Pre-requisite – Functional testing and Java Programming knowledge.

Stage 1 – Selenium (Web Automation) & Restassured (API)

Stage 2 – Framework (Data Driven, Keyword Driven, TestNG, Page Object Model)

Stage 3 – Git & Jenkins (CI pipeline)

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

* Automate only web application
* Open Source
* Language Independent – Java, C#, Python, Javascript, Ruby

Selenium – A suite of tools

1. Selenium IDE
   * No Programming knowledge is required.
   * Record and playback features
   * Plugin – chrome, firefox, edge
   * Use it only for simple or exploratory testing
2. Selenium RC – Depreciated
   * Programming knowledge is required.
   * Architecture

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

1. Selenium WebDriver
   * Programming knowledge is required.
   * Architecture

Source code (Java+Selenium WebDriver) 🡪 Browser

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

Selenium IDE – Overview

UpperCamelCase – MyFirstProject

lowerCamelCase – myFirstProject

Structure of Java Project (Eclipse)

Workspace

Project (UpperCamelCase)

Package (lowercase – com.companyname.purpose)

Class (UpperCamelCase)

Methods & Variables (lowerCamelCase)

Selenium WebDriver

1. Create a java project
2. Download and configure the Selenium jar
3. Get title, url, pagesource
4. Click, Type, Select
5. Inspect -> tagname, attribute, text or not
6. Basic locators
   1. Id
   2. Name
   3. Classname
   4. Tagname
   5. Linktext
   6. Partial link text

When there are duplicate locator then findElement will pick the first element

1. Advance locators
   1. XPath
   2. CSS
2. To inspect 🡪 f12 or ctrl+shift+c
3. Dropdown
   1. With Select tag (Use Select class)
      1. selectByVisibleText()
      2. selectByValue()
      3. selectByIndex() 🡪 starts at 0
   2. Without select tag
      1. Click()
4. Click 🡪 Element should be present and visible
5. ElementInterceptedException – element is hidden by some other element
6. ElementNotInteractable – element is present but not visible
7. Synchronization
   1. Unconditional wait (java)
      1. Thread,sleep(5000) 🡪 wait for given ms
   2. Conditional wait (Selenium libs)
      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 exception
            2. If element is present, it will proceed to do operation immediately if any.
            3. Polling time – 500ms (0.5s)
      2. Explicit wait
         1. Exact condition
         2. Polling time – 500ms
      3. Fluent wait
8. Multiple tabs/windows, frame, alert – switchTo()
9. Multiple tabs/windows
   1. driver.getWindowhandles() 🡪 gets all the session id
10. List vs Set
    1. List 🡪 can contain duplicates.
    2. Set 🡪 cannot contain duplicates.
11. quit vs close
    1. close – close the current session/current tab
    2. quit – close all the session/current browser and also it kills the process associated to it.
12. Alert
    1. Used for handling javascript alerts
13. Frame – An html embedded into another html
    1. Even though the locator is correct, it will throw NoSuchElementException.
    2. Check for tagname – iframe or frame
    3. Switch to frame (anyone)
       1. Using index 🡪 starts at 0
       2. Using name or id as String
       3. Using WebElement
14. Upload
    1. Check for //input[@type='file']
15. Actions – Mouse/Keyboard activities
16. Javascript
    1. Click on hidden elements
    2. Type on read only textbox
    3. Scroll to element
    4. Scroll page

Git - Git is a free and open source distributed version control system

Architecture

Project (local machine) 🡪 Local repository (local machine) 🡪 remote repository (github, bitbucket, aws code commit)

Modified – staging – commit

Steps to track the code to remote

1. git init 🡪 initialize the local repository
2. git add . 🡪 staging (files need to be tracked)
3. git commit -m “message” 🡪 update the local repository
4. git remote add origin <https://github.com/balaji-githubstore/java-selenium-mar-2024-allianz.git> --> Register the remote url with name origin
5. git push -u origin master 🡪 update the remote repository

Framework

1. Unit Test Framework (TestNG)
2. Data Driven Framework (separating test method and test data in different files)
3. Page Object Model – Design pattern

Keyword Driven Framework – for effectively working with webdriver keywords

Maven – Build Management Tool

* Pom.xml (Project Object Model)
  + Will help to configure the jars and it will automatically download it including dependent jars.
  + Goals – helps to generate jars easily.
  + Easily control the project using command line.

Packages

com.allianz.test – test class and test methods

com.allianz.base – browser and report configurations

com.allianz.utils – reusable code for excel, json, csv handling.

com.allianz.pages – page object class and methods

Steps to create a framework:

1. Create a maven project
   1. Add groupid and artifact id (project name)
2. Add dependencies required for the framework (jars)
   1. TestNG - <https://mvnrepository.com/artifact/org.testng/testng>
   2. Selenium - <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>
3. Create one test class and test methods (@Test)
4. Install testng for eclipse plugin – to run the @Test method

A screenshot of a computer

Description automatically generated

1. @Test method will triggered based on the ASCII keycode order.

<http://sticksandstones.kstrom.com/appen.html>

@Test methods without priority will be given higher preference.

1. invocationCount – to trigger the @Test multiple times
2. Configure the code to your git repository - <https://github.com/balaji-githubstore/hybrid-framework-apr-2024-allianz.git>
3. TestNG generates report
   1. Emailable report
   2. Consolidated report (index.html)
4. Every @Test method should have minimum one validation (assertion) – decides whether the test method is pass or fail.
5. Annotations
   1. @BeforeMethod – runs before each @Test method
   2. @AfterMethod – runs after each @Test method whether it pass or fail
6. Use inheritance to configure the @BeforeMethod and @AfterMethod so that the same can be reused by the Test class (child class).
7. DataDriven using **DataProvider** – helps to run the @Test with multiple set of test data.
   1. Create a @Test method with proper arguments.
   2. Create a method that returns two-dimensional array and also provide @DataProvider annotation.
   3. Connect the @Test with @DataProvider.
8. Excel Read
   1. Add dependencies
      1. poi - <https://mvnrepository.com/artifact/org.apache.poi/poi>
      2. poi-ooxml - <https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml\>

Read Excel

1. Location – read/write
2. Open
3. Sheet
4. Row
5. Cell

.xlsx - XSSFWorkbook

.xls – HSSFWorkbook

1. Excel with @DataProvider
2. Page Object Model
   1. Reusability
   2. Maintenance
   3. Readability

Steps to achieve the page object mode:

1. For each webpage, need to create a class – Page class.
2. Operation should happen through methods – Page Methods.
3. Collecting the object repository (locators) at class level or different file.