Pytest(Hybrid framework):

Requirements:

**selenium lib**

**pytest :python unittest framework**

**pytest-html: PyTest html reports**

**pytest-xdist: Run Test Parallel**

**Openpyxl: MS Excel support**

**Allure-pytest: to generate allure reports**

**Structure:**

**Testcases:**

**Tests\_**

**Conftest.py: pytest.fixtures(): executes before every methods**

**package**

**pageObjects:**

**utilities: will read the common data from ini file and provide data to the testcase to re-use data**

**Readproperties.py:**

**Customlogger.py : add logs, basiconfigaurations and add logs to the TC**

**TestData: folder**

**Configurations: f**

**Config.ini: To store the common data**

**Logs: f**

**Reports: f**

**Screenshots: f**

**Run.bat:**

self.driver.save\_screenshot(".\\screenshots"+ "test\_login.png")

**Executes in terminal:**

**(pom-project) C:\Users\DELL\PycharmProjects\pom-project>pytest -v -s TestCases/test\_login.py**

**pytest -s -v TestCases/test\_login.py --browser chrome to run in desired browser**

**pytest -s -v -n=2 TestCases/test\_login.py --browser firefox to run in browsers in parallel**

**pytest -s -v -n=2 --html=reports\report.html TestCases/test\_login.py --browser chrome to generate html reports.**

# hook for adding env info to HTML report

def pytest\_configure(config):  
 config.\_metadata['project Name'] = 'nop Commerce'  
 config.\_metadata['Module Name'] = 'Customers'  
 config.\_metadata['Tester'] = 'jyothi'  
  
# hook for del/modify env info to HTML report  
@pytest.mark.optionalhook  
def pytest\_metadata(metadata):  
 metadata.pop("JAVA\_HOME", None)  
 metadata.pop("Plugins", None)

DDTestcase:

* Data in excel and place in TestData folder
* Create ExcelUtils.py under utilities package
* Create LoginDataDriverTest under Testcases
* And Run Testcase

Import XLUtils under utilities package

Q) How to generate random test data:

2 types of test data:

Static data: which we will prepare before executing testcases like hardcoded the data in excel for login

Dynamic data: which will generate automatically in run time

To validate the registration is done successfyll:

self.msg = self.driver.find\_element(By.TAG\_NAME,"body").text  
print(self.msg)  
if "The new customer has been added successfully." in self.msg:  
 assert True == True  
 self.logger.info("add customer test is passed")  
else:  
 self.driver.save\_screenshot(".\\screenshots\\" + "test\_Addcustomer\_scr.png")  
 self.logger.info("add customer test is failed")

To generate random data while testing:

Ex: when we need to test with the multiple random emailID’s for registration account

self.email = random\_generator() + "@gmail.com"  
self.addcust.setemail(self.email)

below is the method we call to generate email ID’s

**def random\_generator** method is enough

and it will return the random choice as below

def random\_generator(size=8, chars=string.ascii\_lowercase + string.digits):  
 return ''.join(random.choice(chars) for x in range(size))