What is Automation Framework?

* Defines way of doing things
* Common and structured standards of coding
* Increase in speed and accuracy of automation process
* Every member of the team should follow the same standards

Page Object Model:

* It’s a design pattern to structure the tests to create web UI elements and actions
* One page class for each web page in the application
* Page class – Find the WebElements of web page
* Page class – contains Page methods which perform actions on those webelements

Advantages:

* Makes our code cleaner and easy to understand
* Tests are more readable, flexible and maintainable
* Re-usable code that reduces redundancy of code

Framework structure:

Base (base classes)

Pages (Page classes)

Tests (test classes)

Utility (Utility classes)

Configuration files

screenshots

\*\*\* Import Webdriver module from selenium package

\*\*\* Instance of browser webdriver created using driver = Webdriver.Firefox/Chrome/IE

\*\* Maximize the window 🡪 driver.maximize\_window()

\*\* Add implicitly wait to driver 🡪 driver.implicitly\_wait()

Note: In each class(base/page..) – constructor \_\_init\_\_ is must. It accepts driver instance as parameter

Note: All test methods – import unittest module, inherit ‘unittest.TestCase’

If test case needs to run as unit test then we need to provide the following

If \_\_name\_\_==”main”:

Unittest.main()

But for Pytest the above is not required, we can run it from terminal/command prompt

Every page class we create should inherit base – SeleniumDriver class – In this case definitely super().\_\_iniit\_\_(driver) is needed because parent –inherit class also has constructor

Import custom logger function from Utlity package and use it in the selenium driver class as self.log.info(). This custom logger function returns logger which has an ability to write the logs in File with specific format

**Conftest test file for common setup:** Should be under tests package folder

We can initiate driver instance, maximize window and define implicity wait in OneTimeSetup method and also this returns driver instance

After yield keyword – in teardown define driver.quit() – This will quit the browser after each test class methods end

Also we can define class level setup in respective test page class

Example – create login page object in login test class – This can be defined in test class

@pytest.fixture(autouse = True)

def classSetup(Self, OneTimeSetup):

lp = LoginPage(self.driver)

WebDriverFactory.py under base package– This is a WebDriver factory class implementation

It creates a Webdriver instance based on browser configurations (constructor accepts browser as an argument)

getWebDriverInstance method – it contains baseURL and initialize the driver based on browser, maximize window, implicity wait and launch URL and return driver instance

we can create an object for Webdriver factor in Conftest class – OneTimeSetup method

**Assertion without stopping the execution:**

One class for assertion – Teststatus class under utility package – Inherit selenium driver class

Constructor \_\_init\_\_ 🡺 call super call and set resultList as blank list

Methods:

mark - Mark the result of the verification point in a test case

markFinal – Mark the final result of the verification point in a test case. This needs to be called atleast once in a testcase, this should be final test status of the test case

SetResult

**Screenshot:**

Save\_Screenshot 🡪 predefined method to use to take the screenshot in selenium

Screenshot method is defined in Selenium driver class

Filename – contains random generated value

Find the Screenshot directory - “../screenshots/”

relativeFileName 🡪 screenshotdirectory+filename

Find the CurrentDirectory 🡪 os.path.dirname(\_\_file\_\_) – This will provide directory name of current file

Destination File 🡺 os.path.join(currntDirectory, relativeFilename)

Destination Directory 🡪 os.path.join(currentDirectory, screenshotDirectory)

This method is effectively used in teststatus class under utility package

BasePage class contains methods which are common to all pages – example)

Verification of page title

Login is success

Login is failed

Constructor – driver instantiation and also create a instance for util class

Util class -> All most commonly used utilities are implemented in this class

Sleep

getUniqueName

getAlphanumeric

Selenium - Python Project:

**Components:**

Package1: Base Package

Package2: Pages

Package3: Screenshots

Package4: Utilities

Package5: tests

**Files:**

Package1: Base Package:

* Selenium\_library with generic APIs

Package2: Pages:

Pages with element IDs

Package3 : Screenshots

Failed screenshots will be saved under this folder

Package4: Utilities

Custom\_logger : Customized function where we will set the log level and define the format of the log file using File Handler and Set Formatter methods. This will return logger and that will be used in other class files

TestRunStatus: -

Utility class file  
CheckPoint class implementation  
It provides functionality to assert the result

3 methods:

set\_result

mark\_result - Mark the result of the verification point in a test case

mark\_final\_result - Mark the final result of the verification point in a test case. This needs to be called at least once in a test case . This should be final test status of the test case

**List of generic APIs:**

# 1. launch\_url  
# 2. verify\_title  
# 3. get\_by\_type  
# 4. get\_element  
# 5. enter\_text  
# 6. wait\_for\_element  
# 7. element\_click  
# 8. is\_element\_present  
# 9. element\_clear  
#10. is\_element\_displayed  
#11. is\_element\_enabled  
#12. scroll\_browser  
#13. select\_from\_list  
#14. get\_elements  
#15. verify\_text\_contains  
#16. verify\_text\_match  
#17. verify\_list\_match  
#18. verify\_data\_exists\_in\_list