**LAB 6: THỰC HÀNH SELENIUM  
 AUTOMATION VỚI MOUSE & KEYBOARD**

**I/ Yêu cầu thực hành**

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

**II/ Thiết lập project.**

**2.1** Tạo Maven project.

<Tham khảo: REF\_1\_CREATE\_MAVEN\_PROJECT.docx>

**2.2**.Cài đặt TestNG

<Tham khảo: REF\_2\_Setup\_TestNG.docx>

**2.3.** Cài đặt Selenium và Chrome driver.

<Tham khảo: REF\_4\_Setup\_Selenium.docx>

**2.4**. Các hàm thông dụng trong Selenium webdriver.

<Tham khảo: REF\_5\_basic\_func\_webdriver.docx>

**III. Thực hành**

**1.** Yêu cầu viết chương trình tự động hóa quá trình sau.

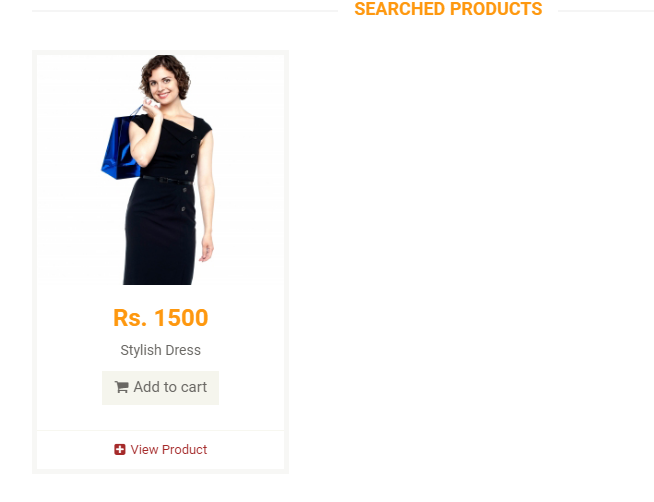
* Gọi trình duyệt Google Chrome.
* Mở URL: [**https://automationexercise.com/products**](https://automationexercise.com/products)Quá trình tải trang không vượt quá 15s.
* Thực hiện tìm kiếm từ khóa “**Stylish Dress**” trong ô tìm kiếm . Quá trình tìm kiếm không vượt quá 5s.
* Kiểm tra kết quả đảm bảo chỉ 1 Item được tìm thấy
* Kiểm tra hình ảnh Item có được hiển thị hay không, đảm bảo item được hiển thị cùng tên tìm kiếm và chứa thông tin giá bán.
* Đảm bảo button “Add to cart” được hiển thị trên Item.
* Thực hiện click vào button “Add to card”  
  Quá trình thêm giỏ hàng không quá 5s.  
  Kiểm tra thêm giỏ hàng thành công hay thất bại.
* Yêu cầu tất cả quá trình nhập văn bản và click chuột đều thực hiện thông qua lớp Actions của selenium.

*Gợi ý:*

1. Tạo Maven project có tên **LAB\_6\_SELENIUM**, cài đặt thư viện testNG và selenium
2. Tạo một class trong package: test/java có tên **MyStoreAutomation.java** Biến URL(kiểu String) : lưu url của trang web.  
    Biến keyword(Kiểu String) : Lưu từ khóa cần tìm kiếm.
3. Tạo phương thức : public void Init() và đánh chú thích @BeforeClass  
   Trong phương thức này thực hiện các task sau:  
    - Khởi tạo dư liệu: gán giá trị cho URL và keyword theo đề bài yêu cầu.  
    - Khởi tạo ChromeDriver .  
    - Mở cửa sổ trình duyệt full kích thước.  
    // maximise the window  
    driver.manage().window().maximize();

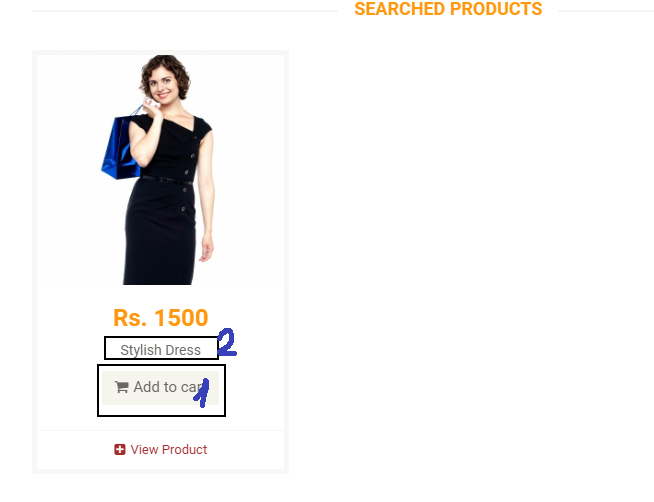
4) Tiến hành truy cập trang để xác định các vị trí web element dựa  
trên **id, class, name,.. xpath**  
  
**#1  
  
  
   
**

**1. inputSearch = ?  
 2. btnSearch = ?  
  
#2**

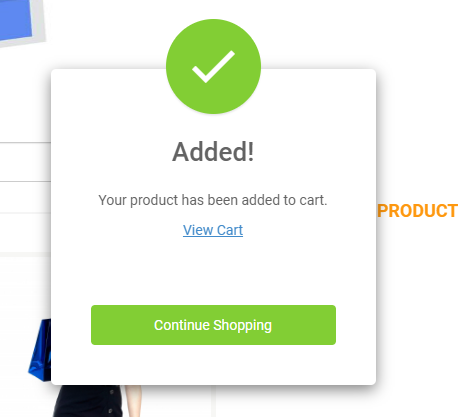


**1. imageLink = ?**

**#3**



**1. btnAddToCart = ?  
 2. productName = ?  
   
#4**

**  
 1.addResponseMsg = ?**

1. Tiến hành viết code cho testcase :

* Viết một method có tên là : **TC\_MyOderSearchValid()** và đánh **@Test** cho method để đánh dấu đây là một testcase trong testNG.
* Trong method này, tiến hành viết code cho từng màn, từng bước.  
    
   1. Sử dụng method driver.get() để truy cập trang :   
   [**https://automationexercise.com/products**](https://automationexercise.com/products)

Sử dụng **WebDriverWait** để cài đặt thời gian tải trang là không vượt quá  
 giới hạn 15**s**.  
   
 Kiểm tra qua trình tải trang có thành công hay không?  
 Mong muốn của quá trình này sẽ là chuyển đến #**1.** Do đó, chúng ta chỉ cần kiểm tra xem ô văn bản để tìm kiếm đã hiển thị hay chưa.  
  
 #Lấy ra ô văn bản nhập tìm kiếm.

| WebDriverWait wait = …  WebElement inputSearch = wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*<id|name|Xpath> (...*)));    //vd: visibilityOfElementLocated(By.id("abcxyz")); |
| --- |

#Sử dụng Assert trong testNG để xác minh ô văn bản đã hiển thị hay chưa. Nếu chưa hiển thị thì sẽ bắn ra 1 message: “ Page load failed”

| *assertEquals*(**true**, inputSearch.isDisplayed(), "Page load failed"); |
| --- |

# Sử dụng Action để nhập keyword vào ô tìm kiếm và sau đó ENTER hoặc click chuột vào biểu tượng tìm kiếm

| Actions action = **new** Actions(driver);  //nhập keyword vào ô văn bản tìm kiếm  action.sendKeys(inputSearch, keyword).build().perform();  // ấn Enter  action.sendKeys(Keys.ENTER).build().perform();    HOẶC WebElement btnSearch = ... // Click chuột vào biểu tượng tìm kiếm action.click(btnSearch).build().perform(); |
| --- |

2. Kiểm tra kết quả tìm kiếm (Màn 2)  
  *Chú ý*: Sau khi thực hiện ENTER hoặc Click chuột vào “Tìm kiếm” ở màn 1,  
 Chúng ta cần tiếp tục sử dụng **WebDriverWait** cài đặt 5s là thời gian tối đa cho quá trình tìm kiếm.  
 Kiểm tra Quá trình tìm kiếm đã thành công hay chưa dựa vào sự xuất hiện của kết quả tìm kiếm: **resultMessage** trên website .  
 assertEquals(true, resultMessage.isDisplayed(), "Search failed");  
 Sử dụng Assert và webdriver để kiểm chứng thông tin sau:   
 - Kiểm tra **imageLink** sản phẩm đã hiển thị hay chưa.  
 3. Thực hiện thêm sản phẩm vào giỏ hàng  
 Đảm bảo “Add to cart” được hiển thị.

| *//kiểm tra* Assert.*assertEquals*(**true**, **btnAddToCart**.isDisplayed()); |
| --- |

- Kiểm tra **productName** của sản phẩm đã hiển thị chưa, nếu đã hiển thị thì kiểm tra tên đó có khớp với từ khóa tìm kiếm (keyword).  
 - Sử dụng Action để click chuột vào button “**Add to cart**”  
 - Sử dụng **WebDriverWait** để đảm bảo quá trình không quá 5s.  
 Assert đảm bảo **addResponseMsg** đã được hiển thị trên website.  
 Đảm bảo việc thêm mới thành công với thông báo sau:  
 “**Your product has been added to cart.**”

| ….. WebElement addResponseMsg = wait.until(ExpectedConditions. …..  String expect = **"Your product has been added to cart."**; String actual = addResponseMsg.getText();  //so sánh Assert.*assertEquals*(expect, actual); |
| --- |

1. Viết một phương thức : public void terminate() và đánh @AfterClass

Thực hiện đóng trình duyệt.

6) Chạy testcase.