Stage 1 – Selenium (Web Automation)

Stage 2 – Framework (reusable code)

Pre-requisite – Java

Selenium – <https://www.selenium.dev/>

* Only web automation
* Language independent – Java, Python, C#, Ruby, Javascript, php, perl

Selenium – A suite of tools

1. Selenium IDE
   1. No programming knowledge is required
   2. Record/playback feature
   3. Plugin – chrome, firefox, edge
   4. Only for simple scripting or exploratory testing
2. Selenium RC – Depreciated
   1. Programming knowledge is must
   2. Architecture

Source code (Java+Selenium RC) 🡪 RC Server (Turn ON/OFF) 🡪 Browser

1. Selenium WebDriver
   1. Programming knowledge is must
   2. Architecture

Source code (Java+Selenium WebDriver) 🡪 Browser

1. Selenium Grid
   1. If you want to scale by distributing and running tests on several machines and manage multiple environments from a central point

UpperCamelCase – MyFirstProject

lowerCamelCase – myFirstProject

Selenium WebDriver:-

1. Create a java project
2. Configure selenium jar
3. get, gettitle, url, page source
4. Click, type, select
5. Inspect – tagname, attributes, text or not
6. Basic locators
   1. id
   2. name
   3. classname
   4. tagname
   5. link text
   6. partial link text

When there are duplicate, findElement method picks the first one.

1. Advance locators
   1. XPath
   2. CSS
2. Page load – wait for page load
3. findElement method – checks for presence of element in 0.5s
4. Sychronization
   1. Unconditional wait (from java lib)
      1. Thread.sleep(5000) 🡪 not recommended
   2. Conditional wait (from selenium lib)
      1. Implicit wait
         1. Default implicit wait – 0s
         2. Applicable for all findElement and findElements method
         3. Example: Implicit wait – 30s
            1. If element is not present, it will check for 30s and then throw error
            2. If element is present, it will do the operation immediately
            3. Polling time – 0.5s (how freq it checks for element)
      2. Explicit wait
      3. Fluent wait
5. Dropdown
   1. With select tag – Select
      1. selectByVisibleText(String text)
      2. selectByValue(String value)
      3. selectByIndex(int index) – starts at 0
   2. Without select tag
      1. Click()
6. Frame, Mutliple tabs/windows, alert – switchTo()
7. List vs Set
   1. List can contains duplicates
   2. Set cannot contain duplicate
8. Mutliple tabs
   1. driver.getWindowHandles() 🡪 gives all session id
9. Close vs quit

Close – close the current session/current tab

Quit – close the current browser/all session and also it will kill the process (driver) to it.

1. Alert – javascript alert
2. Frame – embedding html into another html
   1. Even though locator is correct, we get NoSuchElementException
   2. Check for tagname – frame or iframe
   3. Switch to frame
      1. Using webelement
3. To inspect – ctrl+shift+c
4. Actions – mouse/keyboard
   1. May not give proper error
   2. May not work in headless mode
   3. Do not distrub during runtime

Keyboard

Keydown() & keyUp() -> only for modifier like ctrl, alt, shift

Remaining keys -> sendKeys()

1. Upload
2. Javascript
   1. Click on hidden elements
   2. Type on readonly text box
   3. Scroll page
   4. Scroll to element

Javascript – click & type

Javascript & webelement – click & type

Hybrid Framework

* + - 1. TestNG – Unit Test Framework
      2. Data Driven Framework – to separate test method from test data. (from excel, json, csv)
      3. Page Object Model – Design pattern (for reusing the webelements)

Maven – Build Management Tools

* pom.xml 🡪 Project Object Model –
  + it will helps us to configure jars required for the project and will take care even depedent jars.
  + Easily controlled using command line

Install TestNG plugin to eclipse

Packages:-

com.citi.test - > test class and test methods

com.citi.base -> browser config

com.citi.utilities -> excel

com.citi.page -> page object class and method

Keyword driven framework for calling the reusable methods.

Steps to create a framework

1. Create Maven Project
   1. Provide groupid (com.citi) and artifact id (project name)
2. If any issues, try to force update the maven project.
3. Configure the dependency (jars) in pom.xml
   1. Selenium jar
   2. TestNG
4. Create a test method using testng (@Test)
5. TestNG helps to run the test method and also
   1. generate two report.
   2. Collect failed testcase
6. @Test method runs in ASCII key code order
7. Annotations
   1. @BeforeMethod – runs before each @Test method
   2. @AfterMethod – runs after each @Test method even though @Test methods fails
8. Every @Test method should have minimum one assertion.
9. Methods – reuse the logic then you can create a methods
10. Inheritance – reuse the variable and methods then you can use inheritance
11. Data Driven Activity – DataProvider
    1. Create a @Test method with arguments/parameters
    2. Create a method that return two dimensional array and also need to provide @DataProvider annotations
    3. Connect the @Test method with @DataProvider
12. Connect DataProvider with Excel
    1. Add poi dependency
       1. poi -
       2. Poi-ooxml –
13. Page object model
    1. For each page, class should be created
    2. Method will be used for doing operation
    3. Collecting the webelement (object repo) at class level