Stage 1 – C# Programming

Stage 2 – Web Automation (Selenium WebDriver) and Windows GUI Automation (AutoIT)

Stage 3 – Framework

Stage 4 – Git

Lunch – 1:00 to 2:00 PM

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

* Automate web based application
* Open Source
* Language Independency – Java, C#, Python, Javascript, Ruby

Selenium – A suite of tools

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

Source code (C#+Selenium RC lib) 🡪 RC Server (Turn ON/OFF) 🡪 Browser

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

Source code (C#+Selenium WebDriver lib) 🡪 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 IDE – Overview

1. Install Selenium IDE
2. Start the recording, do some operation on browser, stop the recording
3. Run those recording
4. Analyse the recorded script
5. Every test case should have minimum one validations. (assertions)
6. Verify (soft assertion) vs Assert (hard assertion)
   1. Verify – compared actual value vs expected value. The test will continue even fails
   2. Assert - compared actual value vs expected value. The test will stop on failure.

Base Url - <https://demo.openemr.io/b/openemr>

C# - Programming

1. Installation
   1. Visual Studio IDE (Community)
      1. Open and choose .net desktop development from installer and install it.
2. Architecture

Console app project

Source code (.cs) (.csproj) 🡪 .dll/.exe (MSIL) 🡪 O/P

Class Library/Unit test project

Source code (.cs) (.csproj) 🡪 .dll (MSIL) 🡪O/P

1. Structure of C# project

Solution - UpperCamelCase

Project - UpperCamelCase

.cs file - UpperCamelCase

Namespace - UpperCamelCase

Class - UpperCamelCase

Methods – UpperCamelCase

Variable - lowerCamelCase

1. Datatypes
   1. Pre-defined datatypes
   2. Non-Predefined datatypes – collection of pre-defined datatypes
      1. String
      2. Array
2. Debugging
   1. Continue (Resume)
   2. Terminate
   3. Step into
   4. Step over
3. Methods – building block of the program.
   1. Reusability
   2. Maintenance

Methods

* Static Methods
  + How to create it?
  + How to call it?
    - Classname.Methodname()

accessmodifier static returntype methodname(arguments)

* Non-static Methods
  + How to create it?
  + How to call it?
    - Create object
    - Objref.MethodName()

1. Access modifier
   1. Private – can be accessed within the class
   2. Internal – Within the same assembly (project)
   3. Protected – can be accessed in the child classes
   4. Protected internal – combination
   5. Public – can be accessed from anywhere

Default 🡪

Class 🡪 internal

Method, fields 🡪 private

1. Object creation
   1. Declaration
   2. Instantiation (new)
   3. Initialization
2. Variable
   1. Static variable
   2. Non-static variable
3. Static vs non-static
4. Class & Object
   1. Class – a class is a template or blueprint or type from which object are created
   2. Object
      1. An object is an instance of class.
      2. Every object has its own state (non-static variable) and behaviour (non-static method)
5. Encapsulation
   1. Properties in C#
      1. Get 🡪 read (return)
      2. Set 🡪 write
6. Constructor – pre-requisite for the object
   1. Constructor name should be same as class name.
   2. It is kind of method with no return type.
   3. There will be always a default constructor that helps to load all non-static variable with default values.
   4. It will be called whenever object is created.
   5. When constructor created explicitly then we need to call it.
   6. Can create constructor with
      1. With arguments
      2. Without arguments
7. this keyword –
   1. helps to distinguish to non-static variable from local variables
   2. this keywords points to the current object
8. Can create multiple constructor – static polymorphism
   1. By change in datatypes
   2. By change in sequence of datatypes
   3. By change in number of parameters
9. Method overloading/static polymorphism/compile time polymorphism

Can create multiple method with same name – static polymorphism

* 1. By change in datatypes
  2. By change in sequence of datatypes
  3. By change in number of parameters

Method to be called is resolved during the compile time.

1. Inheritance
   1. Reuse
   2. When we need to reuse the variables and methods from one class then we can implement inheritance instead of re-creating those variables and methods in child class.
2. base keyword – only when parent class constructor is with arguments then use base() keyword and supply the arguments.
3. Whenever we inherit a class then check for
   1. parent class constructor is with argument or not. If it is with argument then use base() and supply the parent class constructor arguments

Selenium WebDriver

1. Create a console app or **Nunit Test Project**
2. Update and Add new libraries (.dll) through nuget packages
   1. WebDriver (Selenium.WebDriver)
   2. WebDriver.Support (Selenium.Support)
3. Navigate to url, get the title, get the url, get the pagesource, close the browser
4. Click, type, Select
5. Inspect 🡪 tagname, attribute, text or not

<a href="/reg/" title="Sign up for Facebook">Sign Up</a>

1. To inspect 🡪 f12 or ctrl+shift+c
2. Basic Locators
   1. Id
   2. Name
   3. Class
   4. Tagname
   5. Linktext
   6. Partial linktext

When there are duplicate locator, FindElement picks the first element.

1. Advance locators
   1. XPath
   2. CSS
2. Dropdown
   1. With select tag – class -SelectElement
      1. selectByText()
      2. selectByValue()
      3. selectByIndex() 🡪 starts at 0
   2. Without select tag
      1. Click()
3. FindElement – takes only 0.5 seconds to check for presence of element.
4. Synchronization
   1. Unconditional wait
      1. Thread.Sleep(5000) 🡪 wait for 5s (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 exception

If element is present, it will do the operation immediately

Polling time – 0.5s (how freq check for locator)

* + 1. Explicit wait
       - 1. Exact condition
         2. Polling time – 0.5s
    2. Fluent wait

1. ElementClickInterceptedException – target element is hidden by some popup

ElementNotIteractable – element is present but not visible

1. Multiple tabs/windows, frame, alert – switchTo()
2. Close vs quit
   1. Close – close the current tab/ current session
   2. Quit – close the browser and also kill the process associated to it.
3. CSS Selector
4. Alert
   1. Javascript alert
5. Frame – html embedded into another html
   1. Even though the locator is correct, we get NoSuchElementException
   2. Check for tagname frame or iframe
   3. Switch to frame using (anyone)
      1. Name or id as String
      2. Index – starts at 0
      3. WebElement
6. Upload the file
7. Actions – mouse & keyboard actions
   1. May not throw proper error
   2. May not work in headless mode
   3. Do not disturb the mouse/keyboard

Keyboard activities

Modifier keys – ctrl, alt, shift -🡪 keyDown() & keyUp()

Rest 🡪 sendKeys()

1. Javascript
   1. Click on hidden elements
   2. Type on readonly textbox

Framework: -

* + - 1. Unit Test Framework – Nunit
      2. Data Driven Framework – Seprating the test methods and test data (excel, csv, json)
      3. Page Object Model – Design pattern – For efficiently handling the webelement

Keyword Driven Framework – for keeping the webdriver keywords

Steps to create a hybrid framework

1. Create new Nunit project
2. Add/update libraries from nuget package
   1. Selenium.WebDriver (WebDriver.dll)
   2. Selenium.Support (Webdriver.Support.dll)
3. Create Test folder, Create Test Class and Create [Test] method
4. Every [Test] method should have minimum one assertion. Assertion decides whether [Test] is pass or fail.
5. Nunit Attributes
   1. [Setup] – runs before each [Test] method (browser launch)
   2. [Teardown] – runs after each [Test] method even though [Test] fails (close the browser)
6. Use inheritance to configure – browser launch and close
7. Use [TestCase] to run one [Test] method with multiple set of [Test] attributes.
   1. Create a [Test] method with proper arguments and use it inside the methods
   2. Create [TestCase] with test data.
8. Data Driven Framework – Separate the [Test] method from its test data in two different files. (Excel, csv, json or at lease separate csharp file)
9. Using [TestCaseSource]
   1. Create a [Test] method with proper arguments and use it inside the methods
   2. Create a static method that return two-dimension array (with all test data)
   3. Connect the [Test] with static methods
10. Using [TestCaseSource] and Excel
11. Page Object Model – Design pattern
    1. Reusability
    2. Maintenance
    3. Readability

Steps to achieve the page object model:-

1. For each webpage, we will create a class – Page Class
2. Operation happens through the methods – Page Method
3. Collecting object repository at class level or different file

Git 🡪 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 🡪 Remote repository (github, aws code commit, bit bucket)

Modified 🡪 staged🡪 commit

Local repo:

git init 🡪 **create a local repository**

git add . 🡪 **track the file need to be committed (staging)**

git commit -m "first commit" 🡪 **updating the local repo**

Remote Repo

git remote add origin <https://github.com/balaji-githubstore/selenium-C--July-2024-kpmg-.git> 🡪 **register the remote url using name origin**

git push -u origin master 🡪 **use origin to push the code**

Reference

1 bit -🡪 0 or 1

1 GB – 1024 MB

1 MB – 1024KB

1KB – 1024B

1B – 8 bits

Reference

1. If we need reuse some logics 🡪 Use methods
2. If we want to reuse methods/variables 🡪 inheritance

# Task 1

1. Create project CSharpProject3
2. Create a class with name as “**Volume**”
3. Create static method for below formula (any 4 methods)

|  |  |
| --- | --- |
|  | [Volume Of A Cylinder](https://byjus.com/maths/volume-of-a-cylinder/) |
| [Volume Of Cuboid](https://byjus.com/maths/volume-of-cuboid/) | [Volume Of Hemisphere](https://byjus.com/maths/volume-of-hemisphere/) |

1. Create a Program class and main method
   1. Call the VolumeOfCylinder() method
   2. Call the VolumeOf Hemisphere() method

1. C# - MCQ

https://forms.office.com/r/dLjJR1n6W5

2. Create a framework with folder base, test, utilities, pages

https://opensource-demo.orangehrmlive.com/web/index.php/auth/login

1. ValidLoginTest

2. InvalidLoginTest

Data 1:

Username: John

Password: admin123

Invalid credentials"

Data 2:

Username: Peter

Password: admin123

Invalid credentials

1. AddEmployeeTest
2. Navigate to the url  
   2. Enter Admin username  
   3. Enter password  
   4. Click Login  
   5. Mouse over on PIM  
   6. Click on Add Employee  
   7. Enter firstname as John  
   8. Enter middlename as W  
   9. Enter lastname as Wick  
   10. Upload the employee image (any image)  
   11. Select status as Enabled  
   12. Click on save

13. Assert the profile name “**John Wick”**