链滴

Pros and cons of end to end testing tools

作者：zxniuniu
原文链接：https://ld246.com/article/1543064084943
来源网站：链滴

My background in test automation is based on using Selenium WebDriver and Java. Since Jav Script has been widely used for testing web applications, I have decided to investigate tools c rrently available on the market. I will use these tools to make different scripts and hence find heir advantages and disadvantages.

I have investigated the following tools and syntax for automation:

1. Selenium Webdriver + Java
2. Selenium WebDriver + Cucumber + Java
3. Selenium WebDriver + Jasmine syntax for JavaScript applications
4. Selenium WebDriver + Mocha syntax for JavaScript applications
5. Cypress
6. Selenium WebDriver + Protractor
7. Selenium WebDriver + NightWatch
8. Selenium WebDriver + WebDriverIO

The focus of this article is not on finding advantages or disadvantages of Selenium WebDriver but I mainly focus on investigating the pros and the cons of different tools/frameworks in c mbination with Selenium WebDriver. Thus this article may be useful for those who already kn w Selenium and would like to gain knowledge of using different tools in combination with Sel nium.

I have created a simple test and will try to implement its script by using all the above automat on tools. I have chosen Amazon website as most of its elements are identified by an id and th s makes it easy to find an element.

1. Let us start from the first one in the above list, which is** “Selenium WebDriver + Java” . * For many years this has been a popular style for making automation and I personally have b en using it. In the following you see a test that is implemented in this style.

```
// open a browser
driver.get("https://www.amazon.com/");

// click on Your Amazon.com link
WebElement yourAmazon = driver.findElement(By.id("nav-your-amazon"));
yourAmazon.click();

// enter your email address
WebElement email = driver.findElement(By.id("ap_email"));
email.sendKeys("Test123@email.com");

// enter your password address
WebElement password = driver.findElement(By.id("ap_password"));
password.sendKeys("1234");

//press sign in button
WebElement signIn =driver.findElement(By.id("signInSubmit"));
signIn.click();

// check if the error message is displayed
WebElement errorMessageBox = driver.findElement(By.id("auth-error-message-box"));
errorMessageBox.isDisplayed();

//close browser
driver.close();
```

After making the above script, advantages of this style can be summarized as follows:

● It has a quick and popular setup. For setting up this test, you only need to import Driver and the Selenium library Java. If you need more information on how to set up your test, please re d here: https://www.guru99.com/intellij-selenium-webdriver.html

● If any help in making scripts is needed, there are lots of sources available online that can be used by testers.

● A very basic knowledge of Java is enough for implementing the scripts.

● Java structure can be adjusted quite easily within a development team who know the Java. I highly recommend using this style if a tester is in a Java team, as it is productive for the team o create scripts. The team members can easily pick up automation tests to help the tester, thu tests are not only implemented by the tester but by the whole team.

Disadvantages of this style can be summarised as follows:

● It is not possible to add a clear description for each step. If you are looking for a clean synta , the combination of Java and Selenium doesn’ t provide it for you. You can add a description of each step as a comment but this will not be displayed in the logs if the test fails.

● You can also add a clear error message for each test to make it easier to find the failing step however it is hard to keep this syntax clean.

==========

2. **Selenium WebDriver + Cucumber + Java:**

https://www.stevefenton.co.uk/2015/01/getting-started-with-bdd-intellij

The test includes three parts:

- Feature file, which is a clear description for each step of your test using Gherkin syntax.
- Step file, which is the script for matching the features and the Java code.
- TestRun, which is the configuration setup for running your test.

Let’ s see how the test is written down :

[1]: Feature file:

```
Feature: CucumberJava
       Scenario: Login functionality exists
       Given I have open amazon website
       And click on your Amazon.com link
       When  enter username "test@gmail.com" and password "123"
       When press sign in button
       Then the error message box is displayed
```

[2]: Step file:

```
  class cucumberJava {
  WebDriver driver = null;

  @Given("^I have open amazon website$")
  public void openBrowser() {
      driver = new FirefoxDriver();
      driver.navigate().to("https://www.amazon.com/");
  }

  @And("^click on your Amazon.com link$")
   public void clickOnYourAmazon(){
      driver.findElement(By.id("nav-your-amazon")).click();
  }

  @When("^enter username \"(.*)\" and password \"(.*)\" $")
     public void UserAndPassword(String username,String password){
       driver.findElement(By.id("ap_email")).sendKeys(username);
       driver.findElement(By.id("ap_password")).sendKeys(password);
  }

  @When("^press sign in button$")
    public void SignInButton(){
        driver.findElement(By.id("signInSubmit")).click();
  }

  @Then("^the error message box is displayed$")
  public void errorMessage() {
      if(driver.findElement(By.id("auth-error-message-boc")).isDisplayed()) {
          System.out.println("Test 1 Pass");
```

```
        } else {
            System.out.println("Test 1 Fail");
        }
        driver.close();
    }
}
```

3. TestRun:

```
package cucumberJava;

import org.junit.runner.RunWith;
import cucumber.junit.Cucumber;

@RunWith(Cucumber.class)
@Cucumber.Options(format = {"pretty", "html:target/cucumber"},
        glue = {"cucumberJava"},
        features = {"src/test/java/cucumberJava/cucumberJava.feature"}

)
public class runTest { }
```

Advantages:

● A descriptive test step is helpful when a test fails. This is considered mainly as an advantage for business people but I believe this is also advantageous for the development team as it giv s them a clean and readable description of test steps (for example why this test, how it is don , etc.), and maintenance of automation tests in large projects will be easier due to the descript ve steps.

● It is a comfortable tool for development teams who build java applications with java langua e, It will help the team to be more productive.

Disadvantages:

● Your feature file can easily become a very large and messy file.

● Running tests can take quite some time, especially when you want to run your test in differ nt browsers.

● If you are going to test a JavaScript application by Cucumber and Java, it might be difficult or the development team to adjust itself which is less productive.

● It was not quick or easy to set up, it took me some time to figure out how to link the featur s file with the steps file!!

● Most people use the BDD given, when, then syntax in a structure which is not sensible. It is ssential to know that this syntax is designed for the following reasons:

1. Given provides context for the test scenario that is going to be executed, such as the po nt in your application that the test occurs as well as any prerequisite data.

2. When specifies the set of actions that triggers the test, such as user or subsystem actions.

3. Then specifies the expected result of the test.

==========

Let us now look at the following tools which are mostly used for JavaScript applications.

3. Selenium WebDriver with Jasmine Synatx:

I started to set up my test based on the information in the blog post: http://blog.scottlogic.com/2017/08/24/adding-webdriver-tests-to-create-react-app-generated-projects.html

The setup is quick and easy, but it has three different parts:

- Page objects: the place where you find your elements by css/xpath.
- Specs: for writing the actual script.
- Configuration files: you can set up your "tear down" over here.

Let us again create our script for checking the correct error message on Amazon and see what the syntax looks like with this structure:

Page object:

```
const yourAmazonSelector = { css: '#nav-your-amazon' };
const emailSelector = { css: '#ap_email' };
const passwordSelector = { css: '#ap_password' };
const signInSelector = { css: '#signInSubmit' };
const errorMessageSelector = { css: '#auth-error-message-box' };

export const yourAmazon = () => body().findElement(yourAmazonSelector);

export const email = () => body().findElement(emailSelector);

export const password = () => body().findElement(passwordSelector);

export const continueSignIn = () => body().findElement(continueButton);

export const signIn = () => body().findElement(signInSelector);

export const errorMessage = () => body().findElement(errorMessageSelector);
```

Specs:

```
import { yourAmazon, email, password, errorMessage, signIn, continueSignIn } from '../pageO
jects/app';
import { load } from '../pageObjects/index';

describe('app', async () => {
 beforeAll(async () => {
   await load();

 });
```

```
 let randomEmail = Math.random();
 let randomPassword = Math.random();

 it('should display the correct error message for incorrect login details', async () => {
   await yourAmazon().click();
   await email().sendKeys(randomEmail + "@gmail.com");
   await password().sendKeys(randomPassword);
   await signIn().click();
   expect(await errorMessage().isDisplayed()).toBe(true);

 });

});
```

Advantages:

● There are no other tools involved in this test, it is only Selenium WebDriver + Jasmine synta
.

● In Jasmine we need to use the describe function that helps for grouping our tests together.

● Jasmine comes with a number of matchers that help you make a variety of assertions. You s ould read the Jasmine documentation to see what they are. To use Jasmine with Karma, some people use the karma-jasmine test runner.

● It has Angular support.

● Jasmine's beforeAll() hook is often useful for sharing test setup - either to reduce test run-t me or simply to make for more focused test cases and it reduces lines of code too!!

Disadvantages:

● There is less documentation available for finding information about setting up your css in t e case of having multiple ids for one element.

● It took me a while to figure out how to use “beforeAll()” instead of “beforeEach()” in my specs: if you use “beforeEach” then you can not use nested test steps, because in each step t tries to open a new browser and running a new test and it runs before each 'it' block!! Howe er if you use “beforeAll”, you can have multiple nested steps in the same test.

==========

4. Selenium WebDriver with Mocha syntax:

Let’s start to make a test with Selenium and Mocha syntax by following this document: http: /testerstories.com/2016/02/javascript-with-selenium-webdriver-and-mocha/

The setup was quick and easy. Mocha is a simple, flexible and fun, java-script test framework or node.js and browsers.

Here's the script using Mocha:

```
 var assert = require('assert'),
   fs = require('fs'),
```

```
  test = require('selenium-webdriver/testing'),
  webdriver = require('selenium-webdriver');

test.describe('My Website', function () {
  this.timeout(15000);
  var driver;
  test.before(function () {
     driver = new webdriver.Builder().withCapabilities(webdriver.Capabilities.chrome()).build();
  });

  var emailRandom = Math.random();
  var passwordRandom = Math.random();

  test.it('should display the correct error message with incorrect login information', function ()
{
     driver.get('https://www.amazon.com/');
     driver.findElement(webdriver.By.id('nav-your-amazon')).click();
     driver.findElement(webdriver.By.id('ap_email')).sendKeys(emailRandom + "@gmail.com");
     driver.findElement(webdriver.By.id('ap_password')).sendKeys(passwordRandom);
     driver.findElement(webdriver.By.id('signInSubmit')).click();
     var errorMessage = driver.findElement(webdriver.By.id('auth-error-message-box')).isDispl
yed();
     if (errorMessage == true){
        return true;
     }else return false;
  });
  test.after(function () {
     driver.quit();
  });
});
```

Advantages:

- Mocha's before() hook is often useful for sharing test setup - either to reduce test run-time or simply to make for more focused test cases.
- There are no other tools involved in this test, it is only Selenium WebDriver + Mocha syntax.
- In Mocha we need to use the describe function that helps for grouping our tests together.

==========

5. Cypress :

While I was busy with writing this blog post for different tools that use Selenium for end to e d testing, cypress got my attention and I started to make a simple script by using Cypress. Cy ress doesn’ t use Selenium and it is a bit different with the above tools that I have experimen ed.

The setup was quick and fast, you don't need any special configuration for running your test.

If you are new in using Cypress, you can follow the following website for installing it: https:// ocs.cypress.io/guides/getting-started/installing-cypress.html#

If you need more sources for getting familiar with cypress, you can use the following: https://xample.cypress.io/

I have created my running script for Amazon website here:

```
describe('should display a correct error message when you enter wrong login information', fu
ction () {
 before(function () {
   cy.visit('https://www.amazon.com/')
 })

 var randomEmail = Math.random();
 var randomPassword = Math.random();

 it('displays a correct error message', function () {
   cy.get('#nav-your-amazon').click()
   cy.get('#ap_email').type(randomEmail + '@gmail.com')
   cy.get('#ap_password').type(randomPassword)
   cy.get('#signInSubmit').click()
   cy.get('#auth-error-message-box').should('be.visible')
 })
})
```

I like cypress a lot because of the following reasons:

● Cypress does not use Selenium: most of the end to end tools that we have experimented wi h, are using Selenium, that’ s why they have almost the same problems.

● Cypress supports any framework or website quite well: There are hundreds of projects using the latest React, Angular, Vue, Elm, etc. frameworks. Cypress also works equally well on older erver rendered pages or applications.

● Cypress tests are only written in JavaScript: While you can compile down to JavaScript from any other language, ultimately the test code is executed inside the browser itself. There are no languages or driver bindings - there is and will only ever be just JavaScript.

● There are no dependencies, you put your test in package.json and that’ s it.

● Cypress runs much, much faster in comparison with the end to end tools by Selenium that e have experimented.

● There is screen shot for every step, of your script, which can be quite helpful if there is any f lse passing or failing test, yeah good for debugging!!

● Cypress has a clear cyntax, it is easy to read it, you will like it!!

Disadvantages:

● The structure was different to the other Selenium end to end tools, so at first you may need to spend more time understanding the structure and finding the best way to create your scrip s.

● Community: As Cypress is relatively new, the community is small. You will have trouble findi g answers to problems etc.

● Features. No file upload support. No cross-browsers testing. Who knows when these things will be covered, as for big projects these features are crucial.

● Page Object Model. It is something that has already been proven by time. Cypress supports a different approach which could be controversial. More detail on this is here: Cypress POM

● It's only available for only one client (language) i.e for JavaScript only. So to work with it you must know JavaScript: however this might be an advantage for JavaScript application, but I w uld like to put it as a disadvantages for those who have difficulties with javascript.

reference for more information

==========

6. Protractor tool with Selenium WebDriver :

I followed this tutorial for making my first script with protractor and Selenium WebDriver:http ://github.com/angular/protractor/blob/master/docs/tutorial.md

And this is my running script for Amazon website:

```
 // spec.js
describe('should display a correct error message when I enter wrong login information', funct
on () {
 it('correct error message', function () {
   browser.waitForAngularEnabled(false);
   browser.get('https://www.amazon.com/');

   //click on orders link
   const Orders = element(by.id('nav-orders'));
   Orders.click();
   // enter your email
   const email = element(by.id('ap_email'));
   email.sendKeys('test@gmail.com');
   // enter your password
   const password = element(by.id('ap_password'));
   password.sendKeys('7899');

   //check if the error box is not displayed
   const errorBox1 = element(by.id('auth-warning-message-box'));
   expect(errorBox1.isPresent()).toEqual(false);

   //click on sign in button
   const signIn = element(by.id('signInSubmit'));
   signIn.click();

   // check if the error box is displayed
   const errorBox = element(by.id('auth-error-message-box'));
   expect(errorBox.isDisplayed()).toEqual(true);

   const message = element(by.css('.a-list-item'));
   expect(message.getText()).toEqual('To better protect your account, please re-enter your pas
word and then enter the characters as they are shown in the image below.');

 });
});
```

Advantages:

● Suitable for both Angular and non-Angular apps. Protractor gives extra advantages for testi g Angular apps but your app should not necessarily use it, If you have an application that is n t Angular and you would like to use Protractor, you always need to add the following to your pec BEFORE opening your browsers:

```
browser.waitForAngularEnabled(false);
```

● Protractor has built a support for identifying the elements for angular.js applications which i the following:

```
by.binding
by.excatBinding
by.model
by.repeater
by.exactRepeater
by.options
```

If you would like to know more about the differences of these Locator Strategies, please read he following article: http://www.webdriverjs.com/angular-specific-locators-in-protractor/

● Parallel testing through several browsers. It supports cross-browser testing. Even more，you can run several browsers instances simultaneously!

Disadvantages:

● Debugging: I personally found it tricky to debug protractor.

● It's available for only one client (language) i.e. for JavaScript only. So you must know JavaScr pt to work with it.

● It does not support automating mobile Apps.

● It is implemented as a wrapper to the WebdriverJs. So there is one more layer added in bet een Selenium server and the Protractor. If there is an issue with WebdriverJs, the Protractor te m should wait for the WebDriverJs team to fix that issue.

==========

7. NightWatch with Selenium WebDriver:

I followed this tutorials for making my first script by NightWatch and Selenium WebDriver: ht ps://github.com/dwyl/learn-nightwatch Below is my script for Amazon website. The test consi ts of three parts: * Package * Nightwatch configuration file * script The package consisted of t e following:

```
{
"name": "nightwatch",
"version": "1.0.0",
"description": "",
```

```
  "main": "index.js",
  "scripts": {
    "test": "nightwatch",
    "e2e-setup": "selenium-standalone install"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "babel-cli": "^6.26.0",
    "babel-plugin-add-module-exports": "^0.2.1",
    "babel-preset-es2015": "^6.24.1",
    "selenium-standalone": "^6.12.0"
  }
}
```

The configuration file for this script consists of the following:

```
  {
    "src_folders" : ["tests"],
    "output_folder" : "reports",

    "selenium" : {
      "start_process" : true,
      "server_path" : "bin\selenium-server-standalone-3.3.1.jar",
      "log_path" : "",
      "port" : 4445,
      "cli_args" : {
        "webdriver.chrome.driver" : "bin\chromedriver.exe"
      }
    },

    "test_settings" : {
      "default" : {
        "launch_url" : "http://localhost",
        "selenium_port"  : 4445,
        "selenium_host"  : "localhost",
        "desiredCapabilities": {
          "browserName": "chrome",
          "javascriptEnabled": true,
          "acceptSslCerts": true
        }
      }
    },

    "scripts": {
      "test-e2e": "nightwatch"
    }

  }
```

There is a bug in NightWatch using Selenium port 4444.
The Selenium server didn’ t run when I specified port 4444, but it run successfully by changi g the port to 4445.

The script looks like this:

```
module.exports = {
    'Should display a correct error message when you enter a wrong login information' : functi
n (client) {
            client
            .url ('http://www.amazon.com/')
            .waitForElementVisible('body', 1000)
            .click('#nav-orders')
        .setValue('#ap_email', 'test@gmail.com')
        .setValue('#ap_password' ,'123')
        .click('#signInSubmit')
        .assert.visible('#auth-error-message-box')
        .end();

    }
};
```

Advantages:

● Clean syntax: Simple but powerful syntax enables you to write tests very quickly.

● Built-in test runner: Built-in command-line test runner can run the tests either sequentially r in parallel, together, by group, tags, or single.

● Cloud services support: Works with cloud testing providers, such as SauceLabs and Browser tack.

● CSS & Xpath support: Either CSS or Xpath selectors can be used to locate and verify elemen s on the page or execute commands.

● Continuous Integration support: JUnit XML reporting is built-in so you can integrate your te ts in your build process with systems such as Teamcity, Jenkins, Hudson etc.

Disadvantages:

● It does not have many choices for unit test frameworks as it has an own testing framework nd also support Mocha.

● Slightly lesser support in compare with WebDriverIO and Protractor.

==========

8. WebDriverIO with Selenium WebDriver: I followed this tutorial for making my first script b WebDriverIO and Selenium WebDriver:http://blog.kevinlamping.com/testing-your-login-an-i -depth-webdriverio-tutorial/

This test also consisted of three parts:

● package

- wdio.config
- script

The package and the configuration file is quite similar to NightWatch test, so I only put the scr pt here:

```
describe('Login Page', function () {
  it('Should display a correct error message with wrong login information', function () {
      browser.url('/');
      browser.click('#nav-your-amazon');
      browser.setValue('#ap_email', 'test@gmail.com');
      browser.setValue('#ap_password', '123');
      browser.click('#signInSubmit');
      browser.isVisible('#auth-error-message-box')
  })
})
```

Advantages:

- It has support for most BDD and TDD test frameworks.
- It has good support, enthusiastic developer community, and end users which give it an edg over NightwatchJS.
- It can be used with ‘webdrivercss’ to compare css stylings of an element in the webpage.
- Works with any testing framework or assertion library: WebdriverIO lets you use your favori e testing framework (Jasmine, Mocha, Cucumber) and assertion library (Chai for Mocha).

Disadvantages:

- Since it is a custom implementation, it is also a disadvantage as it deviates from generic syn ax which may confuse Selenium developers coming from other languages.
- It can be used for automating AngularJS apps but it is not as customized as Protractor.
- Must run with WDIO to debug: Tasks written in this beautiful Selenium API can only be deb gged using the provided WDIO task runner. You can't set breakpoints within tasks, but you c n have WDIO pause the run between Selenium commands.
- I did not find much documents for latest version (4.0.5)

==========

Conclusion: In this document we considered a simple test and created its scripts by different nd to end testing tools. We have experimented different syntaxes on those scripts. In my opin on the difference between these tools is not huge. The tool should be selected mainly based n your application and knowledge of the team. The latter is very important since automation i not the responsibility of individual team members, rather an entire team should contribute t wards it. Learning a new programming or scripting language will definitely enhance the skills f team members but working on a common ground keep all team members motivated.

As a tester, you should find out the answers of the following questions by examining the appl cation you are going to test:

- Is the application built using Angular, React, etc.?
- Are you looking for a special testing framework like Jasmine, Mocha, etc.?
- Are you looking for a tool that supports mobile (APPIUM)?
- Are you looking to test in any specific browser?

Depending on the answers of the above questions, you can select your suitable tool.

谢谢原作者，博客转自：https://blog.scottlogic.com/2018/01/08/pros-cons-e2e-testing-tools.html