**Hướng dẫn kiểm thử tự động**

**Selenium WebDriver với JUnit 5**

**Áp dụng trình bày Presentation:**

**By traltb@fe.edu.vn**

**Bước 1:** Tạo Project Maven tên Exercise1\_SeleniumBasic và thêm dependency vào file pom

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>phuctv</groupId>  
 <artifactId>SeleniumDemo</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 </properties>  
  
 <dependencies>  
 <!-- Selenium Java -->  
 <dependency>  
 <groupId>org.seleniumhq.selenium</groupId>  
 <artifactId>selenium-java</artifactId>  
 <version>4.21.0</version>  
 </dependency>  
  
 <!-- JUnit 5 -->  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-engine</artifactId>  
 <version>5.10.2</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- JUnit Jupiter Params for parameterized tests -->  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-params</artifactId>  
 <version>5.10.2</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- WebDriverManager -->  
 <dependency>  
 <groupId>io.github.bonigarcia</groupId>  
 <artifactId>webdrivermanager</artifactId>  
 <version>5.7.0</version>  
 </dependency>  
 </dependencies>  
</project>

Ghi chú: webdrivermanager sẽ tự động:

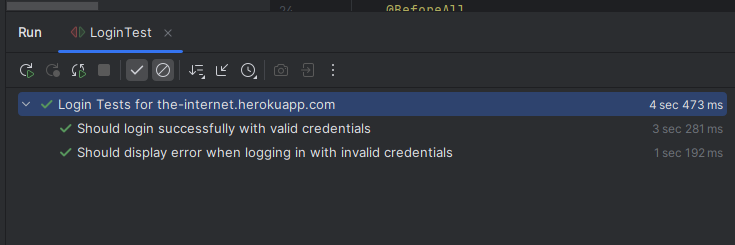
* Tải đúng driver tương thích với trình duyệt
* Thiết lập path driver tự động
* Hỗ trợ Chrome, Firefox, Edge, Opera...

Link to get webdrivermanager dependency:   
<https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager/5.8.0>

Bước 2: Tạo file LoginTest.java tại thư mục src/test/java

import io.github.bonigarcia.wdm.WebDriverManager;  
import org.junit.jupiter.api.\*;  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.WebElement;  
import org.openqa.selenium.chrome.ChromeDriver;  
import org.openqa.selenium.chrome.ChromeOptions;  
import org.openqa.selenium.support.ui.ExpectedConditions;  
import org.openqa.selenium.support.ui.WebDriverWait;  
  
import java.time.Duration;  
import java.util.HashMap;  
import java.util.Map;  
  
import static org.junit.jupiter.api.Assertions.*assertTrue*;  
  
  
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)  
@DisplayName("Login Tests for the-internet.herokuapp.com")  
 public class LoginTest {  
 static WebDriver *driver*;  
 static WebDriverWait *wait*;  
  
 @BeforeAll  
 static void setUp() {  
 WebDriverManager.*chromedriver*().setup(); // Tự động tải và cấu hình  
  
 //disable JavaScript in Chrome  
 ChromeOptions options = new ChromeOptions();  
 Map<String, Object> prefs = new HashMap<>();  
 prefs.put("profile.managed\_default\_content\_settings.javascript", 2); //1: Cho phép (default), 2: Chặn JavaScript  
 options.setExperimentalOption("prefs", prefs);  
 options.addArguments("--incognito"); // Ẩn danh  
  
 *driver* = new ChromeDriver(options);  
 *wait* = new WebDriverWait(*driver*, Duration.*ofSeconds*(10)); // Thêm WebDriverWait  
 *driver*.manage().window().maximize();  
 }  
  
 @Test  
 @Order(1)  
 @DisplayName("Should login successfully with valid credentials")  
 void testLoginSuccess() {  
 *driver*.get("https://the-internet.herokuapp.com/login");  
  
 *driver*.findElement(By.*id*("username")).sendKeys("tomsmith");  
 *driver*.findElement(By.*id*("password")).sendKeys("SuperSecretPassword!");  
 *driver*.findElement(By.*cssSelector*("button[type='submit']")).click();  
 // Chờ thông báo thành công hiển thị  
 WebElement successMsg = *wait*.until(  
 ExpectedConditions.*visibilityOfElementLocated*(By.*cssSelector*(".flash.success"))  
 );  
  
  
 //String successMsg = driver.findElement(By.cssSelector(".flash.success")).getText();  
 *assertTrue*(successMsg.getText().contains("You logged into a secure area!"));  
 }  
  
 @Test  
 @Order(2)  
 @DisplayName("Should display error when logging in with invalid credentials")  
 void testLoginFail() {  
 *driver*.get("https://the-internet.herokuapp.com/login");  
  
 *driver*.findElement(By.*id*("username")).sendKeys("invalid");  
 *driver*.findElement(By.*id*("password")).sendKeys("wrongpassword");  
 *driver*.findElement(By.*cssSelector*("button[type='submit']")).click();  
  
 // Chờ thông báo lỗi hiển thị  
 WebElement errorMsg = *wait*.until(  
 ExpectedConditions.*visibilityOfElementLocated*(By.*cssSelector*(".flash.error"))  
 );  
  
 *assertTrue*(errorMsg.getText().contains("Your username is invalid!"));  
 }  
  
 @AfterAll  
 static void tearDown() {  
 *driver*.quit();  
 }  
}

Kết quả chạy chương trình:



Nội dung LÝ THUYẾT CẦN NẮM:

# Tổng quan về Selenium WebDriver

**Selenium WebDriver** là một thư viện mã nguồn mở dùng để tự động hóa trình duyệt web, hỗ trợ nhiều ngôn ngữ (Java, Python, C#, JS...) và trình duyệt (Chrome, Firefox, Edge...).

Thành phần chính trong 1 Test Script Selenium:

* WebDriver: Đại diện cho một element trên trang web (button, input, v.v)
* Locator: Dùng để tìm element: By.id, By.cssSelector, By.xpath, By.name, By.className,…
* Actions: Các hành vi: .click(), .sendKeys(), .getText()
* Waits: Chờ element có mặt: WebDriverWait, ExpectedConditions. Gồm có :
* Implicit Wait: áp dụng toàn cục:

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

* ExplicitWait: nên ưu tiên để tránh test fail khi trang load chậm

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));  
wait.until(ExpectedConditions.visibilityOf(...))

* Fluent Wait: nâng cao hơn, cho phép xử lý polling, exception.
* ChromeOptions: Tùy chỉnh trình duyệt khi khởi tạo (ẩn danh, tắt JS, headless...): các options có thể có như:
* --incognito: Mở trình duyệt ở chế độ ẩn danh
* --headless: Chạy không hiển thị UI (dùng trong CI/CD)
* prefs.put("profile.managed\_default\_content\_settings.javascript", 2): Tắt javascript

# Một số lưu ý khi test validation form:

* **Sau khi submit**, luôn **chờ element hiển thị (dùng wait)** rồi mới .getText() hoặc .click()
* Kiểm tra **nội dung thông báo** bằng assertTrue(msg.getText().contains(...))
* Nên viết thêm test cho:
  + Trường bỏ trống
  + Sai định dạng email, số điện thoại
  + Password quá ngắn / dài
* Ưu tiên **negative test cases** để đảm bảo form hoạt động tốt

**Bài tập 1:** Áp dụng @ParameterizedTest

Dùng project ở trên, tiếp tục áp dụng kiểm thử đăng nhập với nhiều bộ dữ liệu, cần sử dụng parameterized test.

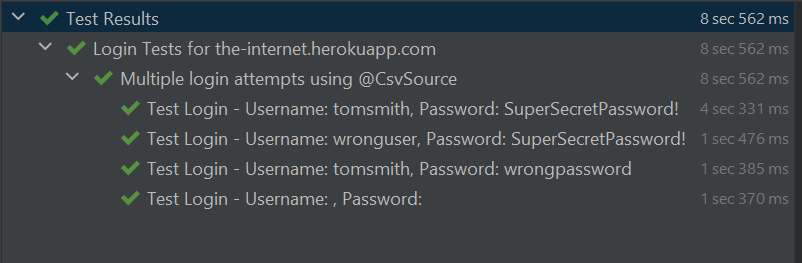
Bước 1: Thêm dependency nếu chưa có (nằm trong JUnit 5):

<!-- JUnit 5 Parameterized Test -->  
<dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-params</artifactId>  
 <version>5.10.2</version>  
 <scope>test</scope>  
</dependency>

Bước 2: Sử dụng @ParameterizedTest + @CsvSource

@Order(3)  
@ParameterizedTest(name = "Test Login - Username: {0}, Password: {1}")  
@CsvSource({  
 "tomsmith, SuperSecretPassword!, success", // valid  
 "wronguser, SuperSecretPassword!, error", // invalid username  
 "tomsmith, wrongpassword, error", // invalid password  
 "'', '', error" // empty credentials  
})  
@DisplayName("Multiple login attempts using @CsvSource")  
void testLoginWithMultipleParameters(String username, String password, String expectedResult) {  
 *driver*.get("https://the-internet.herokuapp.com/login");  
  
 *driver*.findElement(By.*id*("username")).sendKeys(username);  
 *driver*.findElement(By.*id*("password")).sendKeys(password);  
 *driver*.findElement(By.*cssSelector*("button[type='submit']")).click();  
  
 By messageLocator = expectedResult.equals("success")  
 ? By.*cssSelector*(".flash.success")  
 : By.*cssSelector*(".flash.error");  
  
 WebElement message = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(messageLocator));  
  
 if (expectedResult.equals("success")) {  
 *assertTrue*(message.getText().contains("You logged into a secure area!"));  
 } else {  
 *assertTrue*(message.getText().toLowerCase().contains("invalid"));  
 }  
}

Kết quả chạy test:



**Bài tập 2: Đọc dữ liệu test từ file CSV bên ngoài**

Bước 1: Tạo file .csv

File login-data.csv phải nằm trong src/test/resources để JUnit tìm thấy.

Click phải vào thư mục test, New->Directory, chọn **resources**

Click phải vào thư mục này, New->File.

Đặt tên file login-data.csv

Lưu ý: có thể bạn cần phải cài đặt plugin csv trong Intellij nếu đuôi file này không hỗ trợ.

Bằng cách, File->Setting->Plugin, gõ CSV Editor, nhấn Install sau đó Restart lại IDE.

**Bước 2**: Thêm nội dung vào file login-data.csv

username,password,expectedResult  
tomsmith,SuperSecretPassword!,success  
wronguser,SuperSecretPassword!,error  
tomsmith,wrongpassword,error  
, ,error

Lưu ý: bạn phải dùng "" để biểu diễn chuỗi rỗng đúng cách (không phải khoảng trắng).

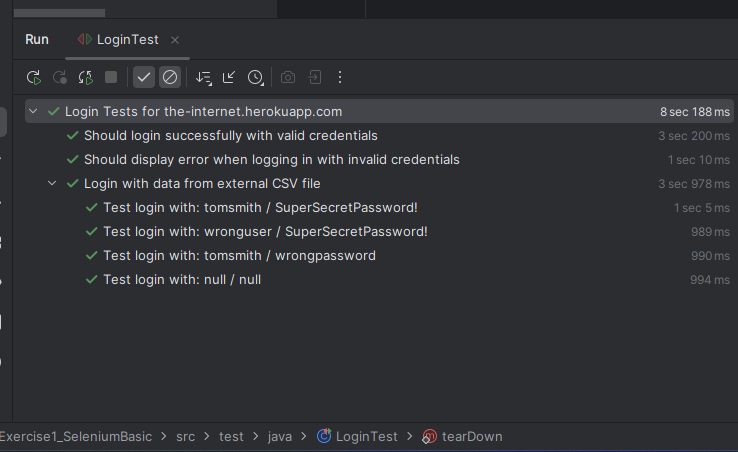
**Bước 3:**

Lưu ý:

Bạn cần thêm xử lý trim() để tránh lỗi khi người dùng copy paste dữ liệu CSV có khoảng trắng:

@ParameterizedTest(name = "Test login with: {0} / {1}")  
@CsvFileSource(resources = "/login-data.csv", numLinesToSkip = 1)  
@DisplayName("Login with data from external CSV file")  
void testLoginWithCSV(String username, String password, String expectedResult) {  
 *driver*.get("https://the-internet.herokuapp.com/login");  
 // Chuyển null thành chuỗi rỗng nếu cần  
 username = (username == null) ? "" : username.trim();  
 password = (password == null) ? "" : password.trim();  
  
 *driver*.findElement(By.*id*("username")).sendKeys(username);  
 *driver*.findElement(By.*id*("password")).sendKeys(password);  
 *driver*.findElement(By.*cssSelector*("button[type='submit']")).click();  
  
 By messageLocator = expectedResult.equals("success")  
 ? By.*cssSelector*(".flash.success")  
 : By.*cssSelector*(".flash.error");  
  
 WebElement message = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(messageLocator));  
  
 if (expectedResult.equals("success")) {  
 *assertTrue*(message.getText().contains("You logged into a secure area!"));  
 } else {  
 *assertTrue*(message.getText().toLowerCase().contains("invalid"));  
 }  
}

**Kết quả test:**



**Bài tập 3: Tổ chức dự án theo mô hình Page Object Model**

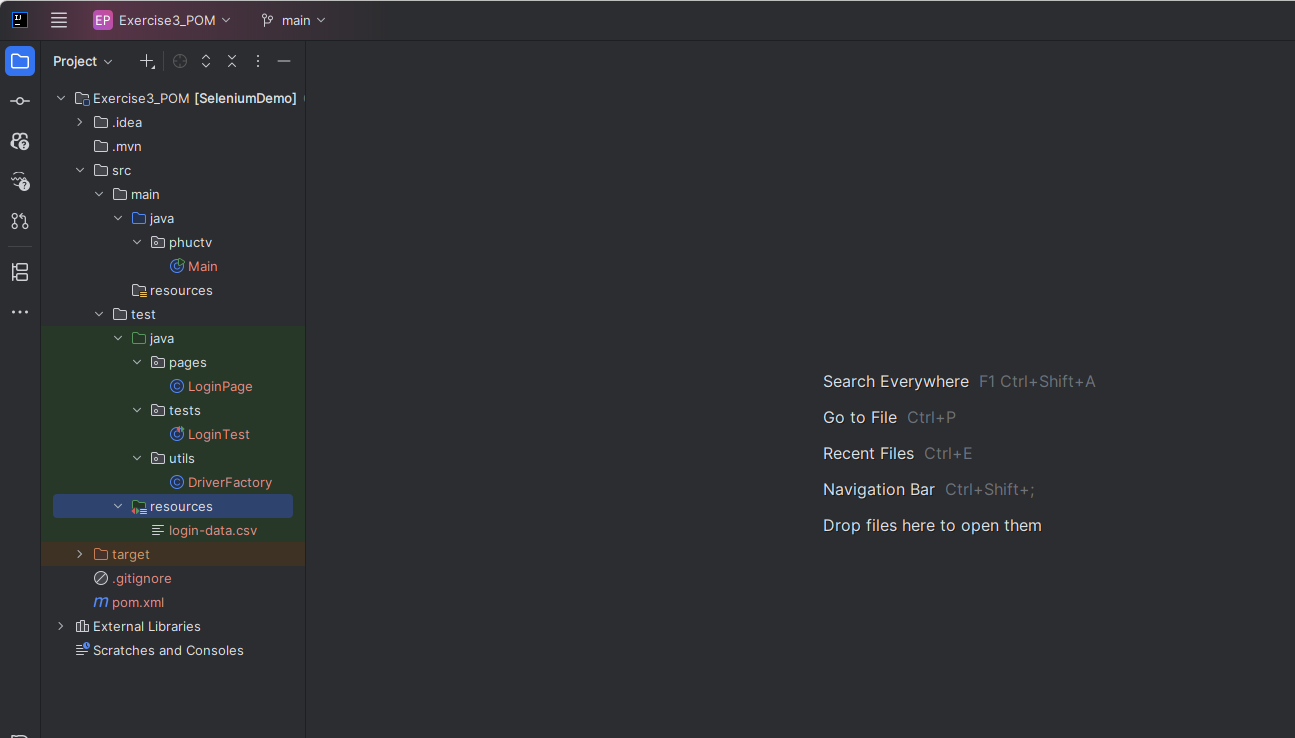
**POM (Page Object Model)** là mô hình thiết kế giúp:

* Tách riêng logic test và logic UI: Mỗi trang là 1 class riêng
* Dễ bảo trì: Nếu UI thay đổi, chỉ sửa tại 1 class
* Dễ tái sử dụng: Dùng lại nhiều lần trong test khác
* Có cấu trúc rõ ràng: Giúp teamwork dễ đọc, CI/CD tốt hơn

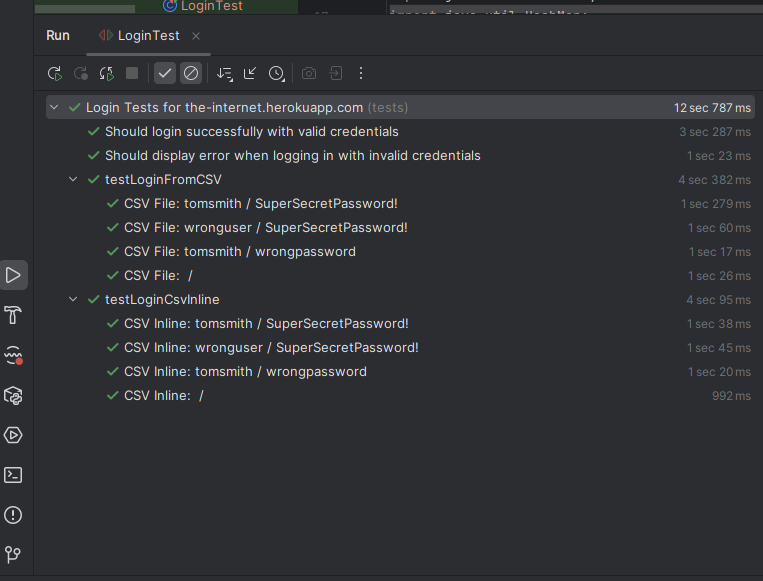
**Copy lại project trên, đặt tên thành dự án khác, tên Exercise3\_POM**

**Bước 1: Cấu trúc dự án như sau:**

****



**Kết quả chạy:**



**Bước 2: Cài đặt từng thành phần**

1. DriverFactory.java – Khởi tạo và cấu hình WebDriver, dùng chung

package utils;  
  
import io.github.bonigarcia.wdm.WebDriverManager;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.chrome.ChromeDriver;  
import org.openqa.selenium.chrome.ChromeOptions;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public class DriverFactory {  
 public static WebDriver createDriver() {  
 WebDriverManager.*chromedriver*().setup();  
  
 ChromeOptions options = new ChromeOptions();  
 Map<String, Object> prefs = new HashMap<>();  
 prefs.put("profile.managed\_default\_content\_settings.javascript", 2);  
 options.setExperimentalOption("prefs", prefs);  
 options.addArguments("--incognito");  
  
 return new ChromeDriver(options);  
 }  
}

1. LoginPage.java – Page Object đại diện cho trang login

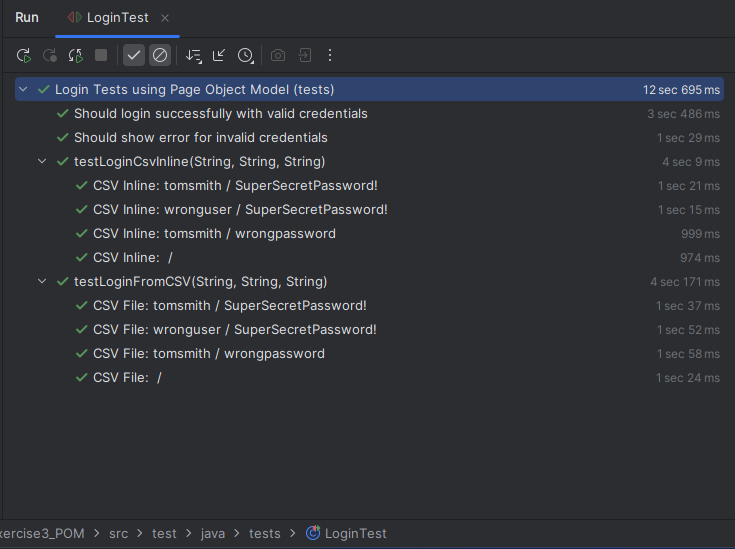
package pages;  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.WebElement;  
  
public class LoginPage {  
 private WebDriver driver;  
  
 // Constructor  
 public LoginPage(WebDriver driver) {  
 this.driver = driver;  
 }  
  
 // Locators  
 private By usernameField = By.*id*("username");  
 private By passwordField = By.*id*("password");  
 private By loginButton = By.*cssSelector*("button[type='submit']");  
 private By successMsg = By.*cssSelector*(".flash.success");  
 private By errorMsg = By.*cssSelector*(".flash.error");  
  
 // Actions  
 public void navigate() {  
 driver.get("https://the-internet.herokuapp.com/login");  
 }  
  
 public void login(String username, String password) {  
 driver.findElement(usernameField).sendKeys(username);  
 driver.findElement(passwordField).sendKeys(password);  
 driver.findElement(loginButton).click();  
 }  
  
 public By getSuccessLocator() {  
 return successMsg;  
 }  
  
 public By getErrorLocator() {  
 return errorMsg;  
 }  
}

1. LoginTest.java – Chuyển test hiện tại sang sử dụng LoginPage

package tests;  
  
import org.junit.jupiter.api.\*;  
import org.junit.jupiter.params.ParameterizedTest;  
import org.junit.jupiter.params.provider.CsvFileSource;  
import org.junit.jupiter.params.provider.CsvSource;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.support.ui.\*;  
import pages.LoginPage;  
import utils.DriverFactory;  
  
import java.time.Duration;  
  
import static org.junit.jupiter.api.Assertions.*assertTrue*;  
  
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)  
@DisplayName("Login Tests using Page Object Model")  
public class LoginTest {  
 static WebDriver *driver*;  
 static WebDriverWait *wait*;  
 static LoginPage *loginPage*;  
  
 @BeforeAll  
 static void setUp() {  
 *driver* = DriverFactory.*createDriver*();  
 *wait* = new WebDriverWait(*driver*, Duration.*ofSeconds*(10));  
 *driver*.manage().window().maximize();  
 *loginPage* = new LoginPage(*driver*);  
 }  
  
 @Test  
 @Order(1)  
 @DisplayName("Should login successfully with valid credentials")  
 void testLoginSuccess() {  
 *loginPage*.navigate();  
 *loginPage*.login("tomsmith", "SuperSecretPassword!");  
 WebElement success = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(*loginPage*.getSuccessLocator()));  
 *assertTrue*(success.getText().contains("You logged into a secure area!"));  
 }  
  
 @Test  
 @Order(2)  
 @DisplayName("Should show error for invalid credentials")  
 void testLoginFail() {  
 *loginPage*.navigate();  
 *loginPage*.login("wronguser", "wrongpassword");  
 WebElement error = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(*loginPage*.getErrorLocator()));  
 *assertTrue*(error.getText().toLowerCase().contains("invalid"));  
 }  
  
 @ParameterizedTest(name = "CSV Inline: {0} / {1}")  
 @Order(3)  
 @CsvSource({  
 "tomsmith, SuperSecretPassword!, success",  
 "wronguser, SuperSecretPassword!, error",  
 "tomsmith, wrongpassword, error",  
 "'', '', error"  
 })  
 void testLoginCsvInline(String username, String password, String expected) {  
 *loginPage*.navigate();  
 username = (username == null) ? "" : username.trim();  
 password = (password == null) ? "" : password.trim();  
  
 *loginPage*.login(username, password);  
 By resultLocator = expected.equals("success") ? *loginPage*.getSuccessLocator() : *loginPage*.getErrorLocator();  
 WebElement result = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(resultLocator));  
  
 if (expected.equals("success")) {  
 *assertTrue*(result.getText().contains("You logged into a secure area!"));  
 } else {  
 *assertTrue*(result.getText().toLowerCase().contains("invalid"));  
 }  
 }  
  
 @ParameterizedTest(name = "CSV File: {0} / {1}")  
 @Order(4)  
 @CsvFileSource(resources = "/login-data.csv", numLinesToSkip = 1)  
 void testLoginFromCSV(String username, String password, String expected) {  
 *loginPage*.navigate();  
 username = (username == null) ? "" : username.trim();  
 password = (password == null) ? "" : password.trim();  
  
 *loginPage*.login(username, password);  
 By resultLocator = expected.equals("success") ? *loginPage*.getSuccessLocator() : *loginPage*.getErrorLocator();  
 WebElement result = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(resultLocator));  
  
 if (expected.equals("success")) {  
 *assertTrue*(result.getText().contains("You logged into a secure area!"));  
 } else {  
 *assertTrue*(result.getText().toLowerCase().contains("invalid"));  
 }  
 }  
  
 @AfterAll  
 static void tearDown() {  
 *driver*.quit();  
 }  
}

**Lưu ý:**

* Đừng thao tác DOM trực tiếp trong test class (driver.findElement...)  
  → luôn thông qua Page Object
* Tránh dùng Thread.sleep() → hãy dùng WebDriverWait
* Với test form, nên kiểm tra cả trường hợp:
  + Trống
  + Sai định dạng
  + Đúng nhưng chưa submit

**Kết quả test:** 

**Bài tập 4: Làm bài tập nâng cao. Xây dựng lớp BasePage và BaseTest**

1. **Xây dựng lớp BasePage**

**BasePage.java** — một phần quan trọng trong mô hình **Page Object Model** giúp:

* Tái sử dụng code thao tác trình duyệt: Chứa WebDriver, WebDriverWait,..để mọi trang con đều có thể dùng lại.
* Quản lý chung các thao tác như: click, sendKeys, wait, scrollToElement(), hoverOverElement(), isTextPresent(),…
* Làm lớp cha (superclass) cho các trang như LoginPage, RegisterPage, v.v…đảm bảo tính kế thừa.

**Copy lại project trên, đặt tên thành dự án khác, tên Exercise4\_BasePage**

Bước 1**:** Tạo file mã nguồn BasePage.java trong thư mục pages

package pages;  
  
import org.openqa.selenium.\*;  
import org.openqa.selenium.support.ui.\*;  
  
import java.time.Duration;  
  
public class BasePage {  
 protected WebDriver driver;  
 protected WebDriverWait wait;  
  
 public BasePage(WebDriver driver) {  
 this.driver = driver;  
 this.wait = new WebDriverWait(driver, Duration.*ofSeconds*(10));  
 }  
  
 // Wait for visibility  
 protected WebElement waitForVisibility(By locator) {  
 return wait.until(ExpectedConditions.*visibilityOfElementLocated*(locator));  
 }  
  
 // Click safely  
 protected void click(By locator) {  
 waitForVisibility(locator).click();  
 }  
  
 // Send keys safely  
 protected void type(By locator, String text) {  
 WebElement element = waitForVisibility(locator);  
 element.clear();  
 element.sendKeys(text);  
 }  
  
 // Get text safely  
 protected String getText(By locator) {  
 return waitForVisibility(locator).getText();  
 }  
  
 // Navigate to a URL  
 public void navigateTo(String url) {  
 driver.get(url);  
 }  
  
 // Check if element is present  
 protected boolean isElementVisible(By locator) {  
 try {  
 return waitForVisibility(locator).isDisplayed();  
 } catch (TimeoutException e) {  
 return false;  
 }  
 }  
}

Bước 2: Cập nhật file LoginPage.java để kế thừa BasePage

package pages;  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
  
public class LoginPage extends BasePage {  
  
 public LoginPage(WebDriver driver) {  
 super(driver);  
 }  
  
 // Locators  
 private By usernameField = By.*id*("username");  
 private By passwordField = By.*id*("password");  
 private By loginButton = By.*cssSelector*("button[type='submit']");  
 private By successMsg = By.*cssSelector*(".flash.success");  
 private By errorMsg = By.*cssSelector*(".flash.error");  
  
 // Actions  
 public void navigate() {  
 navigateTo("https://the-internet.herokuapp.com/login");  
 }  
  
 public void login(String username, String password) {  
 type(usernameField, username);  
 type(passwordField, password);  
 click(loginButton);  
 }  
  
 public By getSuccessLocator() {  
 return successMsg;  
 }  
  
 public By getErrorLocator() {  
 return errorMsg;  
 }  
  
 public String getMessageText(By locator) {  
 return getText(locator);  
 }  
}

1. **Xây dựng lớp BaseTest**

Tạo lớp BaseTest.java để các lớp test kế thừa, tái sử dụng lại, giúp cho việc test các lớp gọn gàng hơn, chỉ tập trung vào logic kiểm thử.

* Tạo @BeforeAll để khởi tạo driver
* Tạo @AfterAll để quit driver
* Cung cấp protected WebDriver để test class con dùng được
* Kết hợp DriverFactory để tùy biến trình duyệt nếu cần

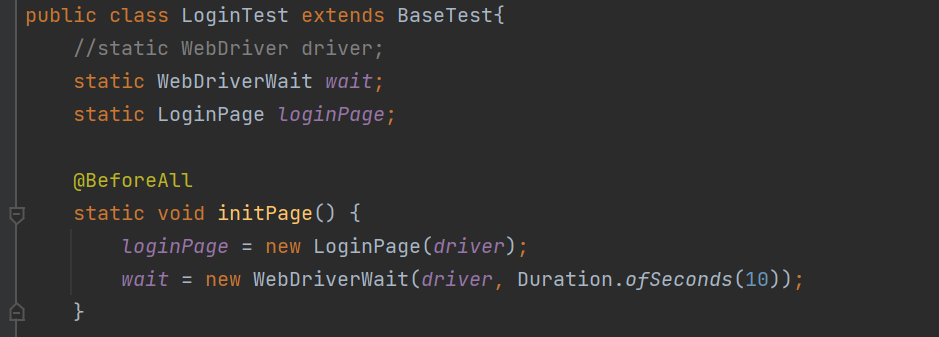
Bước 1: Tạo lớp BaseTest trong thư mục tests

package tests;  
  
import org.junit.jupiter.api.AfterAll;  
import org.junit.jupiter.api.BeforeAll;  
import org.openqa.selenium.WebDriver;  
import utils.DriverFactory;  
  
public abstract class BaseTest {  
 protected static WebDriver *driver*;  
  
 @BeforeAll  
 public static void setUpBase() {  
 *driver* = DriverFactory.*createDriver*();  
 *driver*.manage().window().maximize();  
 }  
  
 @AfterAll  
 public static void tearDownBase() {  
 if (*driver* != null) {  
 *driver*.quit();  
 }  
 }  
}

Lưu ý:

* Khai báo từ khóa abstract cho lớp BaseTest để tránh chạy nhầm test.
* Bạn có thể mở rộng các annotation @BeforeEach, @AfterEach,…để tích hợp để ghi lại log mỗi test start

**Bước 2:** Chỉnh sửa lớp LoginTest.java kế thừa từ BaseTest

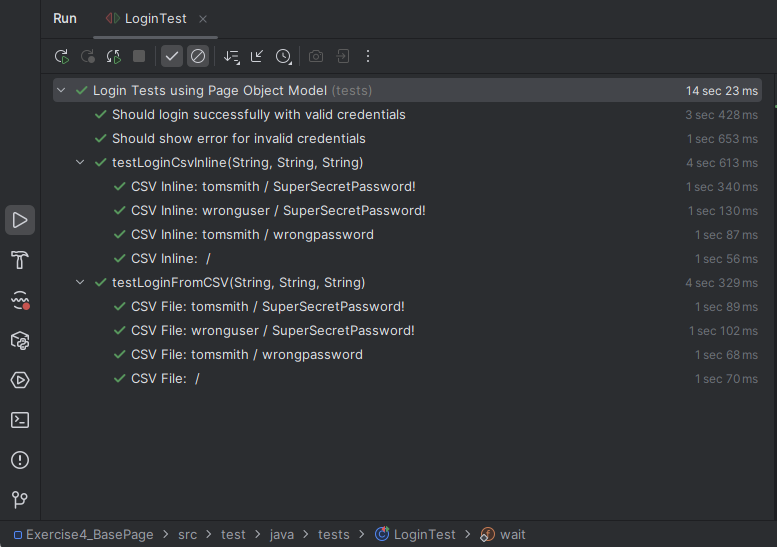


Đảm bảo tính kế thừa

Code đầy đủ lớp LoginTest.java

package tests;  
  
import org.junit.jupiter.api.\*;  
import org.junit.jupiter.params.ParameterizedTest;  
import org.junit.jupiter.params.provider.CsvFileSource;  
import org.junit.jupiter.params.provider.CsvSource;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.support.ui.\*;  
import pages.LoginPage;  
import utils.DriverFactory;  
  
import java.time.Duration;  
  
import static org.junit.jupiter.api.Assertions.*assertTrue*;  
  
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)  
@DisplayName("Login Tests using Page Object Model")  
public class LoginTest extends BaseTest{  
 //static WebDriver driver;  
 static WebDriverWait *wait*;  
 static LoginPage *loginPage*;  
  
 @BeforeAll  
 static void initPage() {  
 *loginPage* = new LoginPage(*driver*);  
 *wait* = new WebDriverWait(*driver*, Duration.*ofSeconds*(10));  
 }  
  
 @Test  
 @Order(1)  
 @DisplayName("Should login successfully with valid credentials")  
 void testLoginSuccess() {  
 *loginPage*.navigate();  
 *loginPage*.login("tomsmith", "SuperSecretPassword!");  
 WebElement success = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(*loginPage*.getSuccessLocator()));  
 *assertTrue*(success.getText().contains("You logged into a secure area!"));  
 }  
  
 @Test  
 @Order(2)  
 @DisplayName("Should show error for invalid credentials")  
 void testLoginFail() {  
 *loginPage*.navigate();  
 *loginPage*.login("wronguser", "wrongpassword");  
 WebElement error = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(*loginPage*.getErrorLocator()));  
 *assertTrue*(error.getText().toLowerCase().contains("invalid"));  
 }  
  
 @ParameterizedTest(name = "CSV Inline: {0} / {1}")  
 @Order(3)  
 @CsvSource({  
 "tomsmith, SuperSecretPassword!, success",  
 "wronguser, SuperSecretPassword!, error",  
 "tomsmith, wrongpassword, error",  
 "'', '', error"  
 })  
 void testLoginCsvInline(String username, String password, String expected) {  
 *loginPage*.navigate();  
 username = (username == null) ? "" : username.trim();  
 password = (password == null) ? "" : password.trim();  
  
 *loginPage*.login(username, password);  
 By resultLocator = expected.equals("success") ? *loginPage*.getSuccessLocator() : *loginPage*.getErrorLocator();  
 WebElement result = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(resultLocator));  
  
 if (expected.equals("success")) {  
 *assertTrue*(result.getText().contains("You logged into a secure area!"));  
 } else {  
 *assertTrue*(result.getText().toLowerCase().contains("invalid"));  
 }  
 }  
  
 @ParameterizedTest(name = "CSV File: {0} / {1}")  
 @Order(4)  
 @CsvFileSource(resources = "/login-data.csv", numLinesToSkip = 1)  
 void testLoginFromCSV(String username, String password, String expected) {  
 *loginPage*.navigate();  
 username = (username == null) ? "" : username.trim();  
 password = (password == null) ? "" : password.trim();  
  
 *loginPage*.login(username, password);  
 By resultLocator = expected.equals("success") ? *loginPage*.getSuccessLocator() : *loginPage*.getErrorLocator();  
 WebElement result = *wait*.until(ExpectedConditions.*visibilityOfElementLocated*(resultLocator));  
  
 if (expected.equals("success")) {  
 *assertTrue*(result.getText().contains("You logged into a secure area!"));  
 } else {  
 *assertTrue*(result.getText().toLowerCase().contains("invalid"));  
 }  
 }  
  
 @AfterAll  
 static void tearDown() {  
 *driver*.quit();  
 }  
}

**Kết quả chạy:**



**Bài tập 5: Dựa trên project này, tiếp tục Kiểm thử trang đăng ký ở link sau:**

<https://demoqa.com/automation-practice-form>

**Áp dụng POM, xây dựng các lớp, kế thừa từ BasePage, BaseTest.**

**B1:**

URL: https://demoqa.com/automation-practice-form

Các trường cần điền: First Name, Last Name, Email, Gender, Mobile...

Các kiểu input: text, radio, checkbox, date picker, file upload...

**B2:**

Kế thừa từ BasePage và BaseTest có sẵn

**Tạo PracticeFormPage.java kế thừa BasePage**

- Tạo lớp xử lý trang form với các thao tác điền dữ liệu, chọn radio, checkbox, upload file

package pages;  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.Keys;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.support.ui.ExpectedConditions;  
import org.openqa.selenium.support.ui.WebDriverWait;  
  
import java.time.Duration;  
  
public class PracticeFormPage extends BasePage {  
  
 public PracticeFormPage(WebDriver driver) {  
 super(driver);  
 wait = new WebDriverWait(driver, Duration.*ofSeconds*(10));  
 }  
  
 // Locators  
 private final By firstNameInput = By.*id*("firstName");  
 private final By lastNameInput = By.*id*("lastName");  
 private final By emailInput = By.*id*("userEmail");  
 private final By genderMale = By.*xpath*("//label[text()='Male']");  
 private final By mobileInput = By.*id*("userNumber");  
 private final By dobInput = By.*id*("dateOfBirthInput");  
 private final By subjectInput = By.*id*("subjectsInput");  
 private final By hobbySports = By.*xpath*("//label[text()='Sports']");  
 private final By uploadPicture = By.*id*("uploadPicture");  
 private final By addressInput = By.*id*("currentAddress");  
 private final By stateInput = By.*id*("react-select-3-input");  
 private final By cityInput = By.*id*("react-select-4-input");  
 private final By submitBtn = By.*id*("submit");  
 private final By modalTitle = By.*id*("example-modal-sizes-title-lg");  
  
 public void openFormPage() {  
 navigateTo("https://demoqa.com/automation-practice-form");  
 }  
  
 public void fillForm() {  
 type(firstNameInput, "Nguyen");  
 type(lastNameInput, "Van A");  
 type(emailInput, "vana@example.com");  
 click(genderMale);  
 type(mobileInput, "0912345678");  
  
 // Date of birth  
 click(dobInput);  
 driver.findElement(dobInput).sendKeys(Keys.*CONTROL* + "a");  
 driver.findElement(dobInput).sendKeys("22 Jul 1995");  
 driver.findElement(dobInput).sendKeys(Keys.*ENTER*);  
  
 // Subject  
 type(subjectInput, "Maths");  
 driver.findElement(subjectInput).sendKeys(Keys.*ENTER*);  
  
 click(hobbySports);  
  
 // Upload picture using absolute path  
 String imagePath = "C:\\Users\\ngung\\OneDrive\\Máy tính\\SWT\\Exercise4\_BasePage\\target\\test-classes\\test-pic.png";  
 driver.findElement(uploadPicture).sendKeys(imagePath);  
  
 type(addressInput, "123 Nguyen Van Cu, Hanoi");  
  
 // Select state and city  
 type(stateInput, "NCR");  
 driver.findElement(stateInput).sendKeys(Keys.*ENTER*);  
 type(cityInput, "Delhi");  
 driver.findElement(cityInput).sendKeys(Keys.*ENTER*);  
  
 click(submitBtn);  
 }  
  
 public String getModalTitle() {  
 return getText(modalTitle);  
 }  
  
 public void waitForFormVisible() {  
 wait.until(ExpectedConditions.*visibilityOfElementLocated*(firstNameInput));  
 }  
  
 public void closeAdIfPresent() {  
 By closeAdBtn = By.*id*("close-fixedban");  
 try {  
 if (isElementVisible(closeAdBtn)) {  
 click(closeAdBtn);  
 }  
 } catch (Exception ignored) {}  
 }  
  
 public void closeFooterIfPresent() {  
 By footer = By.*id*("fixedban");  
 try {  
 if (isElementVisible(footer)) {  
 ((org.openqa.selenium.JavascriptExecutor) driver)  
 .executeScript("arguments[0].remove();", driver.findElement(footer));  
 }  
 } catch (Exception ignored) {}  
 }  
  
 public void waitUntilFormIsFullyReady() {  
 wait.until(ExpectedConditions.*visibilityOfElementLocated*(firstNameInput));  
 }  
  
 public void scrollToForm() {  
 ((org.openqa.selenium.JavascriptExecutor) driver)  
 .executeScript("arguments[0].scrollIntoView(true);", driver.findElement(firstNameInput));  
 }  
}

**Tạo PracticeFormTest.java**

- Dùng JUnit + Selenium để kiểm thử form với từng dòng dữ liệu từ CSV.

package tests;  
  
import org.junit.jupiter.api.Assertions;  
import org.junit.jupiter.api.Test;  
import pages.PracticeFormPage;  
  
class PracticeFormTest extends BaseTest {  
  
 @Test  
 void testFormSubmissionDisplaysSuccessModal() {  
 PracticeFormPage formPage = new PracticeFormPage(*driver*);  
  
 formPage.openFormPage();  
 formPage.waitUntilFormIsFullyReady();  
 formPage.closeAdIfPresent();  
 formPage.closeFooterIfPresent();  
 formPage.waitForFormVisible();  
  
 formPage.fillForm();  
  
 String expectedTitle = "Thanks for submitting the form";  
 String actualTitle = formPage.getModalTitle();  
  
 Assertions.*assertEquals*(expectedTitle, actualTitle, "Modal title doesn't match");  
 }  
}

**B3:**

**- JavaScript và Dynamic Content:**

Thêm waitUntilFormIsFullyReady()

Xử lý quảng cáo với closeAdIfPresent()

Xử lý footer với closeFooterIfPresent()

**- File Upload:**

Đặt test-pic.png trong src/test/resources

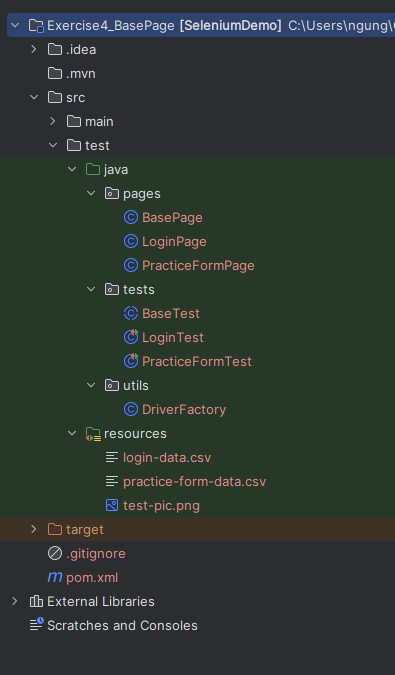
Sử dụng đường dẫn tuyệt đối cho file upload

Đóng quảng cáo và footer:

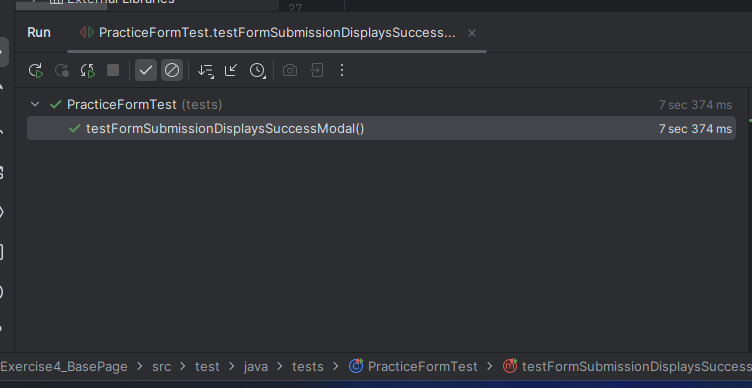
Thêm phương thức closeAdIfPresent()

Thêm phương thức closeFooterIfPresent()

- **Cấu trúc project:**



Kết quả test:



**5.1 Chuyển qua dùng thông tin trong file practice-form-data.csv**

Lưu trữ dữ liệu test ở dạng file CSV để test nhiều dòng dữ liệu

firstName,lastName,email,gender,mobile,birthDate,subject,hobbies,address,state,city,expectedMessage  
Nguyen,Van A,vana@example.com,Male,0912345678,22 Jul 1995,Maths,Sports,123 Nguyen Van Cu Hanoi,NCR,Delhi,Thanks for submitting the form  
John,Doe,invalid-email,Male,1234567890,15 Aug 1990,English,Sports,456 New Street,NCR,Delhi,Form submission failed  
Mary,Jane,mary@test.com,Female,invalid-mobile,10 Jun 1985,Chemistry,Reading,789 Old Road,NCR,Delhi,Form submission failed

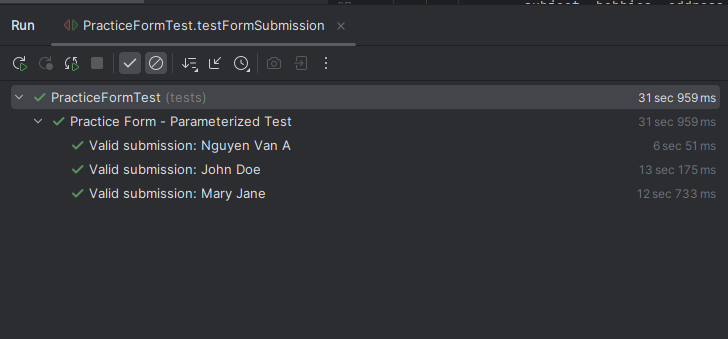
**PracticeFormPage.java**

package pages;  
  
import org.openqa.selenium.By;  
import org.openqa.selenium.Keys;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.support.ui.ExpectedConditions;  
import org.openqa.selenium.support.ui.WebDriverWait;  
  
import java.time.Duration;  
  
public class PracticeFormPage extends BasePage {  
  
 public PracticeFormPage(WebDriver driver) {  
 super(driver);  
 wait = new WebDriverWait(driver, Duration.*ofSeconds*(10));  
 }  
  
 // Locators  
 private final By firstNameInput = By.*id*("firstName");  
 private final By lastNameInput = By.*id*("lastName");  
 private final By emailInput = By.*id*("userEmail");  
 private final By genderMale = By.*xpath*("//label[text()='Male']");  
 private final By mobileInput = By.*id*("userNumber");  
 private final By dobInput = By.*id*("dateOfBirthInput");  
 private final By subjectInput = By.*id*("subjectsInput");  
 private final By hobbySports = By.*xpath*("//label[text()='Sports']");  
 private final By uploadPicture = By.*id*("uploadPicture");  
 private final By addressInput = By.*id*("currentAddress");  
 private final By stateInput = By.*id*("react-select-3-input");  
 private final By cityInput = By.*id*("react-select-4-input");  
 private final By submitBtn = By.*id*("submit");  
 private final By modalTitle = By.*id*("example-modal-sizes-title-lg");  
  
 public void openFormPage() {  
 navigateTo("https://demoqa.com/automation-practice-form");  
 }  
  
 public void fillForm(String firstName, String lastName, String email,  
 String gender, String mobile, String birthDate,  
 String subject, String hobbies, String address,  
 String state, String city) {  
 type(firstNameInput, firstName);  
 type(lastNameInput, lastName);  
 type(emailInput, email);  
  
 // Select gender  
 By genderLocator = By.*xpath*("//label[text()='" + gender + "']");  
 click(genderLocator);  
  
 type(mobileInput, mobile);  
  
 // Date of birth  
 click(dobInput);  
 driver.findElement(dobInput).sendKeys(Keys.*CONTROL* + "a");  
 driver.findElement(dobInput).sendKeys(birthDate);  
 driver.findElement(dobInput).sendKeys(Keys.*ENTER*);  
  
 // Subject  
 type(subjectInput, subject);  
 driver.findElement(subjectInput).sendKeys(Keys.*ENTER*);  
  
 // Select hobbies  
 By hobbiesLocator = By.*xpath*("//label[text()='" + hobbies + "']");  
 click(hobbiesLocator);  
  
 // Upload picture - using relative path from project root  
 String projectPath = System.*getProperty*("user.dir");  
 String imagePath = projectPath + "/src/test/resources/test-pic.png";  
 driver.findElement(uploadPicture).sendKeys(imagePath);  
  
 type(addressInput, address);  
  
 // Select state and city  
 type(stateInput, state);  
 driver.findElement(stateInput).sendKeys(Keys.*ENTER*);  
 type(cityInput, city);  
 driver.findElement(cityInput).sendKeys(Keys.*ENTER*);  
  
 click(submitBtn);  
 }  
  
 public String getModalTitle() {  
 try {  
 return getText(modalTitle);  
 } catch (Exception e) {  
 return "Form submission failed";  
 }  
 }  
  
 public void waitForFormVisible() {  
 wait.until(ExpectedConditions.*visibilityOfElementLocated*(firstNameInput));  
 }  
  
 public void closeAdIfPresent() {  
 By closeAdBtn = By.*id*("close-fixedban");  
 try {  
 if (isElementVisible(closeAdBtn)) {  
 click(closeAdBtn);  
 }  
 } catch (Exception ignored) {}  
 }  
  
 public void closeFooterIfPresent() {  
 By footer = By.*id*("fixedban");  
 try {  
 if (isElementVisible(footer)) {  
 ((org.openqa.selenium.JavascriptExecutor) driver)  
 .executeScript("arguments[0].remove();", driver.findElement(footer));  
 }  
 } catch (Exception ignored) {}  
 }  
  
 public void waitUntilFormIsFullyReady() {  
 wait.until(ExpectedConditions.*visibilityOfElementLocated*(firstNameInput));  
 }  
  
 public void scrollToForm() {  
 ((org.openqa.selenium.JavascriptExecutor) driver)  
 .executeScript("arguments[0].scrollIntoView(true);", driver.findElement(firstNameInput));  
 }  
}

**PracticeFormTest.java**

package tests;  
  
import org.junit.jupiter.api.Assertions;  
import org.junit.jupiter.api.DisplayName;  
import org.junit.jupiter.params.ParameterizedTest;  
import org.junit.jupiter.params.provider.CsvFileSource;  
import pages.PracticeFormPage;  
  
class PracticeFormTest extends BaseTest {  
  
 @ParameterizedTest(name = "Valid submission: {0} {1}")  
 @CsvFileSource(resources = "/practice-form-data.csv", numLinesToSkip = 1)  
 @DisplayName("Practice Form - Parameterized Test")  
 void testFormSubmission(String firstName, String lastName, String email,  
 String gender, String mobile, String birthDate,  
 String subject, String hobbies, String address,  
 String state, String city, String expectedMessage) {  
  
 PracticeFormPage formPage = new PracticeFormPage(*driver*);  
  
 formPage.openFormPage();  
 formPage.waitUntilFormIsFullyReady();  
 formPage.closeAdIfPresent();  
 formPage.closeFooterIfPresent();  
  
 formPage.fillForm(firstName, lastName, email, gender, mobile, birthDate,  
 subject, hobbies, address, state, city);  
  
 String actualMessage = formPage.getModalTitle();  
  
 // Kiểm tra thông điệp xuất hiện trong modal title  
 Assertions.*assertEquals*(expectedMessage, actualMessage,  
 "❌ Modal title doesn't match expected result");  
 }  
}

**Kết quả test:**



**Bài tập 6:**

**Làm 1 project khác áp dụng cho dự án của nhóm.**

Yêu cầu:

* Sử dụng mô hình POM, áp dụng kiểm thử cho ứng dụng của project nhóm với ít nhất 2 chức năng bất kỳ trong danh sách sau: login, đăng ký tài khoản, trang quản lý thông tin cá nhân (sử dụng upload ảnh), thay đổi password, trang CRUD của 1 chức năng bất kỳ.
* Bổ sung tính năng nâng cao (nếu có): CI/CD, Report,…
* Viết System test cho các Test case của 2 chức năng trên theo mẫu excel.

**B1:** Tạo **pom.xml**

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.swp391</groupId>  
 <artifactId>SWP391\_New</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <selenium.version>4.21.0</selenium.version>  
 <webdrivermanager.version>5.8.0</webdrivermanager.version>  
 <testng.version>7.10.2</testng.version>  
 <slf4j.version>2.0.13</slf4j.version>  
 </properties>  
  
 <dependencies>  
 <!-- Selenium WebDriver -->  
 <dependency>  
 <groupId>org.seleniumhq.selenium</groupId>  
 <artifactId>selenium-java</artifactId>  
 <version>${selenium.version}</version>  
 </dependency>  
  
 <!-- WebDriverManager -->  
 <dependency>  
 <groupId>io.github.bonigarcia</groupId>  
 <artifactId>webdrivermanager</artifactId>  
 <version>${webdrivermanager.version}</version>  
 </dependency>  
  
 <!-- TestNG -->  
 <dependency>  
 <groupId>org.testng</groupId>  
 <artifactId>testng</artifactId>  
 <version>${testng.version}</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- SLF4J Logger (fix warning) -->  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-simple</artifactId>  
 <version>${slf4j.version}</version>  
 </dependency>  
 </dependencies>  
</project>

**B2:** Tạo **src/test/java/com/swp391/test/LoginAndChangePasswordTest.java**

package com.swp391.test;  
  
import com.swp391.pages.LoginPage;  
import com.swp391.pages.ChangePasswordPage;  
import io.github.bonigarcia.wdm.WebDriverManager;  
import org.openqa.selenium.chrome.\*;  
import org.openqa.selenium.WebDriver;  
  
public class LoginAndChangePasswordTest {  
  
 public static void main(String[] args) {  
 WebDriver driver = null;  
  
 try {  
 WebDriverManager.*chromedriver*().setup();  
 ChromeOptions options = new ChromeOptions();  
 options.addArguments("--remote-allow-origins=\*");  
 driver = new ChromeDriver(options);  
 driver.manage().window().maximize();  
  
 // 1. Test đăng nhập sai mật khẩu  
 *testLoginWrongPassword*(driver);  
  
 // 2. Đăng nhập đúng để test đổi mật khẩu  
 LoginPage loginPage = new LoginPage(driver);  
 loginPage.open();  
 loginPage.login("vansaolol00@gmail.com", "Han126...7"); // Mật khẩu hiện tại  
  
 if (loginPage.isLoginSuccess()) {  
 System.*out*.println("✅ Đăng nhập thành công!");  
  
 // 3. Test xác nhận mật khẩu không khớp  
 *testChangePasswordConfirmMismatch*(driver);  
  
 // 4. Test mật khẩu hiện tại sai  
 *testChangePasswordWrongCurrent*(driver);  
  
 // 5. Test mật khẩu mới không hợp lệ  
 *testChangePasswordNewTooShort*(driver);  
  
 // 6. Đổi mật khẩu thành công  
 driver.get("http://localhost:8080/JobSeekerProfile");  
 ChangePasswordPage changePage = new ChangePasswordPage(driver);  
 changePage.openProfilePage();  
 changePage.openChangePasswordForm();  
 changePage.changePassword("Han126...7", "Han126...8", "Han126...8");  
  
 if (changePage.isSuccessMessageDisplayed("Password changed successfully!")) {  
 System.*out*.println("✅ Đổi mật khẩu thành công!");  
 } else {  
 System.*out*.println("❌ Đổi mật khẩu thất bại.");  
 }  
  
 } else {  
 System.*out*.println("❌ Đăng nhập thất bại.");  
 }  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 if (driver != null) {  
 driver.quit();  
 System.*out*.println("Browser closed");  
 }  
 }  
 }  
  
 public static void testLoginWrongPassword(WebDriver driver) {  
 LoginPage loginPage = new LoginPage(driver);  
 loginPage.open();  
 loginPage.login("vansaolol00@gmail.com", "wrongpassword");  
 if (loginPage.isErrorMessageDisplayed("Incorrect email or password!")) {  
 System.*out*.println("✅ Test đăng nhập sai mật khẩu: Passed");  
 } else {  
 System.*out*.println("❌ Test đăng nhập sai mật khẩu: Failed");  
 }  
 }  
  
 public static void testChangePasswordConfirmMismatch(WebDriver driver) {  
 driver.get("http://localhost:8080/JobSeekerProfile");  
 ChangePasswordPage changePage = new ChangePasswordPage(driver);  
 changePage.openProfilePage();  
 changePage.openChangePasswordForm();  
 changePage.changePassword("Han126...7", "Han126...8", "KhongKhop");  
 if (changePage.isErrorMessageDisplayed("New password and confirm password do not match")) {  
 System.*out*.println("✅ Test xác nhận mật khẩu sai: Passed");  
 } else {  
 System.*out*.println("❌ Test xác nhận mật khẩu sai: Failed");  
 }  
 }  
  
 public static void testChangePasswordWrongCurrent(WebDriver driver) {  
 driver.get("http://localhost:8080/JobSeekerProfile");  
 ChangePasswordPage changePage = new ChangePasswordPage(driver);  
 changePage.openProfilePage();  
 changePage.openChangePasswordForm();  
 changePage.changePassword("wrongcurrent", "Han126...8", "Han126...8");  
 if (changePage.isErrorMessageDisplayed("Current password is incorrect")) {  
 System.*out*.println("✅ Test mật khẩu hiện tại sai: Passed");  
 } else {  
 System.*out*.println("❌ Test mật khẩu hiện tại sai: Failed");  
 }  
 }  
  
 public static void testChangePasswordNewTooShort(WebDriver driver) {  
 driver.get("http://localhost:8080/JobSeekerProfile");  
 ChangePasswordPage changePage = new ChangePasswordPage(driver);  
 changePage.openProfilePage();  
 changePage.openChangePasswordForm();  
 changePage.changePassword("Han126...7", "123", "123");  
 if (changePage.isErrorMessageDisplayed("Password must be at least 8 characters and include uppercase, lowercase, number, and special character")) {  
 System.*out*.println("✅ Test mật khẩu mới quá ngắn: Passed");  
 } else {  
 System.*out*.println("❌ Test mật khẩu mới quá ngắn: Failed");  
 }  
 }  
}

**B3:** Tạo các class trong thư mục pages:

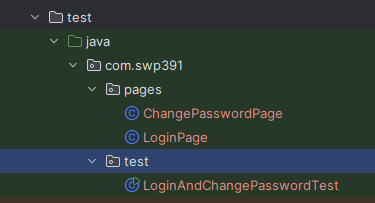
**src/test/java/com/swp391/pages/LoginPage.java**

package com.swp391.pages;  
  
import org.openqa.selenium.\*;  
import org.openqa.selenium.support.ui.\*;  
  
import java.time.Duration;  
  
public class LoginPage {  
 private WebDriver driver;  
 private WebDriverWait wait;  
  
 private By email = By.*name*("email");  
 private By password = By.*name*("password");  
 private By loginBtn = By.*cssSelector*("button[type='submit']");  
  
 public LoginPage(WebDriver driver) {  
 this.driver = driver;  
 wait = new WebDriverWait(driver, Duration.*ofSeconds*(10));  
 }  
  
 public void open() {  
 driver.get("http://localhost:8080/Login.jsp");  
 wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*tagName*("form")));  
 }  
  
 public void login(String user, String pass) {  
 driver.findElement(email).clear();  
 driver.findElement(email).sendKeys(user);  
  
 driver.findElement(password).clear();  
 driver.findElement(password).sendKeys(pass);  
  
 driver.findElement(loginBtn).click();  
 }  
  
 public boolean isLoginSuccess() {  
 return wait.until(ExpectedConditions.*or*(  
 ExpectedConditions.*urlContains*("Dashboard"),  
 ExpectedConditions.*urlContains*("Home"),  
 ExpectedConditions.*urlContains*("JobSeekerProfile"),  
 ExpectedConditions.*not*(ExpectedConditions.*urlContains*("Login"))  
 ));  
 }  
 public boolean isErrorMessageDisplayed(String errorMessage) {  
 try {  
 return wait.until(ExpectedConditions.*textToBePresentInElementLocated*(  
 By.*cssSelector*(".alert-danger, .error-message"), errorMessage));  
 } catch (TimeoutException e) {  
 return false;  
 }  
 }  
}

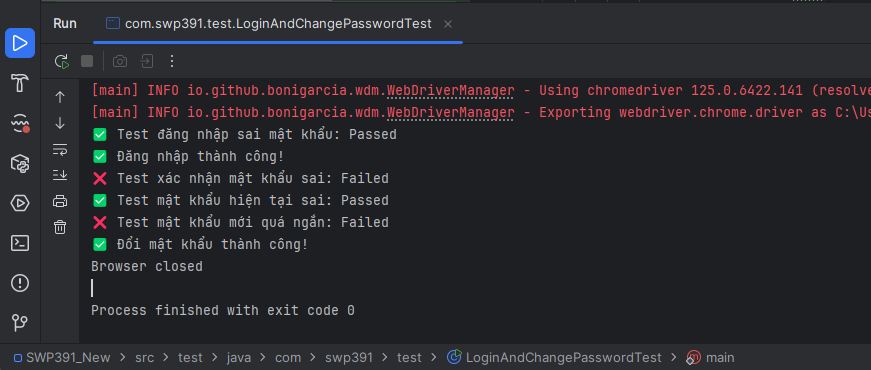
**src/test/java/com/swp391/pages/ChangePasswordPage.java**

package com.swp391.pages;  
  
import org.openqa.selenium.\*;  
import org.openqa.selenium.support.ui.\*;  
  
import java.time.Duration;  
  
public class ChangePasswordPage {  
 private WebDriver driver;  
 private WebDriverWait wait;  
  
 public ChangePasswordPage(WebDriver driver) {  
 this.driver = driver;  
 wait = new WebDriverWait(driver, Duration.*ofSeconds*(10));  
 }  
  
 public void openProfilePage() {  
 WebElement profileLink = wait.until(ExpectedConditions.*elementToBeClickable*(  
 By.*cssSelector*("a.nav-link[href='/JobSeekerProfile']")));  
 profileLink.click();  
 }  
  
 public void openChangePasswordForm() {  
 WebElement changeLink = wait.until(ExpectedConditions.*elementToBeClickable*(  
 By.*cssSelector*("a.btn.btn-outline-primary[href='ChangePassword']")));  
 changeLink.click();  
 wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*tagName*("form")));  
 }  
  
 public void changePassword(String current, String newPass, String confirmPass) {  
 WebElement currentPasswordInput = wait.until(ExpectedConditions.*presenceOfElementLocated*(By.*name*("currentPassword")));  
 currentPasswordInput.clear();  
 currentPasswordInput.sendKeys(current);  
  
 WebElement newPasswordInput = wait.until(ExpectedConditions.*presenceOfElementLocated*(By.*name*("newPassword")));  
 newPasswordInput.clear();  
 newPasswordInput.sendKeys(newPass);  
  
 WebElement confirmPasswordInput = wait.until(ExpectedConditions.*presenceOfElementLocated*(By.*name*("confirmPassword")));  
 confirmPasswordInput.clear();  
 confirmPasswordInput.sendKeys(confirmPass);  
  
 WebElement submitButton = wait.until(ExpectedConditions.*elementToBeClickable*(  
 By.*xpath*("//button[contains(text(),'Change Password')]")));  
 submitButton.click();  
 }  
  
 public boolean isSuccessMessageDisplayed(String message) {  
 try {  
 return wait.until(ExpectedConditions.*textToBePresentInElementLocated*(  
 By.*cssSelector*(".alert-success"), message));  
 } catch (TimeoutException e) {  
 return false;  
 }  
 }  
  
 public boolean isErrorMessageDisplayed(String errorMessage) {  
 try {  
 return wait.until(ExpectedConditions.*textToBePresentInElementLocated*(  
 By.*cssSelector*(".alert-danger, .error-message"), errorMessage));  
 } catch (TimeoutException e) {  
 return false;  
 }  
 }  
}

**-** Cấu trúc project



- Kết quả test



loginPage.open();

loginPage.login("vansaolol00@gmail.com", "wrongpassword");

if (loginPage.isErrorMessageDisplayed("Incorrect email or password!"))

**Input:** Email đúng, password sai.

**Xử lý:** Servlet login sẽ check thông tin trong DB và trả về "Incorrect email or password!" nếu sai.

**Kiểm tra:** Selenium tìm thông báo lỗi có nội dung chính xác đó.

**Kết quả:** Passed → Vì hệ thống hiển thị đúng thông báo và test kiểm tra đúng nội dung đó.

loginPage.open();

loginPage.login("vansaolol00@gmail.com", "wrongpassword");

if (loginPage.isErrorMessageDisplayed("Incorrect email or password!"))

**Input:** Email đúng, password sai.

**Xử lý:** Servlet login sẽ check thông tin trong DB và trả về "Incorrect email or password!" nếu sai.

**Kiểm tra:** Selenium tìm thông báo lỗi có nội dung chính xác đó.

**Kết quả:** Passed → Vì hệ thống hiển thị đúng thông báo và test kiểm tra đúng nội dung đó.

loginPage.login("vansaolol00@gmail.com", "Han126...7");

if (loginPage.isLoginSuccess())

**Input:** Đăng nhập bằng mật khẩu hiện tại "Han126...7".

**Xử lý:** Login thành công thì chuyển hướng đến /JobSeekerProfile.

**Kiểm tra:** isLoginSuccess() kiểm tra URL không còn là Login, tức là đã chuyển trang.

**Kết quả:** Passed → Đăng nhập đúng, điều hướng đúng.

changePage.changePassword("wrongcurrent", "Han126...8", "Han126...8");

if (changePage.isErrorMessageDisplayed("Current password is incorrect"))

**Input:** Nhập sai mật khẩu hiện tại.

**Xử lý:** DAO trả về false → message "Current password is incorrect" được set.

**Kết quả:** Passed → Message khớp, UI hiển thị đúng → test pass.

**Yêu cầu:**

1. **Nộp lên edunext file word chứa:**

* **Exercise 5, Exercise 6**

**Ghi lại từng Bước làm. Chụp màn hình cho đoạn code, kết quả chạy tương ứng. Giải thích về kết quả.**

1. **Up code lên thư mục Automation Testing trên github cá nhân đã đăng ký, chứa 2 folder Exercise 5, Exercise 6**

**. Và file Excel System Test cho 2 chức năng của bài Exercise 6**

**Lưu ý:**

Up đầy đủ tất cả các folder chứa code các bài tập đã làm các slot trước trước báo cáo Presentation ở slot 19 như:

* **Unit Test**
* **Static Testing: Review code, Review Requirements,…**
* **Automation Testing with Selenium: Exercise 5, Exercise 6**