Stage 1 – Selenium (Web Automation)

Stage 2 – Framework (reusable code)

Feb 2023 - <https://github.com/balaji-githubstore/citifeb2023.git>

Mar 2023 - <https://github.com/balaji-githubstore/health_record_bdd_citi_mar2023.git>

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

If element is not present, it will check for 30s and then throw error

If element is present, it will do the operation immediately

Polling time – 0.5s (how freq it checks for element)

* + 1. Explicit wait
    2. Fluent wait

1. 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()
2. Frame, Mutliple tabs/windows, alert – switchTo()
3. List vs Set
   1. List can contains duplicates
   2. Set cannot contain duplicate
4. Mutliple tabs
   1. driver.getWindowHandles() 🡪 gives all session id
5. 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

March 13, 2023

BDD – Behaviour Driven Development

* Actual BDD – Focus – Understanding requirements
* Modified BDD – ATDD – Acceptance Test Driven Development – Focus:writing acceptance testing

Structured scenarios –

* Create a feature file (.feature)

Feature: Login

Scenario: Valid Login

Given – Pre-requisite

When – Actions/ Operations

Then – Verifications

Scenario: Valid Login

Given I have browser with fb page

When I provide the credential

Then I should get access to news feed

Where to implement BDD?

1. Everyone knows about the requirements
2. Some of the team member knows the requirements
3. Only Top level management knows the requirements
4. Some other organization developed the product but it is new to current organization
5. Need to do R&D and then proceed further

ATDD:-

* Unstructure requirement document will be served to development and testing team
* Convert the unstructure requirement document into acceptance tests (in the gherkins format)

Feature: Login

Scenario: Valid Login

**Given** I have browser with fb page

**When** I enter username as ‘jack’

**And** I enter password as ‘jack123’

**And** I click on login

**Then** I should get access to news feed with title as ‘News Feed’

Scenario: Invalid Login

**Given** I have browser with fb page

**When** I enter username as ‘jack1’

**And** I enter password as ‘jack123’

**And** I click on login

**Then** I should not get access to news feed with error message as “Invalid Credentials”

Cucumber Architecture

Feature (.feature) 🡪 step definition (.java)

Steps to create BDD framework using Java

1. Create maven project – provide groupid and artifact id
2. Add dependency required
   1. Cucumber Java - <https://mvnrepository.com/artifact/io.cucumber/cucumber-java/7.11.1>
   2. Cucumber TestNG - <https://mvnrepository.com/artifact/io.cucumber/cucumber-testng/7.11.1>
   3. Selenium - <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java/4.8.1>
3. Cucumber plugin should be install in eclipse
4. Create a feature file (.feature)
   1. Feature header
   2. Feature description
      1. Unformatted desc
      2. Formatted desc

In order to [Business goal]

As a [Role]

I want to [visible change in the application]

1. Create a Scenario
   1. Provide scenario title
   2. Add given, when, then
2. Create a runner file under src/test/java 🡪
   1. helps to trigger feature file
   2. helps to connect feature file and step definition

*Create a runner file and mention feature file details*

package com.citi.runner;

import io.cucumber.testng.AbstractTestNGCucumberTests;

import io.cucumber.testng.CucumberOptions;

@CucumberOptions(

features = {"src/test/resources/Feature/Login.feature"}

)

public class RunnerTest extends AbstractTestNGCucumberTests {

}

1. *Create the step definition and mention in runner file*
2. Report
   1. Publish – on the web
   2. plugin – html, json, pretty, progress
3. Step paramertization
   1. To reduce duplicate step definition.
4. Tags 🡪 And, or, not

* Tags can be provided to scenario and also for feature
* @invalid 🡪 triggers feature or scenario contains @invalid tag
* @invalid or @valid 🡪 triggers feature or scenario contains either @valid or @invalid
* @valid and @high 🡪 triggers feature or scenario only that contains both tags
* @login and not @invalid 🡪 🡪 triggers feature or scenario that contains @login and should not contain @invalid

1. dryRun=true 🡪 to check missing step defintions
2. Hooks
   1. @Before 🡪 runs before each sceanrio
   2. @After 🡪 runs after each scenario whether it is passed or failed
3. Scenario Outline
   1. Helps to reduce duplicates in the feature file
4. Background
   1. Should be used only when you have repeated given for all scenario under one feature file
5. Datatable
   1. To send tabluar or bulk data to one step defintion
6. Scenario outline with datatable
7. Dependency injection

BDD Important Concepts:- **Improve Readability and Reusability**

1. Step parameterization – code resuse
2. Scenario outline – one scenario – multiple set of test data
3. Background – repeated given
4. Datatable – tabular data

Advantages of BDD

1. Living documentation
2. Collabrate easily with non-technical

Git Concepts:- Git is a [free and open source](https://git-scm.com/about/free-and-open-source) distributed version control system

Architecture

Project (local machine) 🡪 local repository (local machine) -> remote repository ( github, aws code commit, gitlab, bitbucket)

Steps to push to github using eclipse

1. Create local repo (project->team->share project->create repository)
2. Update local repo 🡪 Stage all and commit
3. Register the remote url using name origin
4. Update remote repo 🡪 push

Reference:-

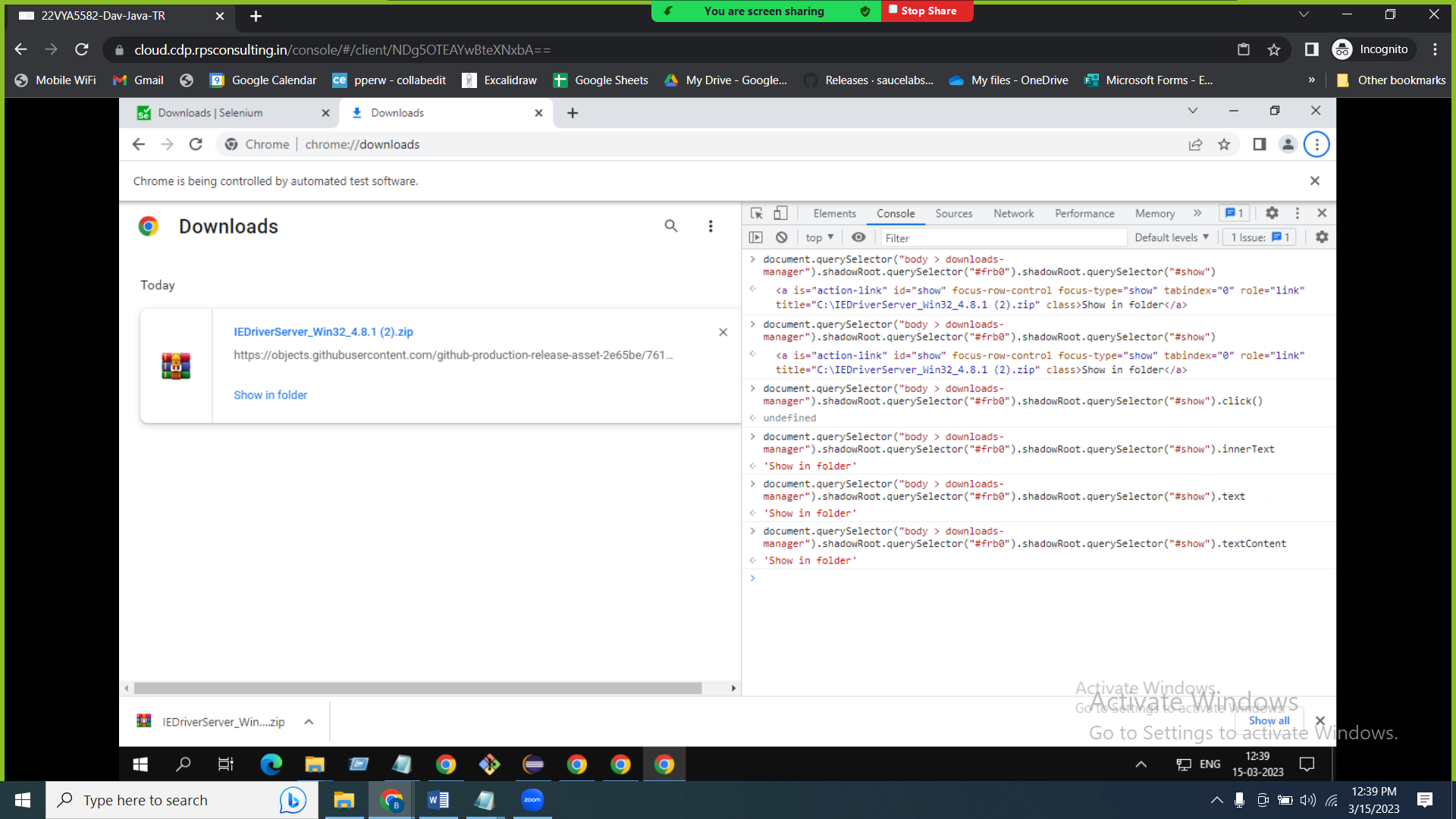
Selenium

1. ChromeOptions
2. Shadow elements (using javascript)
3. Selectorhub

Framework

* Keyword driven framework

Git concepts



@addpatient

Scenario: Add Valid Patient

Given I have browser with openemr page

When I enter username as 'admin'

And I enter password as 'pass'

And I select language as 'English (Indian)'

And I click on login

And I click on patient menu

And I click on new-search menu

And I fill the patient record

| firstname | lastname | gender | dob |

| John | Wick | Male | 2023-03-14 |

And I click on create new patient

And I click on confirm create new patient

And I store the alert text and handle it

And I close happy birthday popup if available

Then the alert message should contatin 'Tobacco'

And I should get the added patient name 'John Wick'