Selenium

Selenium automates browsers. What you do with that power is up to you

Setting up Selenium

- Include Selenium Java as a dependency in your pom.xml
- Download the appropriate Chromedriver
- Provide the location of the chromedriver
  - System.setProperty("webdriver.chrome.driver", "C:/webdrivers/chromedriver.exe");
- Instantiate the WebDriver
  - WebDriver driver = new ChromeDriver();
- Navigate to the website
  - driver.get("http://google.com");

Selenium Locators

- Locators are how you actually locate elements on the page
  - Located inside of the “By” class that belongs to the Selenium WebDriver API
  - The “easy” locators (takes only a little bit of time to learn)
    - Locate by name attribute
    - Locate by id attribute
    - Locate by class attribute
    - Locate by link text (a tags)
    - Locate by partial link text (a tags)
    - Locate by tag name
  - The “hard” locators (takes a bit more time to learn)
    - CSS Locator: CSS selectors
    - XPath Locator: XPath syntax

CSS Locator

- Common CSS selectors:
  - Tag
    - p { color: ‘red’ }
    - Styling all p tags red
  - Class selector
    - .some-class { color: ‘red’ }
  - Id selector
    - #some-id { color: ‘red’ }
  - Descendant selector (combinations of selectors here)
    - #some-id div p.some-class { color: ‘red’ }
    - Style the p tags with a class of some-class that are a child (or grand child, great grand child, etc.) of a div tag, which is then a child (or grand child, etc.) of an element with an id of some-id
  - Attribute selector
    - a[href=‘http://google.com’] { color: ‘red’ }
    - Select an a tag with an attribute href with the value of ‘http://google.com’
- You can utilize the CSS selector conventions to select elements within Selenium using a CSS locator strategy

XPath

- This is another way of selecting elements
- It is more flexible than CSS selectors, which is one reason that we might be utilizing Xpath when working with Selenium
- We have two types of paths:
  - Absolute Xpath
    - /html/body/div[1]/p[1]
    - We're going from the parent-most element to a specific child element (being very specific)
    - Absolute Xpath is unstable, however
      - Developers will often change the HTML, and if you utilize the entire structure from top to bottom, your locator might break
  - Relative Xpath
    - //p[1]
    - You don't need to traverse from the html element all the way down
    - You just use // to skip over to the relevant element
    - "Select the very first p tag within the HTML document"

Selenium waits

- Two categories:
  - Explicit: Explicitly state which element to wait for
  - Implicit: Configured once for a WebDriver object, whereby any elements that are not immediately available will just be automatically waited on
- In general, explicit waits are recommended over implicit waits due to them being much more readable and not being a “black-box” like implicit waits
  - Anybody reading the automation code will understand exactly which elements need time to appear (explicit waits)
  - With implicit waits, we have no idea which elements actually need time to appear or just appear instantly or are already there

E2E testing

- We can utilize Selenium to automate actions in the browser for performing E2E tests
- End to end testing is a “User interface” (UI) test. It is not interacting directly with any code in our application
- Hence, E2E testing is known as a “black-box” test, because it’s not actually testing the code directly in the application
- Unit tests and integration tests, on the other hand, are considered “white-box” tests, because they do indeed interact with individual modules / integrated modules of code.

Blackbox v. Whitebox testing

- Blackbox
  - E2E tests
  - User Acceptance Tests (Alpha and beta tests)
  - Do you need to know what the code is to perform these tests?
    No -> therefore it is blackbox tests
- Whitebox
  - Unit tests
  - Integration tests
  - Do you need to know what the code is to write these tests?
    Yes -> therefore they are whitebox tests