E2E testing web apps locally or in the cloud (Sauce Labs) using Nighwatch.js and WebDriver in Javascript

Overview

In this tutorial, we will create a test in Javascript using Nightwatch.js for E2E tests, web based, running either locally or in the cloud by using Sauce Labs.

Requirements

- Install NodeJS
- Install all dependencies using "npm"

Description

This tutorial is based on some examples from this project.

You may start by cloning the project's repository.

```
git clone https://github.com/dwyl/learn-nightwatch
```

Nightwatch.js configuration is managed in a configuration file, similar to the following one.

nightwatch.conf.js

```
require('env2')('.env'); // optionally store your environment variables in .env
const seleniumServer = require("selenium-server");
const chromedriver = require("chromedriver");
const PKG = require('./package.json'); // so we can get the version of the project
const SCREENSHOT_PATH = "./node_modules/nightwatch/screenshots/" + PKG.version + "/";

const config = { // we use a nightwatch.conf.js file so we can include comments and helper functions
  "src_folders": [
    "test/e2e"     // we use '/test' as the name of our test directory by default. So 'test/e2e' for 'e2e'.
  ],
  "output_folder": "./node_modules/nightwatch/reports", // reports (test outcome) output by Nightwatch
  "selenium": {
    "start_process": true,
    "server_path": seleniumServer.path,
    "log_path": "",
    "host": "127.0.0.1",
    "port": 4444,
    "cli_args": {
      "webdriver.chrome.driver" : chromedriver.path
    }
  },
  "test_workers" : {"enabled" : true, "workers" : "auto"}, // perform tests in parallel where possible
  "test_settings": {
    "default": {
      "launch_url": "http://localhost", // we're testing a Public or "staging" site on Saucelabs
      "selenium_port": 80,
      "selenium_host": "ondemand.saucelabs.com",
      "silent": true,
      "screenshots": {
        "enabled": true, // save screenshots to this directory (excluded by .gitignore)
        "path": SCREENSHOT_PATH
      },
      "username" : "${SAUCE_USERNAME}",     // if you want to use Saucelabs remember to
      "access_key" : "${SAUCE_ACCESS_KEY}", // export your environment variables (see readme)
      "globals": {
        "waitForConditionTimeout": 10000    // wait for content on the page before continuing
      }
    },
```

```
"local": {
  "launch_url": "http://localhost",
  "selenium_port": 4444,
  "selenium_host": "127.0.0.1",
  "silent": true,
  "screenshots": {
    "enabled": true, // save screenshots taken here
    "path": SCREENSHOT_PATH
  }, // this allows us to control the
  "globals": {
    "waitForConditionTimeout": 15000 // on localhost sometimes internet is slow so wait...
  },
  "desiredCapabilities": {
    "browserName": "chrome",
    "chromeOptions": {
      "args": [
        `Mozilla/5.0 (iPhone; CPU iPhone OS 5_0 like Mac OS X) AppleWebKit/534.46
        (KHTML, like Gecko) Version/5.1 Mobile/9A334 Safari/7534.48.3`,
        "--window-size=640,1136" // iphone 5
      ]
    },
    "javascriptEnabled": true,
    "acceptSslCerts": true
  }
},
"chrome": { // your local Chrome browser (chromedriver)
  "desiredCapabilities": {
    "browserName": "chrome",
    "javascriptEnabled": true,
    "acceptSslCerts": true
  }
},
"chromemac": { // browsers used on saucelabs:
  "desiredCapabilities": {
    "browserName": "chrome",
    "platform": "OS X 10.11",
    "version": "47"
  }
},
"ie11": {
  "desiredCapabilities": {
    "browserName": "internet explorer",
    "platform": "Windows 10",
    "version": "11.0"
  }
},
"firefox" : {
  "desiredCapabilities": {
    "platform": "XP",
    "browserName": "firefox",
    "version": "33"
  }
},
"internet_explorer_10" : {
  "desiredCapabilities": {
    "platform": "Windows 7",
    "browserName": "internet explorer",
    "version": "10"
  }
},
"android_s4_emulator": {
  "desiredCapabilities": {
    "browserName": "android",
    "deviceOrientation": "portrait",
    "deviceName": "Samsung Galaxy S4 Emulator",
    "version": "4.4"
  }
},
"iphone_6_simulator": {
  "desiredCapabilities": {
    "browserName": "iPhone",
```

```
        "deviceOrientation": "portrait",
        "deviceName": "iPhone 6",
        "platform": "OSX 10.10",
        "version": "8.4"
      }
    }
  }
}
module.exports = config;

function padLeft (count) { // theregister.co.uk/2016/03/23/npm_left_pad_chaos/
  return count < 10 ? '0' + count : count.toString();
}

var FILECOUNT = 0; // "global" screenshot file count
/**
 * The default is to save screenshots to the root of your project even though
 * there is a screenshots path in the config object above! ... so we need a
 * function that returns the correct path for storing our screenshots.
 * While we're at it, we are adding some meta-data to the filename, specifically
 * the Platform/Browser where the test was run and the test (file) name.
 */
function imgpath (browser) {
  var a = browser.options.desiredCapabilities;
  var meta = [a.platform];
  meta.push(a.browserName ? a.browserName : 'any');
  meta.push(a.version ? a.version : 'any');
  meta.push(a.name); // this is the test filename so always exists.
  var metadata = meta.join('~').toLowerCase().replace(/ /g, '');
  return SCREENSHOT_PATH + metadata + '_' + padLeft(FILECOUNT++) + '_';
}

module.exports.imgpath = imgpath;
module.exports.SCREENSHOT_PATH = SCREENSHOT_PATH;
```

The capabilities (devices/browsers) we want, along with the reporter/JUnit related configurations, are all defined in the previous configuraiton file.

Nightwatch.js provides a built-in JUnit reporter, which we'll use.

> **Please note**
>
> The tests below don't use the Page Objects pattern. However, it would be a good recommendation to do so as mentioned by Nightwatch.js.
>
> You may find other test examples, using the Page Objects pattern, in nightwatch.js repository here.

Test 1: Verify the content of a page on GitHub

test/e2e/github.js

```
var conf = require('../../nightwatch.conf.js');

module.exports = {
  'Demo test GitHub': function (browser) {
    browser
      .url('http://www.github.com/dwyl')   // visit the url
      .waitForElementVisible('body'); // wait for the body to be rendered
      // check if we are seeing the Mobile Version of GitHub
      browser.element('css selector', '.switch-to-desktop', function(result) {
        if(result.status != -1) { //Element exists, do something
          browser.click('.switch-to-desktop')
          .waitForElementVisible('body'); // wait for the body to be rendered
        }
      });
    // part two:
    browser
      .assert.containsText('body', 'dwyl.com') // assert body contains text
      .saveScreenshot(conf.imgpath(browser) + 'dwyl.png')
      .end();
    }
  };
```

Test 2: Verify the title of some page

test/e2e/guineaPig.js

```
var config = require('../../nightwatch.conf.js');

module.exports = { // addapted from: https://git.io/vodU0
  '@tags': ['guineaPig'],
  'Guinea Pig Assert Title': function(browser) {
    browser
      .url('https://saucelabs.com/test/guinea-pig')
      .waitForElementVisible('body')
      .assert.title('I am a page title - Sauce Labs')
      .saveScreenshot(config.imgpath(browser) + 'a-screenshot-description.png')
      .clearValue('#i_am_a_textbox')
      .setValue('#i_am_a_textbox', 'nightwatch roolz!')
      .saveScreenshot(config.imgpath(browser) + 'nightwatch-roolz.png')
      .end();
  }
};
```



Please note

Beware that some Javascript files, even though in the "tests" folder, may not be actual/real test cases (e.g. the file *test/e2e /upload_screenshots_to_s3.js* provided in the upstream project).

Test 3: upload screenshots to S3

test/e2e/upload_screenshots_to_s3.js

```
require('env2')('.env'); // optionally store youre Evironment Variables in .env
var conf = require('../../nightwatch.conf.js')
var fs = require('fs'); // read the screenshot files
var path = require('path');
var AWS = require('aws-sdk');
var mime = require('mime-types');
AWS.config.region = process.env.AWS_REGION;
var s3bucket = new AWS.S3({params: {Bucket: process.env.AWS_S3_BUCKET}});

function s3_create () {
  if(!process.env.AWS_ACCESS_KEY_ID) {
    console.log(`If you want to upload Screenshots to S3
      please set your AWS Environment Variables (see readme).`);
  }
  else {
    var SP = conf.SCREENSHOT_PATH;
    var version = SP.split('/')[ SP.split('/').length - 2 ];
    fs.writeFileSync(conf.SCREENSHOT_PATH + 'index.html', // don't overwrite index
      fs.readFileSync(path.join(__dirname + '/index.html')), 'utf8');
    // fs.createReadStream(path.join(__dirname + '/index.html'))
    //   .pipe(fs.createWriteStream(conf.SCREENSHOT_PATH + 'index.html'));
    // fist read the list of screenshots
    var images = fs.readdirSync(SP).filter(file => {
      return fs.statSync(SP + file).isFile()
        && file.indexOf('.png') > -1; // only screenshot images
    })
    // create meta.json with list of screenshots
    var meta = {images: images}
    fs.writeFileSync(path.join(SP, 'meta.json'), JSON.stringify(meta, null, 2));

    // get list of files to upload to S3 (including meta.json & index.html)
    fs.readdirSync(SP).forEach(function (file) {
      var filepath = path.join(SP, file);
      var mimetype = mime.lookup(filepath);
      if (mimetype) {
        var s3path = version + '/uat' +
          filepath.split('node_modules/nightwatch/screenshots/' + version)[1];
        var s3obj = new AWS.S3({ params: {
          Bucket: process.env.AWS_S3_BUCKET,
          ACL: 'public-read',
          Key: s3path,
          ContentType: mimetype,
        }});
        // upload (stream) the files to S3 in parallel
        s3obj.upload({Body: fs.createReadStream(filepath)}).send(function(e, data) {
          if (e) {
            console.log(' >>> ERROR:', e);
          }
          if (filepath.indexOf('index.html') > -1) {
            console.log('Uploaded', images.length, 'screenshots >> ', data.Location);
          }
        });
      }
    });
  }
}
s3_create();
```

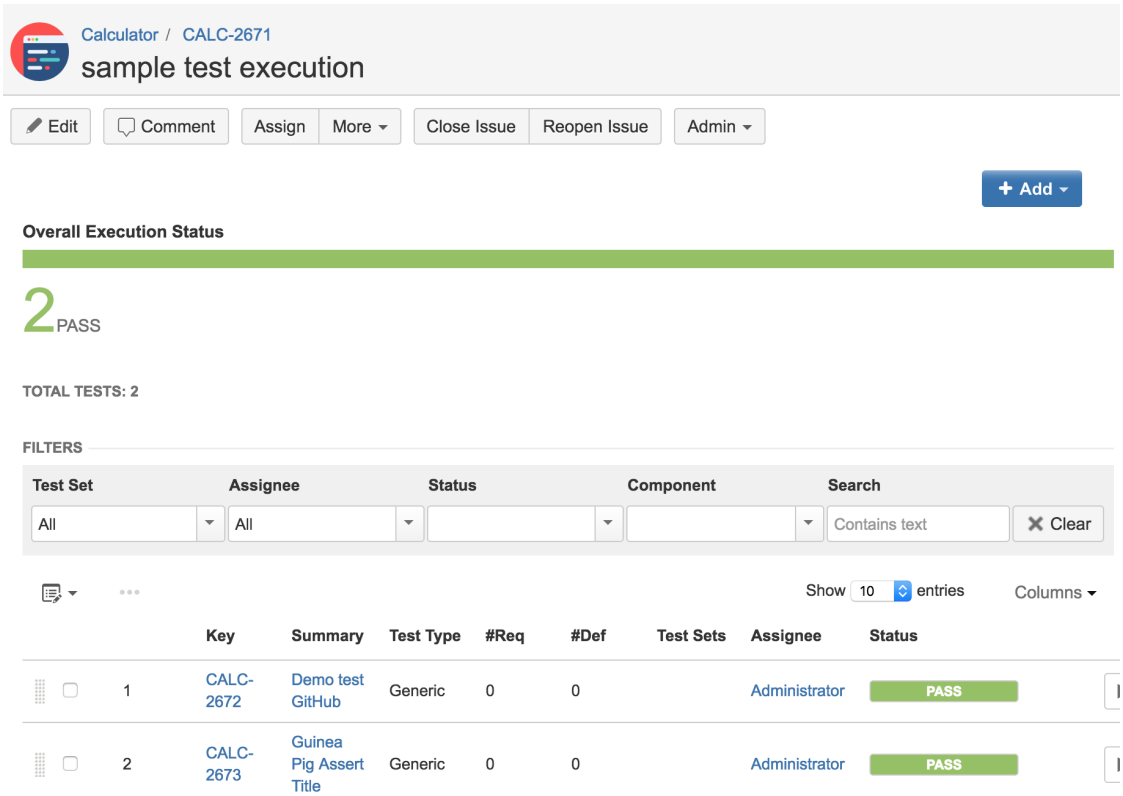
Running tests locally

Test(s) then can be run using NPM "test" task.

```
npm test
```

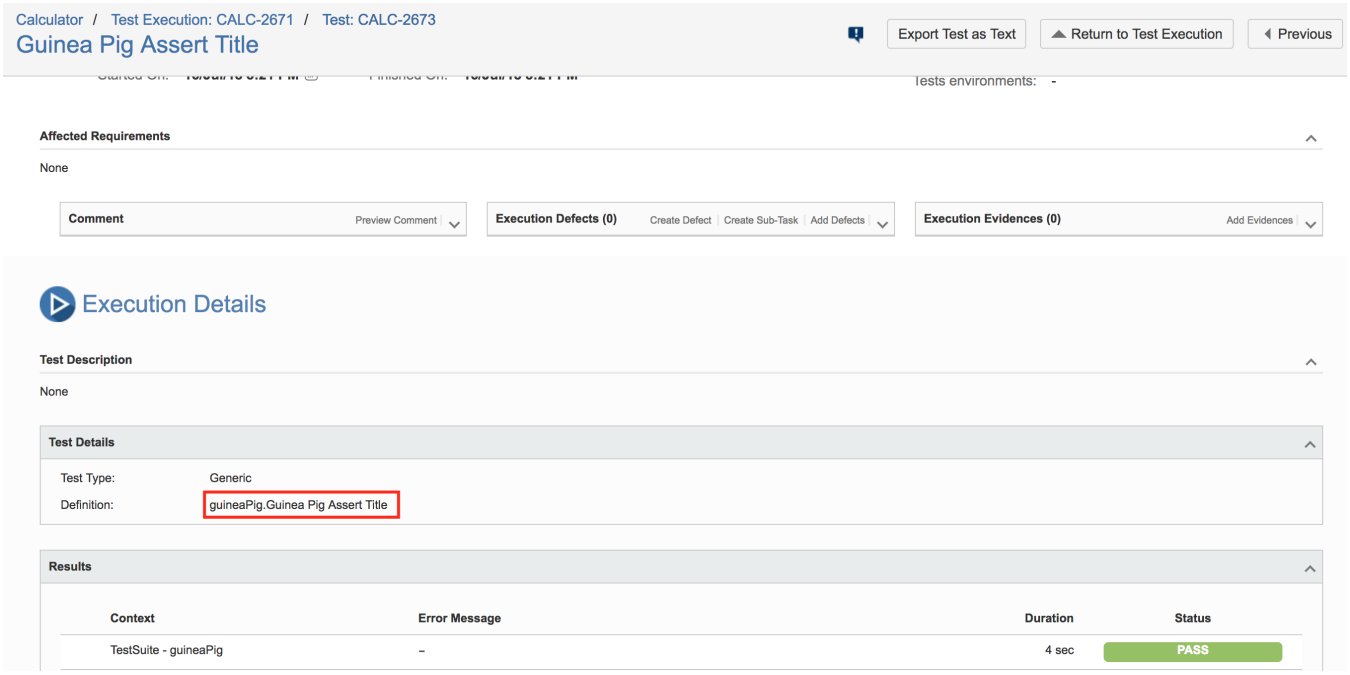
After successfully running the tests, JUnit XML files will be created, under the `node_modules/nightwatch/reports/` folder (e.g. CHROME_67.0.3396.99_Mac OS X_github.xml, CHROME_67.0.3396.99_Mac OS X_guineaPig.xml).

Reports can be imported to Xray (either by the REST API or through the **Import Execution Results** action within the Test Execution).



Tests are mapped to Generic Tests in Jira, and the **Generic Test Definition** field contains the name of the file along with the title of the function implementing the automate test.

The execution screen details will provide information on the overall test run result.



Running in the cloud using SauceLabs

Before running the test(s), you need to export some environment variables with your Sauce Lab's username along with the respective access key, which you can obtain from within the User Settings section in your Sauce Lab's profile page.

```
export SAUCE_USERNAME=<your Sauce Labs username>
export SAUCE_ACCESS_KEY=<your Sauce Labs access key>
```

Test(s) then can be run in parallel using NPM.

```
npm run sauce
```

Multiple JUnit XML files will be created, one per capability (device+browser), under the `node_modules/nightwatch/reports/` folder.

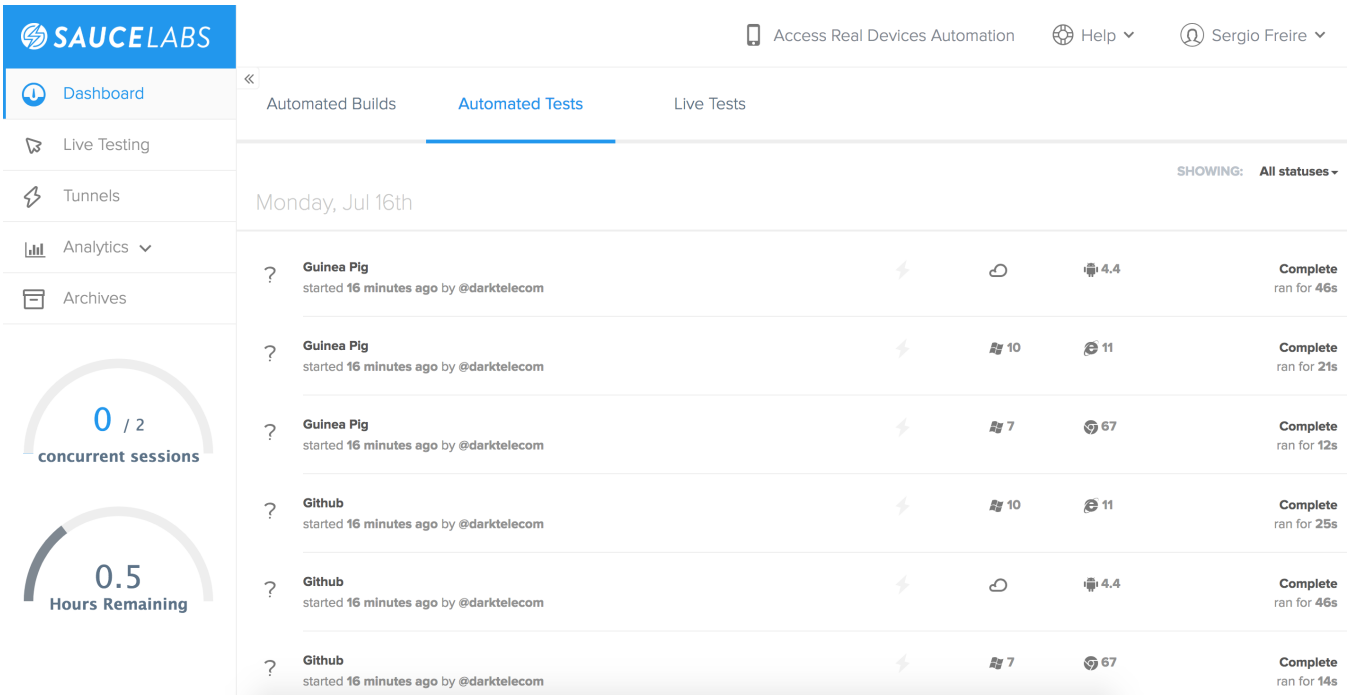
Please note

Since the previous command generates multiple JUnit XML files, we may need to merge them into a single XML file so it can be submitted into a Test Execution more easily. That can be achieved by using the junit-merge utility. In this case, we'll loose the visibility of results on a per Test Environment basis though.

```
junit-merge  -o results.xml -d node_modules/nightwatch/reports/
```

If we want to have visibility of the results per Test Environment, then we need to separately upload them and identify the Test Environment on each REST API request.

In Sauce Labs you can see some info about it.



References

- https://github.com/nightwatchjs/nightwatch
- http://nightwatchjs.org/
- https://github.com/dwyl/learn-nightwatch
- https://github.com/nightwatchjs/nightwatch/tree/master/examples