**ASSIGNMENT 2.  
SELENIUM WEBDRIVER**

**I/ Yêu cầu**

1. Học viên đã có kiến thức cơ bản về java và TestNG, Selenium
2. Thực hành viết code automation theo TestCase được cho trước.
3. Thực hành với TestNG và đọc kết quả testcase
4. Lưu lại code.

**II. Thực hành**

**1.** Viết chương trình kiểm thử tính năng đăng nhập hệ thống với tập dữ liệu đầu vào cho trước:

* Gọi trình duyệt Google Chrome.
* Mở URL: [**https://automationexercise.com/products**](https://automationexercise.com/products)
* Đảm bảo người dùng chưa đăng nhập tài khoản.
* Lần lượt thực hiện đăng nhập với tập thông tin tài khoản được cung cấp trong file excel.<assignment2\_data\_test.xlsx>

Đảm bảo đăng nhập thành công với tài khoản trên.

* Thực hiện đăng xuất tài khoản trên nếu đăng nhập thành công.
* Thực hiện chụp ảnh màn hình nếu đăng nhập thất bại và tiến hành xuất ra báo cáo.

*Yêu cầu và gợi ý chi tiết:*

1. Yêu cầu tạo cấu trúc cho project:

- Tạo project có tên là: **ASSIGNMENT\_2\_SELENIUM**.  
- Tạo 1 thư mục có tên là **configuration**, bên trong tạo 1 properties file có tên là configs.properties

- Tạo một thư mục có tên là data, bên trong đặt file excel chứa data test được cung cấp sẵn: **assignment2\_data\_test.xlsx**  
- Tạo một thư mục có tên là **drive** chứa chrome driver  
Trong **src/test/ java**, tạo package “com.automation.testcase” chứa các class TestNG

Trong **src/main/ java**, tạo 3 package tương ứng dưới đây:

- com.automation.base : chứa các base class  
 - com.automation.utils : chứa các class hỗ trợ chụp ảnh màn hình, hỗ trợ đọc properties file ...  
 - com.automation.pom : chứa các class thực thi kiến trúc POM.  
 <Các bạn có thể xem lại bài học Data driven>.

1. Yêu cầu xác định trước các thành phần cần thao tác trên trang web và cài đặt vào properties file.

|  | icon\_signin\_... |
| --- | --- |
|  | icon\_signout\_... |
|  | login\_email\_…  login\_pasword\_…  login\_signin\_... |

3. Thêm các thư viện cần thiết vào file pom.xml   
 Selenium, testNG, ReportNG, Guice, POI

4. Gợi ý trong package : com.automation.utils  
 4.1 Thêm class: **CaptureScreenshot** có nhiệm vụ chụp ảnh màn hình khi test fail và lưu vào 1 thư mục có tên **screenshots.**Tên của ảnh chính là phần tên của email đăng nhập.  
Ví dụ: funixExample1@gmail.com → funixExample1.png

| **public** **class** CaptureScreenshot {  **public** **static** **void** takeScreenshot(WebDriver driver, String name) {  **try** {  *………..//code here……………..*  String imageName = name + ".png";  *………..//code here……………..*  File destiny = **new** File("./screenshots/" + imageName);  FileHandler.*copy*(source, destiny);  } **catch** (Exception ex) {  System.***out***.println("Đã xảy ra lỗi khi chụp màn hình!");  ex.printStackTrace();  }  } |
| --- |

4.2 Thêm class: **PropertiesFileUtils** có nhiệm vụ đọc **configs.proprties** file .

| **public** **class** PropertiesFileUtils {  //Đường dẫn đến properties files trong folder configuration  **private** **static** String *CONFIG\_PATH* = ………..  //2. Lấy ra giá trị property bất kỳ theo key.  **public** **static** String getProperty(String key) {  *………..//code here……………..*  } |
| --- |

Các bạn có thể xem lại <REF6\_Properties\_files.docx>

5. Gợi ý trong package : com.automation.base  
 Tạo class base : **DriverInstance.java**  
 Khai báo các phương thức sau:

| **public** **class** DriverInstance {  **protected** WebDriver driver;  @BeforeClass  **public** **void** initDriverInstance() {  // khởi tạo chrome driver   //và set max size cho cửa sổ trình duyệt  *………..//code here……………..*  }  @AfterClass  **public** **void** closeDriverInstance() {  //đóng trình duyệt  *………..//code here……………..*  }  } |
| --- |

Lý do ta cần đến class base là bởi vì với nhưng method mà ít thay đổi, có thể dùng chung, chúng ta chỉ cần khai báo 1 lần ở lớp cha (lớp base) không cần mất thời gian khai báo lại ở các lớp con, các lớp con mặc định có thể sử dụng được phương thức này. (Ví dụ ở đây là 2 phương thức được đánh @BeforeClass, @AfterClass)  
  
6. Gợi ý trong package : com.automation.pom  
 Viết class **LoginPage.java** và các method tương ứng thao tác với login.  
 Đọc properties file để lấy ra các id, name, xpath .. tương ứng với các thành phần mà chúng ta đã cài đặt tại bước #2.  
 Sử dụng Explicit wait với thời gian tải trang không quá 30s

| **public** **class** LoginPage {  **private** WebDriverWait wait;  **public** LoginPage(WebDriver driver) {  // **TODO** Auto-generated constructor stub  wait = **new** WebDriverWait(driver, 30);  }  **public** **void** enterEmail(String email) **throws** InterruptedException {  String elementIdentified = PropertiesFileUtils.getProperty(….);  WebElement inputEmail = wait.until(....)  *………..//code here……………..*  Thread.*sleep*(2000);  }  **public** **void** enterPassword(String password) **throws** InterruptedException {  *………..//code here……………..*  Thread.*sleep*(2000);  }  **public** **void** clickSignIn() **throws** InterruptedException {  *………..//code here……………..*  Thread.*sleep*(2000);  }  } |
| --- |

Sử dụng T**hreed.sleep()** để tạo khoảng nghỉ giữa các bước. đơn vị là miliseconds.

7. Trong package com.automation.testcase  
 Tạo class có tên là: **TC\_LoginTest** kế thừa từ **DriverInstance**  
 Sử dụng Data driven đọc data test từ file excel.  
 <Các bạn có thể xem lại phần này trong các video bào Data Driven>

| **public** **class** TC\_LoginTest **extends** DriverInstance {    @Test(dataProvider = "Excel")  **public** **void** TC01\_LoginFirstAccount(String email, String password) **throws** InterruptedException {  //Lấy URL từ properties file và tải trang.  ………..//code here……………..  WebDriverWait wait = **new** WebDriverWait(driver, 30);   //Lấy định danh iconSignin từ properties file và tìm kiếm, click  WebElement iconSignIn = wait.until………..  ………..  LoginPage loginPage = **new** LoginPage(driver);  PageFactory.initElements(driver, loginPage);    //Thực hiện đăng nhập qua loginPage.  ………..//code here……………..  //Lấy định danh iconSignout từ properties,  // kiểm tra SignOut có hiển thị ko, nếu hiển thị thì click  WebElement iconSignout = wait.until………..  ………..//code here……………..  Thread.sleep(2000);  }    @DataProvider(name="Excel")  **public** Object[][] testDataGenerator() **throws** IOException {  //đọc dữ liệu đầu vào từ file excel  **return** data;    }  @AfterMethod  **public** **void** takeScreenshot(ITestResult result) **throws** InterruptedException {  //ITestResult để lấy trạng thái và tên và tham số của từng Test Case  // phương thức này sẽ được gọi mỗi khi @Test thực thi xong,  // ta có thể kiểm tra kết quả testcase tại đây.  **if** (ITestResult.**FAILURE** == result.getStatus()) {  **try** {  //1. Lấy tham số (parameters) đầu vào của TC vừa chạy  //email:0, password:1  String email = (String)result.getParameters()[0];  //2. Lấy ra phần tên trong email để làm tên của screenshot  // Tìm vị trí(int index) của ký tự ‘@’ và lấy ra chuỗi con  // đứng trước @ qua thư viện của String là: indexOf() và subString()  ………..//code here……………..  **int** index = …..  String accountName = ….   ……………………………………………………...  } **catch** (Exception e) {  System.**out**.println("Lỗi xảy ra screenshot " + e.getMessage());  }  }  } |
| --- |

8. Tạo testSuite cho testNG và tích hợp ReportNG,  
 file **testng.xml** sẽ được đặt trong src/test/java