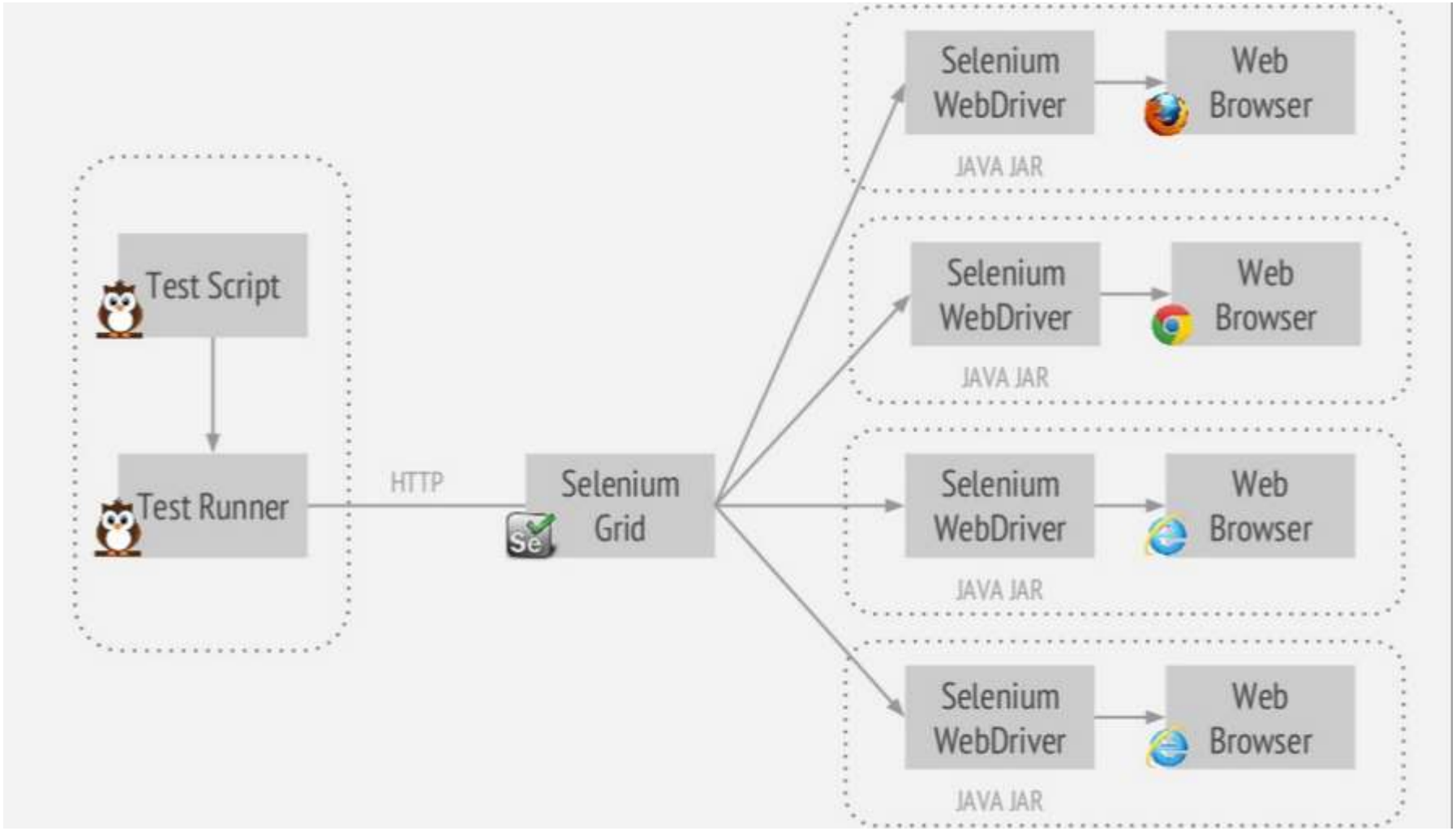
Feel free to contribute to this comparison: http://goo.gl/Vq3WeO

Why nightwatch.js?

- Easy to write tests
- Github activity
- Free
- Good documentation
- Supports IE (we know that no one cares, but in the end everybody cares.)
- It is working now. ☺
- More info:
  - ~10k downloads/month
  - ~150 forks (github)
  - ~2300 stars (github)

Nightwatch.js architecture

What are behind it?



From: http://nightwatchjs.org/guide#theory-of-operation and http://www.slideshare.net/sethmcl/join-the-darkside-nightwatchjs

Directory structure

How to start? What files should I have? How the test groups works?

- Install:
  - $ npm install nightwatch
  - Run Selenium server
- nightwatch.js
  - #!/usr/bin/env node
  - require('nightwatch/bin/runner.js');
- nightwatch.json
- Tests/
  - Login.js
  - Accounts/0001_landing_page.js
  - Accounts/0002_register.js
  - dashboard/0001_widgets.js
  - **Your own structure, you define it!**

Setup nightwatch.json

```
"selenium" : {
  "start_process" : false,
  "server_path" : "",
  "log_path" : "",
  "host" : "127.0.0.1",
  "port" : 4444,
  "cli_args" : {
    "webdriver.firefox.driver" : ""
  }
```

```
"src_folders" : ["tests"],
"output_folder" : "reports",
"custom_commands_path" : "",
"custom_assertions_path" : "",
"globals_path" : "",
```

```
"test_settings" : {
  "default" : {
    "launch_url" : "http://localhost:8000",
    "selenium_port"  : 4444,
    "selenium_host"  : "localhost",
    "silent": true,
    "screenshots" : {
      "enabled" : false,
      "path" : ""
    },
    "desiredCapabilities": {
      "browserName": "firefox",
      "javascriptEnabled": true,
      "acceptSslCerts": true
    }
  },
  "chrome" : {
    "desiredCapabilities": {
      "browserName": "chrome",
      "javascriptEnabled": true,
      "acceptSslCerts": true
    }
  }
```

Test sample workflow

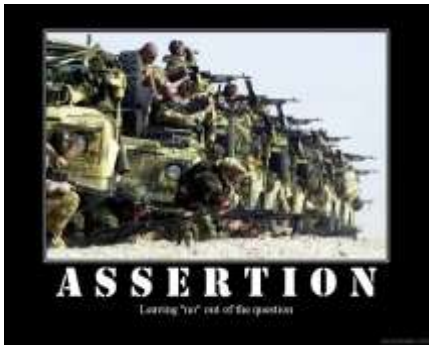
1. Open Browser URL
2. Locate the username field and fill it with

```
module.exports = {
  "Login Existing User" : function (browser) {
    browser
      .url(browser.launchUrl)
      .waitForElementVisible('body', 1000, 'Page loads')
      .pause(1000)
      .setValue('input[name=identification]', 'admin')
      .setValue('input[name=password]', 'doNotGiveAtry')
      .click('button[type=submit]')
      .waitForElementVisible('div[id=playground]', 1000, 'Login successful')
      .end();
  }
};
```

You need to know Javascript and how to use CSS or Xpath Selectors.
You can learn about it here: http://www.w3.org/TR/CSS21/selector.html
or http://www.w3schools.com/css/css_attribute_selectors.asp

Assertation or Verification

- assert.title('automated tests rocks')
  - If false, log failure and stop to run the test suite.



- verification.title('automated tests rocks')
  - If false, log failure and continue running the test suite.

What kind of assertations exists?

- .assert.title(‘Catalogue’)
- .assert.urlContains('done')
- .assert.elementPresent('#results_size', ’0 results')

```
this.demoTest = function (browser) {
  browser.getText("#main ul li a.first", function(result) {
    this.assert.equal(typeof result, "object");
    this.assert.equal(result.status, 0);
    this.assert.equal(result.value, "nightwatchjs.org");
  });
};
```

- css properties, … and more available!

How to run?

- $ node nightwatch.js –-env=chrome
- Windows you can test with ie:
  - $ node nightwatch.js –-env=chrome,ie, firefox
- You can skip test groups:
  - $ node nightwatch –-env=chrome –-skip-group=accounts,dashboard
- You can run the only ones that has a specific tag:
  - $ node nightwatch –-env=chrome –-tag=login

What the output it retrieves?

Console output

```
[2 Advancedsearch] Test Suite
=============================

Running:  Searching using Advanced Search

✔  Page loads
✔  Login successful
✔  Dashboards opens and link is visible
✔  Advanced search loads with success
✔  Questionset 1 Loaded with success
✖  Search Returned with success  - expected "visible" but got: not found

FAILED:  1 assertions failed, 5 passed and 1 skipped (16.388s)
```

What the output it retrieves?

Junit format

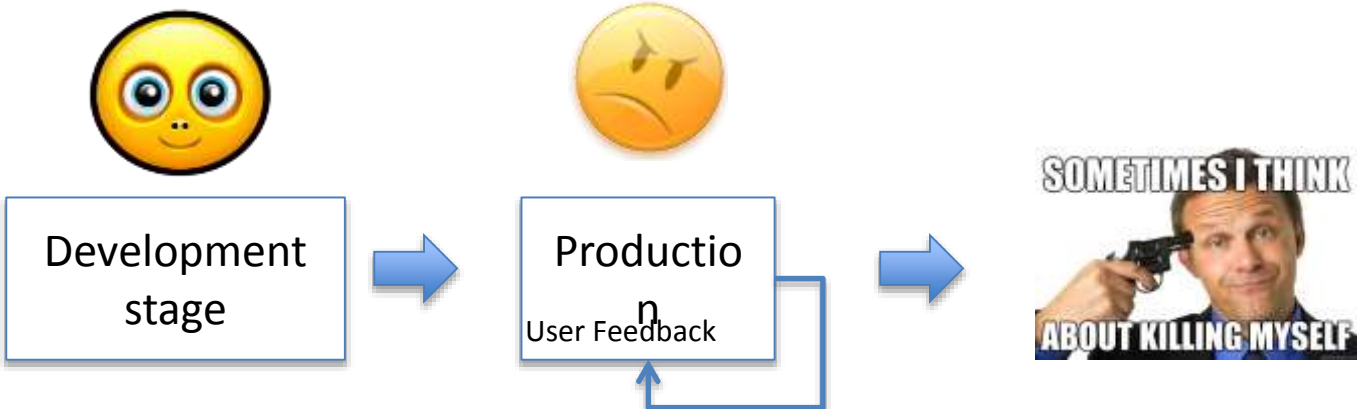
```
<?xml version="1.0" encoding="UTF-8"?>
<testsuites name="2_advancedsearch" errors="0" failures="1" tests="7">
   <testsuite errors="0" failures="1" hostname="" id="" name="Searching using Advanced
Search" package="2_advancedsearch" skipped="1" tests="7" time="" timestamp="">
      <testcase name="Page loads" />
      <testcase name="Login successful" />
      <testcase name="Dashboards opens and link is visible" />
      <testcase name="Advanced search loads with success" />
      <testcase name="Questionset 1 Loaded with success" />
      <testcase name="Search Returned with success">
         <failure message="Expected &quot;visible&quot; but got: &quot;not found&quot;" />
      </testcase>
   </testsuite>
</testsuites>
```

Can I use it in my own project?

Write tests? Nooooooooooooo!

- Is your project web? If yes:
  - Yes, you should.
  - We have a Windows machine up and running You can integrate Jenkins of the Bio group.
  - In some cases you may need to change the webapp to become the life easier.
  - You will avoid headaches in the future ☺

If you do **NOT** use Browse Automating Testing:

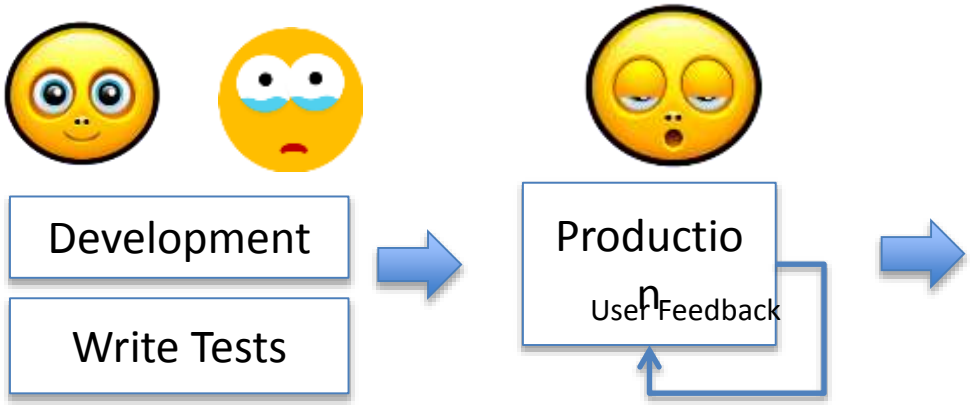


Can I use it in my own project?

Write tests? Easy life! But yes, you still spend time, but you win upfront.

- Is your project web? If yes:
  - Yes, you should.
  - We have a Windows machine up and running You can integrate Jenkins of the Bio group.
  - In some cases you may need to change the webapp to become the life easier.
  - You will avoid headaches in the future ☺

If you do **USE** Browse Automating Testing:



Credits & References

- Ricardo Ribeiro
  - He did the first steps in the nightwatch.js
  - Deployed the full solution in Windows environment
  - Write tests for EMIF Catalogue
- http://nightwatchjs.org/
- http://www.slideshare.net/sethmcl/join-the-darkside-nightwatchjs
- It will be available in my slideshare:
  - http://www.slideshare.net/bastiao

See it running…

- EMIF Catalogue, let’s check the tests

Thanks for your attention!

Luís A. Bastião Silva

University of Aveiro
DETI / IEETA
bastiao@ua.pt

universidade de aveiro

deti departamento de electrónica, telecomunicações e informática

ieeta instituto de engenharia electrónica e telemática de aveiro