Selenium Web Driver

Selenium is a library that allows to perform task in web applications.

Kickoff

- Dowload chrome driver
- Create a new maven project and add Selenium Java dependacy.
- Import the webdriver in the project (e.g. past it in the root)
- Define a system property to specify where the chromedriver is.

```
    System.setProperty("webdriver.chrome.driver", "chromedriver"); ← Use the Path and the extension.
```

- Define an implicitly wait (it is a good practice)

```
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
```

- Open a website

```
    driver.get("santiagovasco.com");
```

- Close the process

```
    driver.close();
```

```
public static void main(){
    WebDriver driver = new ChromeDriver();
    System.setProperty("webdriver.chrome.driver", "chromedriver");
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
    driver.get("http://santiago vasco.com");
    TimeUnit.SECONDS.sleep(10);
    driver.close();
}
```

Locators

DOM → Document object Model. The locators will identify web elements in the DOM to perform accions whit them.

Types of locators:

• ID	`<div id="content" class="mw-body" role="main"> ... </div>`
• Xpath	it is like a sql Query `//*[@id="Content"]/div[2]` get the second div element under the div whit ID= content.
• CssLocator	`#content > div.mw-indicators.mw-body-content` it largest than Xpath but has some advantages over xpath.
• ClassName	`<div class="mw-indicators mw-body-content">`
• TagName	`<div class="mw-indicators m...`
• linkText	

Comparing Perfomance of locators

ID	Faster locator	Some apps auto generates the IDs, like Angular apps...
CSS		Not all browsers supports the CSS.
Name		
xpath	Slowest locator	try to avoid xpath always.

Accions

.Getting text from a WebPage

```
WebElement webPageTitle = driver.findElement(By.cssSelector("#SelectorBlaBla"));
String webPageTitleText = webPageTitle.getText();
Assert.equals("Expected Title", webPageTitleText");
```

.Click on a button or a link

```
WebElement webPageLink = driver.findElement(By.cssSelector("#SelectorBlaBla"));
webPageLink.click(); ← void method.
// Then we can continue getting web elements from the DOM.
```

.Write text

```
WebElement inputField = driver.findElement(By.cssSelector("#SelectorBlaBla"));
inputField.sendKeys("This is the text");
```

.Working whit Tables

```
WebElement item = driver.findElement(By.xpath("/html/body/table/tbody[1]/tr[2]/td[1]"));
Sout(item.getText()); // prints 1

WebElement item = driver.findElement(By.xpath("/html/body/table/tbody[1]/tr[1]/td[2]"));
Sout(item.getText()); // prints Second Col

List<WebElement> items = driver.findElements(By.xpath("/html/body/table/tbody[1]/tr"));
items.forEach(webElement -> Sout(webElement.getText())); // prints First Col Second Col Third Col
                                                                   1     2     3
```

Working whit this table

First Col	Second Col	Third Col
1	2	3

.Select

```
webElement selectable = driver.findElement(By.id("some-id"));
Select select = new Select(selectable);
select.selectByIndex(0);
```

.Accions

Used to perform accions like drag or drop elements, right click, press keys, etc

```
WebElement draggable = driver.findElement(By.id("draggable-id"));
WebElement droppable = driver.findElement(By.id("droppable-id"));
Actions dragElement = new Actions(driver);
dragElement.dragAndDrop(draggable, droppable).build().perform();
```

drag and drop the element.

```
Actions contextClick = new Actions(driver);
contextClick.contextClick().build().perform();
```

This is the right click.

Types of waits

Implicit wait

time the web driver will wait for an element before throwing an Element not found exception.

```
driver.manage().timeouts().implicitWait(10,TimeUnit.SECONDS);
```

Is a good practice set this at the begining because there could be issues loading elements or the web app could take some time to load comptettly. To prevent some useless WebElement not found exceptions.

Explicit Wait

Can be used to wait an amount of time for a specific condition of a web element.

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

Fluent Wait

Is used to wait an amount of timen and ignore the exception.

Declared →

```
Wait fluentWait = new FluentWait<WebDriver>(driver)
        .whitTimeout(Duration.ofSeconds(15))        ← Timeout ta throw Exception
        .pollingEvery(Duration.ofSeconds(5))        ← Check every 5 seconds ignoring the exeption
        .ignoring(NoSuchElementException.class);       until reach the 15 seconds of timeout.
```

Use it →

```
fluentWait.until( new Function<WebDriver, WebElement>() {
        public WebElement apply(WebDriver driver) {
                return driver.findElement(By.id(" id"));
        }
});
```

Javascript Executor

Can be useful for example if a particular web element is not clickable. whit the Javascritp Executor we can invoke JS functions.

```
//Some elements could not be clickable, like a span or a div acting as a button.
WebElement webButton = driver.findElement(By.id("some-id"));
((JavascriptExecutor) driver).executeScript("arguments[0].click();", webButton);
```

```
//Setting up timeouts.
                                                                                                1s.
((JavascriptExecutor) driver).executAsyncScript("window.setTimeout(arguments[arguments.length -1],1000);");
```

```
// Changing the webpage
((JavascriptExecutor) driver).executeScript("window.location ='http://santiagovarxo.com`");
```

```
// Scroll by pixels
((JavascriptExecutor) driver).executeScript("window.scrollBy(0,1000)");
```

Alerts

The idea is to show up the Alert (Pop-up) when something goes wrong like fill a mandatory field.
When an alert is present we cannot get objects from the webpage.
A good pradice is instantiate a WebDriverWait to be sure that we will wait until the alert is present.

```
WebElement alertBotton = driver.findElement(By.cssSelector("blablabla");
alertBotton.click();
WebDriverWait wait = new WebDriverWait(driver, 15); //wait a max of 15s for the alert.
wait.until(ExpectedCondition.alertIsPresent());
Alert alert = driver.switchTo().alert();
alert.accept();
```

Working whit iFrames

Many of the moder webapp uses iframes thos is a DOM inside other DOM. The proble is if the web driver load faster than the iframe, the web driver wont be able to see it.

```
                 Good Practice: only switch ones per scritp.
    // we can switch between frame using index, id or webElement. Index star whit 0 which is the parent.
    var titleText = driver.switchTo().frame(1).findElement(By.cssSelector("blabla")).getText();
```