CLASSNOTES: DAY 6
Today's schedule:
- Reviews:
- Practices:
- Alerts
- Iframes
- Windows

- XPATH :
- How do you handle dynamic web elements?
- How do you handle web elements that has dynamic attribute value?

#1- We use xpath methods such as contains, starts-with, and ends-with.
    - We locate the part of the attribute that is not dynamic, and use that part within the xpath locators to locate the web element.

#2- We can locate the static (stable) parent or child web elements and move from there.

- How do we move from parent to direct child using xpath?
    - We use "/" single slash to go to direct child.

- How do we move from parent to direct child using xpath?
    - We use "//" double slash to go to any child.

- How do we move from child to parent using xpath?
    - We use "/.."
    - This will take our locator to the parent of currently located web element.

- What is the syntax with the indexing in xpath?
- How do we use indexes with xpath?
    - //tagName[@attribute='value'][3]
    - If we use indexes without paranthesis, this way will only work in direct siblings.

    - (//tagName[@attribute='value'])[3]
    - If we use indexes with paranthesis, this way it will work and index through the whole html page.

- TestNG:
- What is TestNG?
- TestNG is a UNIT TESTING FRAMEWORK.

- Is this tool created for testers?

- No. It is created for developers.

- Is this the only UNIT TESTING FRAMEWORK?
- No.
- JUnit
- NUnit

- How do we create structures or change behaviors of methods using TestNG?
- ANNOTATIONS

- Using annotations we can change the behaviors of the methods.

- Common annotations we will be using?
  - @Test
  - BeforeMethod, AfterMethod
  - BeforeClass, AfterClass
  - BeforeTest, AfterTest
  - BeforeSuite, AfterSuite

- @Test :
  - This annotation will convert a regular Java method into a runnable TestNG test.
  - Each test run by default in alphabetical order.
  - But we can change the running order using priority
  - Each test is independent from each other UNLESS we create dependency.

```
@Test (priority = 1)
public void test1(){
    //code

    //assertion

    //code (this code will not be ran if assertion fails)

    //assertion
}

@Test (priority = 2, dependency="test2")
public void test2(){
}
```

- BeforeMethod:
  - This will run once before each test.
  - The number of tests we have will determine the number this method will be running.
  - If I have 10 @Test, this will run 10 times before each one of them.

- AfterMethod:
    - Same as BeforeMethod, but it will run after.
    - It will just run once after each @Test.

- BeforeClass:
    - This will run ONLY ONE TIME in each class, before everything else.
    - This does not care the number of @Test we have in the class.
    - Regardless it runs one time.

- AfterClass
    - Same as BeforeClass, but it will run after.
    - It will just run once after everything is done in the class.

- Ex: If we want to open a new browser before each test, and close the browser after each test, where do we put our setup and teardown lines?
    #1- BeforeMethod : WebDriver driver = WebDriverFactor.getDriver("chrome");
    #2- AfterMethod   : driver.close();

- Assertions:
    - What do assertions do?
    - Assertions are used to verify if actual = expected
    - String, int, List<WebElement>, List<String>
    - Assertions determine if a test passes or fails.

- Which assertions we have seen so far?
    - Assert.assertEquals()
        - Accepts 2 arguments.
        - Both arguments have to be same type.
        - It will compare and determine fail or pass.

    - Assert.assertTrue()
        - Accepts a boolean value.
        - If boolean returns true --> test passes
        - If boolean returns false --> test fails

    - Assert.fail()
        - If you call this method in a @Test your test will fail regardless.

- DROPDOWNS:
- How many types of dropdowns we have?
- 2

#1- HTML Dropdowns (non-select)

- If the dropdowns are created using anything other than <select> tag, it is a "non-select dropdown"

#2- Select Dropdowns
- If dropdown is created using <select> tag, it is a select dropdown
- To be able to use the Select class from Selenium, it has to be <select> dropdown.

- How do we handle select dropdowns?
#1- I would create Select class object
#2- I would locate the <select> dropdown and pass it into the constructor.
#3- Now I can use the Select object with the methods that comes from Select class.

- getOptions :
    - to get all <options>
    - Return type: List<WebElement>

- getFirstSelectedOption :
    - returns the currently selected option as a single web element.
    - Return type: WebElement

- selectByIndex
- selectByValue
- selectByVisibleText

Notes coming next class:
- Alerts
- Iframes
- Windows