

## ỦY BAN NHÂN DÂN TP. HỒ CHÍ MINH

## TRƯỜNG ĐẠI HỌC SÀI GÒN

**KHOA CÔNG NGHỆ THÔNG TIN**

ĐỒ ÁN KIỂM THỬ PHẦN MỀM

# **TÊN ĐỀ TÀI : SỬ DỤNG SELENIUM TRONG KIỂM THỬ WEBAPP**

Nhóm 10:

Trần Phan Thanh Long-3118410248

Lại Đức Trọng - 3118410451

Huỳnh Minh Quang - 3118410344

Giáo viên giảng dạy:Trương Tấn Khoa

TP.Hồ Chí Minh, ngày … tháng … năm 2020

### Chương I: TestNG

#### **TestNG là gì?**

**Định nghĩa về TestNG theo tài liệu chính thức như sau:**

TestNG là một testing framework lấy cảm hứng từ JUnit và NUnit, nhưng giới thiệu một số chức năng mới giúp nó mạnh mẽ hơn và dễ sử dụng hơn.

TestNG là một testing framework tự động mã nguồn mở; trong đó từ NG có nghĩa là NextGeneration. TestNG tương tự như JUnit (đặc biệt là JUnit 4), nhưng nó không phải là một phần mở rộng của JUnit. Nó được lấy cảm hứng từ JUnit. Nó được thiết kế để tốt hơn JUnit, đặc biệt là khi thử nghiệm các Class tích hợp. Người tạo ra TestNG là Cedric Beust.

Loại bỏ hầu hết các hạn chế của các framework cũ, TestNG cung cấp cho nhà phát triển khả năng viết các testcase linh hoạt và mạnh mẽ hơn. Vì nó vay mượn nhiều từ Java Annotations (được giới thiệu với JDK 5.0) để xác định các TestCase.

**Các tính năng của TestNG.**

Hỗ trợ Annotation

Hỗ trợ thử nghiệm các lớp tích hợp (ví dụ: không cần tạo một instance mới cho các test class).

Tách biệt các code test chạy trong compile-time khỏi các cấu hình chạy trong run-time và thông tin dữ liệu.

Giới thiệu đến sự ra đời của Test groups. Khi bạn đã tổng hợp các bài kiểm tra của mình, bạn chỉ có thể yêu cầu TestNG chạy tất cả các bài kiểm tra "front-end" hoặc các bài kiểm tra "fast", "slow", "database", v.v.

Hỗ trợ các phương pháp thử nghiệm phụ thuộc, thử nghiệm song song, thử nghiệm tải và lỗi một phần.

Plug-in API linh hoạt.

Hỗ trợ kiểm tra đa luồng.

#### **Các Annotations thông dụng**

Cách truyền thống để khai báo một test method trong Junit 3 là thêm tiền tố Test trong tên của method.

Nhìn thì có vẻ đây là cách tốt để phân biệt các method có vai trò đặc biệt, nhưng khi app mở rộng với muôn vàn các method khác nhau, muôn vàn các method đến từ nhiều thư viện khác nhau thì sao? Rõ ràng cách tỏ ra có khuyết điểm khi app càng mở rộng hơn.

Và sau đó sự ra đời của Anootations trong JDK 5 đã giải quyết được vấn đề này, và TestNG cũng đã tận dụng các annotations một cách triệt để.

|  |  |  |
| --- | --- | --- |
| # | Annotation | Description |
|  | **@BeforeSuite** |  |
|  | **@AfterSuite** |  |
|  | **@BeforeClass** |  |
|  | **@AfterClass** |  |
|  | **@BeforeTest** |  |
|  | **@AfterTest** |  |
|  | **@BeforeGroups** |  |
|  | **@AfterGroups** |  |
|  | **@BeforeMethod** |  |
|  | **@AfterMethod** |  |
|  | **@DataProvider** |  |
|  | **@Factory** |  |
|  | **@Listeners** |  |
|  | **@Parameters** |  |
|  | **@Test** |  |

#### **Suite Test**

#### **HTML, XML Reports**

### Chương II: Cấu hình selenium grid

1. **Selenium Grid**

Selenium-grid cho phép tester test với nhiều browser trên nhiều thiết bị (hệ điều hành) một cách đồng thời. Selenium sử dụng lý thuyết hub-node nơi bạn chỉ cần chạy test trên một máy duy nhất gọi là hub, nhưng tác vụ test sẽ được xử lý trên nhiều máy khác được liên kiết với hub được gọi là các node.

Hub nói một cách tổng quát - nó được xem như một bộ não trung tâm cho mọi hoạt động, và Selenium Grid Hub cũng hoạt động theo concept này. Selenium grid hub là điểm các đoạn testcase được khởi chạy và thì thực thi trên các node. Các node có thể là bất kì thiết bị nào chạy trên bất kì hệ điều hành nào (ios, window, linux … ). Khi một hub gọi một đoạn code test, các node thực thi đoạn code như một bản sao của hub.

1. **Cài đặt Selenium Grid Hub**

Bước 1: Tải thư viện .jar selenium grid:

Tải selenium-standalone-server.jar https://www.selenium.dev/downloads/ trên cả máy hub và máy node.

Bước 2: Khởi chạy hub bằng câu lênh:

java -jar <selenium\_stanalone\_server\_file\_name\_here> -role hub -host <hub\_ip\_here>

<selenium\_stanalone\_server\_file\_name\_here> : tên của file jar vừa tải về.

<hub\_id\_here>: địa chỉ của máy chủ (hub).

**Example: java -jar selenium-server-standalone-3.141.59.jar -role hub -host localhost**



*Lưu ý: máy phải cài Java JDK và cấu hình variable environment: JAVA\_HOME (đường dẫn đến thư mục cài JDK) và PATH (đường dẫn đến thư mục bin của JDK)*

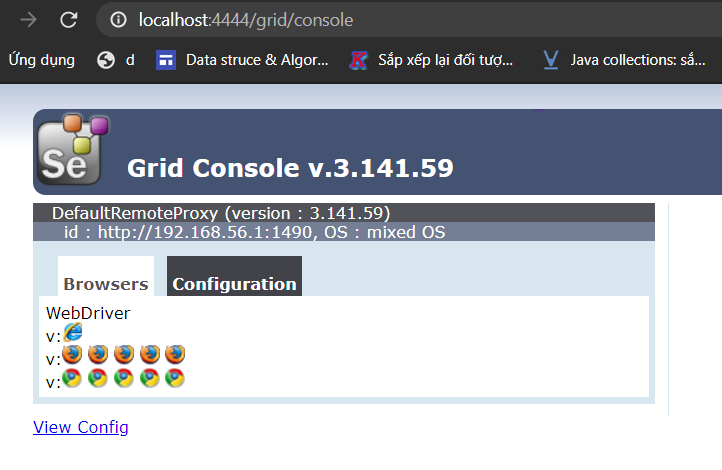
Bước 3: Khởi chạy các node:

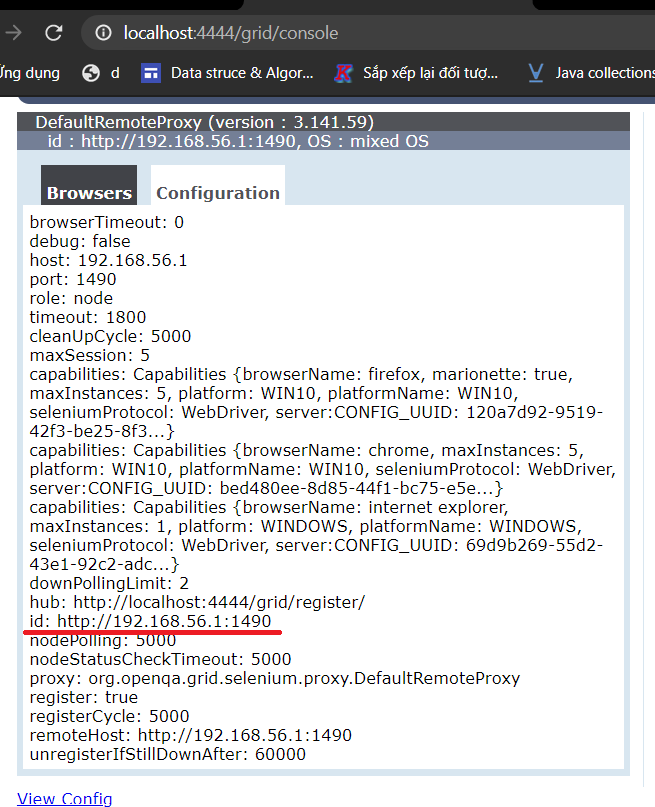
java -Dwebdriver.chrome.driver="chromedriver.exe" -Dwebdriver.internetexplorer.driver="IEDriverServer.exe" -Dwebdriver.gecko.driver="geckodriver.exe" -jar "selenium-server-standalone-3.141.59.jar" -role node -hub <hub\_register\_uri>

<hub\_register\_uri>: đường dẫn đăng ký node của máy hub.

**Example:**

**java -Dwebdriver.chrome.driver="chromedriver.exe" -Dwebdriver.internetexplorer.driver="IEDriverServer.exe" -Dwebdriver.gecko.driver="geckodriver.exe" -jar "selenium-server-standalone-3.141.59.jar" -role node -hub "http://localhost:4444/grid/register/"**





Ở tab **Configuration,** id chính là địa chỉ của máy node được đăng ký với máy hub, máy hub sẽ kết nối với máy node thông qua id này.

Bước 4: Viết class config cho cả project:

public class SetupDriver {

private WebDriver driver = null;

private String browser = null;

private String baseUrl = null;

private String os = null;

private String node = null;

public SetupDriver(String os, String browser, String baseUrl, String node) throws MalformedURLException {

this.browser = browser;

this.os = os;

this.baseUrl = baseUrl;

this.node = node;

Platform platform = Platform.fromString(os.toUpperCase());

if(browser.equalsIgnoreCase("chrome")) {

ChromeOptions chromeOptions = new ChromeOptions();

chromeOptions.setCapability("platform", platform);

this.driver = new RemoteWebDriver(new URL(node + "/wd/hub"), chromeOptions);

} else if (browser.equalsIgnoreCase("firefox")) {

FirefoxOptions firefoxOptions = new FirefoxOptions();

firefoxOptions.setCapability("platform", platform);

this.driver = new RemoteWebDriver(new URL(node + "/wd/hub"), firefoxOptions);

} else if (browser.equalsIgnoreCase("ie11")) {

InternetExplorerOptions ieOption = new InternetExplorerOptions();

ieOption.setCapability("platform", platform);

this.driver = new RemoteWebDriver(new URL(node + "/wd/hub"), ieOption);

}

this.driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);

this.driver.manage().window().maximize();

this.driver.get(baseUrl);

}

public String getOs() {

return this.os;

}

public String getBrowser() {

return this.browser;

}

public String getBaseUrl() {

return this.baseUrl;

}

public String getNode() {

return this.node;

}

public WebDriver getDriver() {

return this.driver;

}

}

Bước 5: Gọi class config trong các class chạy testcase:

public class GoogleSearchTest {

public WebDriver driver;

@BeforeClass(alwaysRun = true)

@Parameters({"os", "browser", "url", "node"})

public void setUp(String os, String browser, String url, String node) throws MalformedURLException {

System.out.println(os + browser + url + node);

SetupDriver setupTestDriver = new SetupDriver(os, browser, url, node);

driver = setupTestDriver.getDriver();

}

}

Bước 6: Viết file .xml để chạy test case trên nhiều driver cùng lúc:

<!--src/test/resources/launchers/google.xml-->

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

<suite name="Google Search Tests" verbose="1" parallel="tests">

<parameter name="url" value="https://www.google.com/"/>

<parameter name="os" value="windows"/>

<parameter name="node" value="http://192.168.56.1:1490"/>

<test name="Tests on Firefox" preserve-order="true" group-by-instances="true">

<parameter name="browser" value="firefox"/>

<classes>

<class name="tiktzuki.Junit.GoogleSearchTest"/>

</classes>

</test>

<test name="Tests on Chrome" preserve-order="true" group-by-instances="true">

<parameter name="browser" value="chrome"/>

<classes>

<class name="tiktzuki.Junit.GoogleSearchTest"/>

</classes>

</test>

<!--

<test name="Tests on IE11" preserve-order="true" group-by-instances="true">

<parameter name="browser" value="ie11"/>

<classes>

<class name="tiktzuki.Junit.GoogleSearchTest"/>

</classes>

</test> -->

</suite>

### Chương III. Tiến hành test app user manager

1. Đăng nhập

Mở trình duyệt và đi trến trang web “http://localhost:4200”

Kiểm tra title trang web

Nhập username và password.

Click vào nút “Login”

Kiểm tra đăng nhập thành công hay không  
Đóng trình duyệt

1. Đăng ký

Mở trình duyệt và đi trến trang web “<http://localhost:4200>”

Click vào nút “Create Account”

Điền thông tin user mới

Click vào nút “Sign up”

Kiểm tra tài khoản mới có tạo thành công hay không

Đóng trình duyệt

1. Thêm user

Mở trình duyệt và đi trến trang web “<http://localhost:4200>”

Đăng nhập vào trang web

Chọn nút “Create User” từ menu

Nhập thông tin user mới

Click vào nút submit

Kiểm tra user đã được tạo hay chưa

Đóng trình duyệt

1. Sửa thông tin user

Mở trình duyệt và đi trến trang web “<http://localhost:4200>”

Đăng nhập vào trang web

Chọn nút “User Manager” từ menu

Tìm user cần thay đổi thông tin và click nút “Modify”  
Thay đổi thông tin user

Click vào nút Submit

Nhấn vào nút “User Manage” trên menu

Tìm user vừa thay đổi thông tin và click nút “Modify”

Kiểm tra thông tin user đã được thay đổi

Đóng trình duyệt

1. Xóa user

Mở trình duyệt và đi trến trang web “<http://localhost:4200>”

Đăng nhập vào trang web

Chọn nút “User Manager” từ menu

Tìm user cần thay đổi thông tin và click nút “Delete”

Kiểm tra danh sách user

Đóng trình duyệt

1. Sửa thông tin cá nhân

Mở trình duyệt và đi trến trang web “<http://localhost:4200>”

Đăng nhập vào trang web

Chọn nút “Profile” từ menu

Thay đổi thông tin cá nhân

Click vào nút submit

Reload trang web

Kiểm tra thông tin user đã được thay đổi

Đóng trình duyệt

### Chương IV. TestCase: [1]

# **Bibliography**

|  |  |
| --- | --- |
| [1] | T.P.T.Long, "TestCase.xlsx," Sai Gon, 2021. |