**End to End Python Selenium Framework**

* Standards of writing selenium tests in framework
* Creating Browser invocation Fixtures in conftest.py
* Setting up Base Class to hold all common Utilities
* Inheriting Base Class to all tests to remove Fixtures redundant code
* Passing Command Line options to select browser at run time
* Implementing Page Object Mechanism
* Smarter way of returning Page Objects from Navigation Methods
* Creating Selenium WebDriver Custom Utilities in Base Class
* Parameterizing WebDriver tests with Multiple Datasets
* Organizing Data from Separate Data files and injecting into fixture at run time
* Implementing Logging feature to WebDriver tests
* Text Execution Html reporting
* Automatic Screenshot Capture on Test failures
* Integrating Selenium Python Framework to Jenkins CI tools with Jenkin build Parameterization
* GitHub Basics for Project version control

**Creating Python Selenium Framework**

1. **Standards of writing selenium tests in Framework –**
   1. Let’s create repo for end-to-end python selenium framework and we are naming in **python-selenium-framework**.
   2. Clone the repo to local machine and then create a package named **test** in it. (A package in python basically is a folder which contains \_\_init\_\_.py empty file)
   3. Then create a file named **test\_e2e.py** and open it in visual studio code or any editor. We will write every single test case in it using pytest standards.
   4. As per the standards the code should be inside a function/method and that method should be inside the class, let’s name that class **TestOne()** and the method **test\_e2e(self)**.
   5. Now copy the whole selenium code of **shopping\_phones.py** file of Notes > Selenium > By Me, inside that **test\_e2e** function.
   6. It will look like this –

***from* selenium *import* webdriver**

***from* selenium.webdriver.chrome.service *import* Service**

***from* selenium.webdriver.common.by *import* By**

***from* selenium.webdriver.support.wait *import* WebDriverWait**

***from* selenium.webdriver.support *import* expected\_conditions *as* EC**

**class TestOne():**

**def test\_e2e(*self*):**

**browser = webdriver.Chrome(*service*=Service("C:\\Users\\sande\\Desktop\\Programs\\chromedriver.exe"))**

**browser.get('https://rahulshettyacademy.com/angularpractice/')**

**wait = WebDriverWait(browser, 10)**

***#click shop button***

**browser.find\_element(By.CSS\_SELECTOR, "a[href\*=shop]").click()**

***#getting all the products in page***

**products = browser.find\_elements(By.XPATH, "//div[@class='card h-100']")**

***for* product *in* products:**

***if* product.find\_element(By.XPATH, "div/h4/a").text == 'Blackberry':**

**product.find\_element(By.XPATH, "div/button").click()**

***#checkout button***

**browser.find\_element(By.CSS\_SELECTOR, "a[class\*='btn-primary']").click()**

**browser.find\_element(By.CSS\_SELECTOR, "button[class\*='btn-success']").click()**

***#input box - entering "ind"***

**browser.find\_element(By.CSS\_SELECTOR, "#country").send\_keys('ind')**

***#wait till the link text India is present and then click on it***

**wait.until(EC.presence\_of\_element\_located((By.LINK\_TEXT, "India")))**

**browser.find\_element(By.LINK\_TEXT, "India").click()**

***#click on t&c checkbox and then on purchase button***

**browser.find\_element(By.XPATH, "//div[@class='checkbox checkbox-primary']").click()**

**browser.find\_element(By.CSS\_SELECTOR, "[type='submit']").click()**

***#get the success msg printed on webpage verify it if it's matching***

**success\_text = browser.find\_element(By.CLASS\_NAME, "alert-success").text**

***assert* "Success! Thank you!" *in* success\_text**

***#taking a screenshot of the page***

**browser.get\_screenshot\_as\_file('ss.png')**

**browser.quit()**

1. **Creating browser invocation fixtures in conftest.py –**
   1. The browser invocation code should not be in our **test\_e2e.py** file, means it should be separated from our selenium code.
   2. To do that we will first create a file named **conftest.py** in **test** package. (Our all the fixtures should be in that conftest.py file)
   3. We will first define the method named **setup** and inside it we will put the code which will handle the browser invocation and we will put the marking **(@pytest.fixture**) before the method definition. We will import the **webdriver, service** and **pytest** first so we don’t get the error. It will look like this –

***from* selenium *import* webdriver**

***from* selenium.webdriver.chrome.service *import* Service**

***import* pytest**

**@pytest.fixture(*scope*='class')**

**def setup():**

**browser = webdriver.Chrome(*service*=Service("C:\\Users\\sande\\Desktop\\Programs\\chromedriver.exe"))**

**browser.get('https://rahulshettyacademy.com/angularpractice/')**

**browser.maximize\_window()**

* 1. But there is a flaw in this code that is- we can’t return this browser driver to **test\_e2e.py** class and one more thing should be here which is yield keyword for the teardown of our program which is closing the browser.
  2. So, to overcome that issue we can use request instance available for fixtures and our code will look like this-

***from* selenium *import* webdriver**

***from* selenium.webdriver.chrome.service *import* Service**

***import* pytest**

**@pytest.fixture(*scope*='class')**

**def setup(*request*):**

**browser = webdriver.Chrome(*service*=Service("C:\\Users\\sande\\Desktop\\Programs\\chromedriver.exe"))**

**browser.get('https://rahulshettyacademy.com/angularpractice/')**

**browser.maximize\_window()**

***#whichever class request browser pass this browser***

***request*.cls.browser = browser**

***yield***

**browser.close()**

* 1. Now in our **test\_e2e.py** we have to set it this fixture marking before the class name and to access the browser driver we have to use *self.browser* and the code will look like this-

***#shopping\_phones***

***from* selenium *import* webdriver**

***from* selenium.webdriver.chrome.service *import* Service**

***from* selenium.webdriver.common.by *import* By**

***from* selenium.webdriver.support.wait *import* WebDriverWait**

***from* selenium.webdriver.support *import* expected\_conditions *as* EC**

***import* pytest**

**@pytest.mark.usefixtures("setup")**

**class TestOne():**

**def test\_e2e(*self*):**

***# we defined browser opening inside the 'setup method' in 'conftest.py'***

***#  file we will use it with the help of fixture.***

**wait = WebDriverWait(*self*.browser, 10)**

***#click shop button***

***self*.browser.find\_element(By.CSS\_SELECTOR, "a[href\*=shop]").click()**

***#getting all the products in page***

**products = *self*.browser.find\_elements(By.XPATH, "//div[@class='card h-100']")**

***for* product *in* products:**

***if* product.find\_element(By.XPATH, "div/h4/a").text == 'Blackberry':**

**product.find\_element(By.XPATH, "div/button").click()**

***#checkout button***

***self*.browser.find\_element(By.CSS\_SELECTOR, "a[class\*='btn-primary']").click()**

***self*.browser.find\_element(By.CSS\_SELECTOR, "button[class\*='btn-success']").click()**

***#input box - entering "ind"***

***self*.browser.find\_element(By.CSS\_SELECTOR, "#country").send\_keys('ind')**

***#wait till the link text India is present and then click on it***

**wait.until(EC.presence\_of\_element\_located((By.LINK\_TEXT, "India")))**

***self*.browser.find\_element(By.LINK\_TEXT, "India").click()**

***#click on t&c checkbox and then on purchase button***

***self*.browser.find\_element(By.XPATH, "//div[@class='checkbox checkbox-primary']").click()**

***self*.browser.find\_element(By.CSS\_SELECTOR, "[type='submit']").click()**

***#get the success msg printed on webpage verify it if it's matching***

**success\_text = *self*.browser.find\_element(By.CLASS\_NAME, "alert-success").text**

***assert* "Success! Thank you!" *in* success\_text**

***#taking a screenshot of the page***

***self*.browser.get\_screenshot\_as\_file('ss.png')**

1. **Setting up Base Class to hold all common utilities and Inheriting BaseClass to all tests to remove Fixtures redundant code–**
   1. Now we will put the fixture marking (@pytest.mark.usefixtues) in baseclass for that we will first create the **utilities** package.
   2. To create the package, we will simply make a new folder name utilities and create an empty **\_\_init\_\_.py** file inside it.
   3. Now we will create the **BaseClass.py** file and we will define the fixture inside it, it will look like this-

***import* pytest**

***#setup method comes from conftest.py***

**@pytest.mark.usefixtures('setup')**

**class BaseClass:**

***pass***

* 1. After this we can inherit the baseclass to test\_e2e.py class and other class too when we need this fixture, we don’t have to put @pytest before method definition now. The code will look like this-

***from* selenium.webdriver.common.by *import* By**

***from* selenium.webdriver.support.wait *import* WebDriverWait**

***from* selenium.webdriver.support *import* expected\_conditions *as* EC**

***from* utilities.BaseClass *import* BaseClass**

***import* pytest**

**class TestOne(BaseClass):**

**def test\_e2e(*self*):**

***# we defined browser opening inside the 'setup method' in 'conftest.py'***

***#  file we will use it with the help of fixture.***

**wait = WebDriverWait(*self*.browser, 10)**

***#click shop button***

***self*.browser.find\_element(By.CSS\_SELECTOR, "a[href\*=shop]").click()**

***#getting all the products in page***

**products = *self*.browser.find\_elements(By.XPATH, "//div[@class='card h-100']")**

***for* product *in* products:**

***if* product.find\_element(By.XPATH, "div/h4/a").text == 'Blackberry':**

**product.find\_element(By.XPATH, "div/button").click()**

***#checkout button***

***self*.browser.find\_element(By.CSS\_SELECTOR, "a[class\*='btn-primary']").click()**

***self*.browser.find\_element(By.CSS\_SELECTOR, "button[class\*='btn-success']").click()**

***#input box - entering "ind"***

***self*.browser.find\_element(By.CSS\_SELECTOR, "#country").send\_keys('ind')**

***#wait till the link text India is present and then click on it***

**wait.until(EC.presence\_of\_element\_located((By.LINK\_TEXT, "India")))**

***self*.browser.find\_element(By.LINK\_TEXT, "India").click()**

***#click on t&c checkbox and then on purchase button***

***self*.browser.find\_element(By.XPATH, "//div[@class='checkbox checkbox-primary']").click()**

***self*.browser.find\_element(By.CSS\_SELECTOR, "[type='submit']").click()**

***#get the success msg printed on webpage verify it if it's matching***

**success\_text = *self*.browser.find\_element(By.CLASS\_NAME, "alert-success").text**

***assert* "Success! Thank you!" *in* success\_text**

***#taking a screenshot of the page***

***self*.browser.get\_screenshot\_as\_file('ss.png')**

1. **Passing command line options to select browser at runtime–**
   1. To pass the argument which browser we want to open for our test we need to add adoption method in **conftest.py**, then we can get the browser name from command line and open a browser according to it using if and else, the code will look like this-

***from* selenium *import* webdriver**

***from* selenium.webdriver.chrome.service *import* Service**

***import* pytest**

***#add a command line option of '--browser name\_of\_browser'***

**def pytest\_addoption(*parser*):**

***parser*.addoption(**

**"--browser", *action*='store', *default*='chrome'**

**)**

**@pytest.fixture(*scope*='class')**

**def setup(*request*):**

***#this is how we can get the value of '--browser' key***

**browser\_name = *request*.config.getoption("browser")**

***if* browser\_name == "chrome":**

**browser = webdriver.Chrome(*service*=Service("C:\\Users\\sande\\Desktop\\Programs\\chromedriver.exe"))**

***elif* browser\_name == "firefox":**

**browser = webdriver.Firefox(*service*=Service("C:\\Users\\sande\\Desktop\\Programs\\geckodriver.exe"))**

***elif* browser\_name == 'edge':**

**browser = webdriver.Edge(*service*=Service("C:\\Users\\sande\\Desktop\\Programs\\msedgedriver.exe"))**

**browser.get('https://rahulshettyacademy.com/angularpractice/')**

**browser.maximize\_window()**

***#whichever class request browser pass this browser***

***request*.cls.browser = browser**

***yield***

**browser.close()**

* 1. And **test\_e2e.py** will look like this-

***from* selenium.webdriver.common.by *import* By**

***from* selenium.webdriver.support.wait *import* WebDriverWait**

***from* selenium.webdriver.support *import* expected\_conditions *as* EC**

***from* utilities.BaseClass *import* BaseClass**

**class TestOne(BaseClass):**

**def test\_e2e(*self*):**

***# we defined browser opening inside the 'setup method' in 'conftest.py'***

***#  file we will use it with the help of fixture.***

**wait = WebDriverWait(*self*.browser, 10)**

***#click shop button***

***self*.browser.find\_element(By.CSS\_SELECTOR, "a[href\*=shop]").click()**

***#getting all the products in page***

**products = *self*.browser.find\_elements(By.XPATH, "//div[@class='card h-100']")**

***for* product *in* products:**

***if* product.find\_element(By.XPATH, "div/h4/a").text == 'Blackberry':**

**product.find\_element(By.XPATH, "div/button").click()**

***#checkout button***

***self*.browser.find\_element(By.CSS\_SELECTOR, "a[class\*='btn-primary']").click()**

***self*.browser.find\_element(By.CSS\_SELECTOR, "button[class\*='btn-success']").click()**

***#input box - entering "ind"***

***self*.browser.find\_element(By.CSS\_SELECTOR, "#country").send\_keys('ind')**

***#wait till the link text India is present and then click on it***

**wait.until(EC.presence\_of\_element\_located((By.LINK\_TEXT, "India")))**

***self*.browser.find\_element(By.LINK\_TEXT, "India").click()**

***#click on t&c checkbox and then on purchase button***

***self*.browser.find\_element(By.XPATH, "//div[@class='checkbox checkbox-primary']").click()**

***self*.browser.find\_element(By.CSS\_SELECTOR, "[type='submit']").click()**

***#get the success msg printed on webpage verify it if it's matching***

**success\_text = *self*.browser.find\_element(By.CLASS\_NAME, "alert-success").text**

***assert* "Success! Thank you!" *in* success\_text**

***#taking a screenshot of the page***

***self*.browser.get\_screenshot\_as\_file('ss.png')**

1. **Implementing page object mechanism –**
   1. The complete test involves multiple pages for example in this testing framework homepage, checkout page and confirm page we have to put all operation objects in the separate class.
   2. First create a package in our directory, lets name it **pageObjects** inside that package we have to create the **\_\_init\_\_.py** file so it will act as a package then we will create three files for one **HomePage.py**, second **CheckoutPage.py** and the third is **ConfirmPage.py.**
   3. Now let’s work on **homepage** first, so open the **homepage.py** now and inside that file we will write the code to perform all the operations we want with selenium for homepage. The code will look like this-

***from* selenium.webdriver.common.by *import* By**

**class HomePage:**

***#to receive the driver/browser from the test\_e2e.py, we need to define the constructor***

**def \_\_init\_\_(*self*, *browser*):**

***self*.browser = *browser***

***#assigning the selector to shop variable for clicking the shop button***

**shop = (By.CSS\_SELECTOR, "a[href\*=shop]")**

**def getShopItems(*self*):**

***#that \* tells it to take it as webelement, it deserializes it as tuple***

***return* *self*.browser.find\_element(\*HomePage.shop)**

* 1. And this is the **test\_e2e.py** page it will look like this-

***from* selenium.webdriver.common.by *import* By**

***from* selenium.webdriver.support.wait *import* WebDriverWait**

***from* selenium.webdriver.support *import* expected\_conditions *as* EC**

***from* utilities.BaseClass *import* BaseClass**

***from* pageObjects.HomePage *import* HomePage**

***from* pageObjects.CheckoutPage *import* CheckOutPage**

***from* pageObjects.ConfirmPage *import* ConfirmPage**

**class TestOne(BaseClass):**

**def test\_e2e(*self*):**

**'''We defined browser opening inside the 'setup method' in 'conftest.py'**

**file we will use it with the help of fixture.'''**

**wait = WebDriverWait(*self*.browser, 10)**

***#passing browser/driver to homepage.py***

**homePage = HomePage(*self*.browser)**

***#click shop button***

**homePage.getShopItems().click()**

* 1. And then we will do the same with **chekoutpage.py** and the code will look like this-

***from* selenium.webdriver.common.by *import* By**

**class CheckOutPage:**

***#to receive the driver/browser from the test\_e2e.py, we need to define the constructor***

**def \_\_init\_\_(*self*, *browser*):**

***self*.browser = *browser***

***#assigning all the titles of the products/cards to cardTitle variable***

**cardTitle = (By.CSS\_SELECTOR, ".card-title a")**

***#add to cart button***

**addToCart = (By.CSS\_SELECTOR, ".card-footer button")**

***#checkout button***

**checkoutButton = (By.CSS\_SELECTOR, "a[class\*='btn-primary']")**

***#checkout button of next page***

**checkoutButtonNext = (By.CSS\_SELECTOR, "button[class\*='btn-success']")**

**def getCardTitles(*self*):**

***#that \* tells it to take it as webelement, it deserializes it as tuple***

***return* *self*.browser.find\_elements(\*CheckOutPage.cardTitle)**

**def getAddToCart(*self*):**

***return* *self*.browser.find\_elements(\*CheckOutPage.addToCart)**

**def getCheckoutButton(*self*):**

***return* *self*.browser.find\_element(\*CheckOutPage.checkoutButton)**

**def getCheckoutButtonNext(*self*):**

***return* *self*.browser.find\_element(\*CheckOutPage.checkoutButtonNext)**

* 1. And **test\_e2e.py** will look like this-

***from* selenium.webdriver.common.by *import* By**

***from* selenium.webdriver.support.wait *import* WebDriverWait**

***from* selenium.webdriver.support *import* expected\_conditions *as* EC**

***from* utilities.BaseClass *import* BaseClass**

***from* pageObjects.HomePage *import* HomePage**

***from* pageObjects.CheckoutPage *import* CheckOutPage**

***from* pageObjects.ConfirmPage *import* ConfirmPage**

**class TestOne(BaseClass):**

**def test\_e2e(*self*):**

**'''We defined browser opening inside the 'setup method' in 'conftest.py'**

**file we will use it with the help of fixture.'''**

**wait = WebDriverWait(*self*.browser, 10)**

***#passing browser/driver to homepage.py***

**homePage = HomePage(*self*.browser)**

***#click shop button***

**homePage.getShopItems().click()**

***#passing the browser/driver to checkoutpage.py***

**checkOutPage = CheckOutPage(*self*.browser)**

***#getting all the products/cards in page***

**products = checkOutPage.getCardTitles()**

**i = -1**

***for* product *in* products:**

**i = i + 1**

**productTitle = product.text**

***if* (productTitle ==  'Blackberry'):**

**checkOutPage.getAddToCart()[i].click()**

***#click checkout button***

**checkOutPage.getCheckoutButton().click()**

***#checkout button of next page***

**checkOutPage.getCheckoutButtonNext().click()**

* 1. And we will do the same with **confirmpage.py** and the code will look like this-

***from* selenium.webdriver.common.by *import* By**

**class ConfirmPage:**

***#to receive the driver/browser from the test\_e2e.py, we need to define the constructor***

**def \_\_init\_\_(*self*, *browser*):**

***self*.browser = *browser***

***#assigning the selector to shop variable for clicking the shop button***

**inputBox = (By.CSS\_SELECTOR, "#country")**

**def getInputBox(*self*):**

***#that \* tells it to take it as webelement, it deserializes it as tuple***

***return* *self*.browser.find\_element(\*ConfirmPage.inputBox)**

* 1. And **test\_e2e.py** will look like this-

***from* selenium.webdriver.common.by *import* By**

***from* selenium.webdriver.support.wait *import* WebDriverWait**

***from* selenium.webdriver.support *import* expected\_conditions *as* EC**

***from* utilities.BaseClass *import* BaseClass**

***from* pageObjects.HomePage *import* HomePage**

***from* pageObjects.CheckoutPage *import* CheckOutPage**

***from* pageObjects.ConfirmPage *import* ConfirmPage**

**class TestOne(BaseClass):**

**def test\_e2e(*self*):**

**'''We defined browser opening inside the 'setup method' in 'conftest.py'**

**file we will use it with the help of fixture.'''**

**wait = WebDriverWait(*self*.browser, 10)**

***#passing browser/driver to homepage.py***

**homePage = HomePage(*self*.browser)**

***#click shop button***

**homePage.getShopItems().click()**

***#passing the browser/driver to checkoutpage.py***

**checkOutPage = CheckOutPage(*self*.browser)**

***#getting all the products/cards in page***

**products = checkOutPage.getCardTitles()**

**i = -1**

***for* product *in* products:**

**i = i + 1**

**productTitle = product.text**

***if* (productTitle ==  'Blackberry'):**

**checkOutPage.getAddToCart()[i].click()**

***#click checkout button***

**checkOutPage.getCheckoutButton().click()**

***#checkout button of next page***

**checkOutPage.getCheckoutButtonNext().click()**

***#passing the browser/driver to confirmpage.py***

**confirmPage = ConfirmPage(*self*.browser)**

***#input box - entering "ind"***

**confirmPage.getInputBox().send\_keys('ind')**

* 1. Now we will do the same with the remaining web-element objects. The code will look like this-

***from* utilities.BaseClass *import* BaseClass**

***from* pageObjects.HomePage *import* HomePage**

***from* pageObjects.CheckoutPage *import* CheckOutPage**

***from* pageObjects.ConfirmPage *import* ConfirmPage**

**class TestOne(BaseClass):**

**def test\_e2e(*self*):**

**'''We defined browser opening inside the 'setup method' in 'conftest.py'**

**file we will use it with the help of fixture.'''**

***#passing browser/driver to homepage.py***

**homePage = HomePage(*self*.browser)**

***#click shop button***

**homePage.getShopItems().click()**

***#passing the browser/driver to checkoutpage.py***

**checkOutPage = CheckOutPage(*self*.browser)**

***#getting all the products/cards in page***

**products = checkOutPage.getCardTitles()**

**i = -1**

***for* product *in* products:**

**i = i + 1**

**productTitle = product.text**

***if* (productTitle ==  'Blackberry'):**

**checkOutPage.getAddToCart()[i].click()**

***#click checkout button***

**checkOutPage.getCheckoutButton().click()**

***#checkout button of next page***

**checkOutPage.getCheckoutButtonNext().click()**

***#passing the browser/driver to confirmpage.py***

**confirmPage = ConfirmPage(*self*.browser)**

***#input box - entering "ind"***

**confirmPage.getInputBox().send\_keys('ind')**

***#wait till the link text India is present and then click on it***

**confirmPage.getWaitForInd()**

***#select india***

**confirmPage.getSelectInd().click()**

***#click on t&c checkbox***

**confirmPage.getCheckBox().click()**

***#click on purchase button***

**confirmPage.getPurchaseButton().click()**

***#get the success msg printed on webpage verify it if it's matching***

**success\_text = confirmPage.getSuccessMsg().text**

***assert* "Success! Thank you!" *in* success\_text**

***#taking a screenshot of the page***

***self*.browser.get\_screenshot\_as\_file('ss.png')**

1. **Smarter way of returning Page Objects from Navigation Methods–**
   1. As we know, after opening the webpage we have to click on shop button and we passed the browser object to **checkoutpage.py** file but we can just simply define that in **homepage.py** file and the code will look like this-

***from* utilities.BaseClass *import* BaseClass**

***from* pageObjects.HomePage *import* HomePage**

***from* pageObjects.CheckoutPage *import* CheckOutPage**

***from* pageObjects.ConfirmPage *import* ConfirmPage**

**class TestOne(BaseClass):**

**def test\_e2e(*self*):**

**'''We defined browser opening inside the 'setup method' in 'conftest.py'**

**file we will use it with the help of fixture.'''**

***#passing browser/driver to homepage.py***

**homePage = HomePage(*self*.browser)**

***#click shop button and return and pass the browser/driver to CheckoutPage.py***

**checkOutPage = homePage.getShopItems()**

***#getting all the products/cards in page***

**products = checkOutPage.getCardTitles()**

**i = -1**

***for* product *in* products:**

**i = i + 1**

**productTitle = product.text**

***if* (productTitle ==  'Blackberry'):**

**checkOutPage.getAddToCart()[i].click()**

***#click checkout button***

**checkOutPage.getCheckoutButton().click()**

***#checkout button of next page***

***#click checkout button on next page and return and pass the browser/driver to***

***# ConfirmPage.py***

**confirmPage = checkOutPage.getCheckoutButtonNext()**

***#input box - entering "ind"***

**confirmPage.getInputBox().send\_keys('ind')**

***#wait till the link text India is present and then click on it***

**confirmPage.getWaitForInd()**

***#select india***

**confirmPage.getSelectInd().click()**

***#click on t&c checkbox***

**confirmPage.getCheckBox().click()**

***#click on purchase button***

**confirmPage.getPurchaseButton().click()**

***#get the success msg printed on webpage verify it if it's matching***

**success\_text = confirmPage.getSuccessMsg().text**

***assert* "Success! Thank you!" *in* success\_text**

***#taking a screenshot of the page***

***self*.browser.get\_screenshot\_as\_file('ss.png')**

* 1. HomePage.py will look like this-

***from* selenium.webdriver.common.by *import* By**

***from* pageObjects.CheckoutPage *import* CheckOutPage**

**class HomePage:**

***#to receive the driver/browser from the test\_e2e.py, we need to define the constructor***

**def \_\_init\_\_(*self*, *browser*):**

***self*.browser = *browser***

***#assigning the selector to shop variable for clicking the shop button***

**shop = (By.CSS\_SELECTOR, "a[href\*=shop]")**

**def getShopItems(*self*):**

***#that \* tells it to take it as webelement, it deserializes it as tuple***

***self*.browser.find\_element(\*HomePage.shop).click()**

***#passing the browser/driver to CheckoutPage.py so we don't to pass it from***

***# the test\_e2e.py file***

**checkOutPage = CheckOutPage(*self*.browser)**

***return* checkOutPage**

* 1. CheckoutPage.py will look like this-

***from* selenium.webdriver.common.by *import* By**

***from* pageObjects.ConfirmPage *import* ConfirmPage**

**class CheckOutPage:**

***#to receive the driver/browser from the test\_e2e.py, we need to define the constructor***

**def \_\_init\_\_(*self*, *browser*):**

***self*.browser = *browser***

***#assigning all the titles of the products/cards to cardTitle variable***

**cardTitle = (By.CSS\_SELECTOR, ".card-title a")**

***#add to cart button***

**addToCart = (By.CSS\_SELECTOR, ".card-footer button")**

***#checkout button***

**checkoutButton = (By.CSS\_SELECTOR, "a[class\*='btn-primary']")**

***#checkout button of next page***

**checkoutButtonNext = (By.CSS\_SELECTOR, "button[class\*='btn-success']")**

**def getCardTitles(*self*):**

***#that \* tells it to take it as webelement, it deserializes it as tuple***

***return* *self*.browser.find\_elements(\*CheckOutPage.cardTitle)**

**def getAddToCart(*self*):**

***return* *self*.browser.find\_elements(\*CheckOutPage.addToCart)**

**def getCheckoutButton(*self*):**

***return* *self*.browser.find\_element(\*CheckOutPage.checkoutButton)**

**def getCheckoutButtonNext(*self*):**

***self*.browser.find\_element(\*CheckOutPage.checkoutButtonNext).click()**

***#passing the browser driver to ConfirmPage.py so we don't have to pass it from***

***# the test\_e2e.py file***

**confirmPage = ConfirmPage(*self*.browser)**

***return* confirmPage**

1. **Creating Selenium Webdriver Custom Utilities in Base Class–**
   1. Now let’s work on webdriver wait, for that we can define it inside the **BaseClass** in a function and make it generic so it can be used for any element verification.
   2. First, we will remove the code from confirmpage.py and it will look like this-

***from* selenium.webdriver.common.by *import* By**

**class ConfirmPage:**

***#to receive the driver/browser from the test\_e2e.py, we need to define the constructor***

**def \_\_init\_\_(*self*, *browser*):**

***self*.browser = *browser***

***#assigning the selector to shop variable for clicking the shop button***

**inputBox = (By.CSS\_SELECTOR, "#country")**

***#type ind and search for it and then click on India option***

**selectInd = (By.LINK\_TEXT, "India")**

***#t&c checkbox***

**checkBox = (By.XPATH, "//div[@class='checkbox checkbox-primary']")**

***#purchase button***

**purchaseButton = (By.CSS\_SELECTOR, "[type='submit']")**

***#get success msg***

**successMsg = (By.CLASS\_NAME, "alert-success")**

**def getInputBox(*self*):**

***#that \* tells it to take it as webelement, it deserializes it as tuple***

***return* *self*.browser.find\_element(\*ConfirmPage.inputBox)**

**def getSelectInd(*self*):**

***return* *self*.browser.find\_element(\*ConfirmPage.selectInd)**

**def getCheckBox(*self*):**

***return* *self*.browser.find\_element(\*ConfirmPage.checkBox)**

**def getPurchaseButton(*self*):**

***return* *self*.browser.find\_element(\*ConfirmPage.purchaseButton)**

**def getSuccessMsg(*self*):**

***return* *self*.browser.find\_element(\*ConfirmPage.successMsg)**

* 1. Then we will make changes, we will use **BaseClass.py** to define the **verifyLinkPresence** method which will check for the link presence. Here’s the code for it-

***from* selenium.webdriver.support.wait *import* WebDriverWait**

***from* selenium.webdriver.support *import* expected\_conditions *as* EC**

***from* selenium.webdriver.common.by *import* By**

***import* pytest**

***#setup method comes from conftest.py***

**@pytest.mark.usefixtures('setup')**

**class BaseClass:**

**def verifyLinkPresence(*self*, *link\_text*):**

**element = WebDriverWait(*self*.browser, 10).until(EC.presence\_of\_element\_located((By.LINK\_TEXT, *link\_text*)))**

* 1. And our **test\_e2e.py** will look like this-

***from* utilities.BaseClass *import* BaseClass**

***from* pageObjects.HomePage *import* HomePage**

**class TestOne(BaseClass):**

**def test\_e2e(*self*):**

**'''We defined browser opening inside the 'setup method' in 'conftest.py'**

**file we will use it with the help of fixture.'''**

***#passing browser/driver to homepage.py***

**homePage = HomePage(*self*.browser)**

***#click shop button and return and pass the browser/driver to CheckoutPage.py***

**checkOutPage = homePage.getShopItems()**

***#getting all the products/cards in page***

**products = checkOutPage.getCardTitles()**

**i = -1**

***for* product *in* products:**

**i = i + 1**

**productTitle = product.text**

***if* (productTitle ==  'Blackberry'):**

**checkOutPage.getAddToCart()[i].click()**

***#click checkout button***

**checkOutPage.getCheckoutButton().click()**

***#checkout button of next page***

***#click checkout button on next page and return and pass the browser/driver to***

***# ConfirmPage.py***

**confirmPage = checkOutPage.getCheckoutButtonNext()**

***#input box - entering "ind"***

**confirmPage.getInputBox().send\_keys('ind')**

***#verify the link presence***

***self*.verifyLinkPresence("India")**

***#select/click india***

**confirmPage.getSelectInd().click()**

***#click on t&c checkbox***

**confirmPage.getCheckBox().click()**

***#click on purchase button***

**confirmPage.getPurchaseButton().click()**

***#get the success msg printed on webpage verify it if it's matching***

**success\_text = confirmPage.getSuccessMsg().text**

***assert* "Success! Thank you!" *in* success\_text**

***#taking a screenshot of the page***

***self*.browser.get\_screenshot\_as\_file('ss.png')**

* 1. \*Let’s create the test for homepage form also in that test we will put the data in the form and then submit the form, for handle the web-elements of homepage we will put all those elements in **HomePage.py** file in **pageObjects** package and then we will create one **test\_homepage.py** call the objects there same as **test\_e2e.py**.
  2. After this the **HomePage.py** will look like this-

***from* selenium.webdriver.common.by *import* By**

***from* pageObjects.CheckoutPage *import* CheckOutPage**

**class HomePage:**

***#to receive the driver/browser from the test\_e2e.py, we need to define the constructor***

**def \_\_init\_\_(*self*, *browser*):**

***self*.browser = *browser***

***#assigning the selector to shop variable for clicking the shop button***

**shop = (By.CSS\_SELECTOR, "a[href\*=shop]")**

***#for testTwo or testHomePage.py***

**name = (By.CSS\_SELECTOR, "[name='name']")**

**email = (By.NAME, "email")**

***#to click on the checkbox***

**check = (By.ID, "exampleCheck1")**

**gender = (By.ID, "exampleFormControlSelect1")**

**submit = (By.XPATH, "//input[@value='Submit']")**

**successMsg = (By.CSS\_SELECTOR, "[class\*='alert-success']")**

**def getShopItems(*self*):**

***#that \* tells it to take it as webelement, it deserializes it as tuple***

***self*.browser.find\_element(\*HomePage.shop).click()**

***#passing the browser/driver to CheckoutPage.py so we don't to pass it from***

***# the test\_e2e.py file***

**checkOutPage = CheckOutPage(*self*.browser)**

***return* checkOutPage**

**def getName(*self*):**

***return* *self*.browser.find\_element(\*HomePage.name)**

**def getEmail(*self*):**

***return* *self*.browser.find\_element(\*HomePage.email)**

**def getCheck(*self*):**

***return* *self*.browser.find\_element(\*HomePage.check)**

**def getGender(*self*):**

***return* *self*.browser.find\_element(\*HomePage.gender)**

**def getSubmit(*self*):**

***return* *self*.browser.find\_element(\*HomePage.submit)**

**def getSuccessMsg(*self*):**

***return* *self*.browser.find\_element(\*HomePage.successMsg)**

* 1. And **test\_homepage.py** will look like this-

***from* utilities.BaseClass *import* BaseClass**

***from* pageObjects.HomePage *import* HomePage**

**class TestHomepage(BaseClass):**

**def test\_homepageForm(*self*):**

***#passing browser/driver to homepage.py***

**homepage = HomePage(*self*.browser)**

***#fill name in name field in the homepage form***

**homepage.getName().send\_keys("Sandeep")**

***#fill email in email field in the homepage form***

**homepage.getEmail().send\_keys("sandeeppatel8878@gmail.com")**

***#click on checkbox***

**homepage.getCheck().click()**

***#we are getting this from the Baseclass.py***

***self*.selectOptionByText(homepage.getGender(), "Male")**

***#click on the submit button***

**homepage.getSubmit().click()**

***#checking if success text is present in homepage or not after we are done***

***# with the all the operations***

**alertText = homepage.getSuccessMsg().text**

***assert*("Success" *in* alertText)**

1. **Parameterizing Web-driver test with Multiple DataSets–**
   1. Let’s say if we want to run this test with multiple datasets and each time it should pick different dataset, and to do that we can send that data as tuple in parameter. Here’s the example of the code-

***from* utilities.BaseClass *import* BaseClass**

***from* pageObjects.HomePage *import* HomePage**

***import* pytest**

**class TestHomepage(BaseClass):**

**def test\_homepageForm(*self*, *getData*):**

***#passing browser/driver to homepage.py***

**homepage = HomePage(*self*.browser)**

***#fill name in name field in the homepage form***

**homepage.getName().send\_keys(*getData*[0])**

***#fill email in email field in the homepage form***

**homepage.getEmail().send\_keys(*getData*[1])**

***#click on checkbox***

**homepage.getCheck().click()**

***#we are getting this from the Baseclass.py***

***self*.selectOptionByText(homepage.getGender(), *getData*[2])**

***#click on the submit button***

**homepage.getSubmit().click()**

***#checking if success text is present in homepage or not after we are done***

***# with the all the operations***

**alertText = homepage.getSuccessMsg().text**

***assert*("Success" *in* alertText)**

***self*.browser.refresh()**

**@pytest.fixture(*params*=[("sandeep", "test@gmail.com", "Male"), ("sweety", "test1@gmail.com", "Female")])**

**def getData(*self*, *request*):**

***return* *request*.param**

* 1. Only issue with this is that we can’t pass our data like this -> getData[0]. We can overcome this by using passing data as dictionary, code will look like this-

***from* utilities.BaseClass *import* BaseClass**

***from* pageObjects.HomePage *import* HomePage**

***import* pytest**

**class TestHomepage(BaseClass):**

**def test\_homepageForm(*self*, *getData*):**

***#passing browser/driver to homepage.py***

**homepage = HomePage(*self*.browser)**

***#fill name in name field in the homepage form***

**homepage.getName().send\_keys(*getData*["firstname"])**

***#fill email in email field in the homepage form***

**homepage.getEmail().send\_keys(*getData*["email"])**

***#click on checkbox***

**homepage.getCheck().click()**

***#we are getting this from the Baseclass.py***

***self*.selectOptionByText(homepage.getGender(), *getData*["gender"])**

***#click on the submit button***

**homepage.getSubmit().click()**

***#checking if success text is present in homepage or not after we are done***

***# with the all the operations***

**alertText = homepage.getSuccessMsg().text**

***assert*("Success" *in* alertText)**

***self*.browser.refresh()**

**@pytest.fixture(*params*=[{"firstname":"sandeep", "email":"test@gmail.com", "gender":"Male"}, {"firstname":"sweety", "email":"test1@gmail.com", "gender":"Female"}])**

**def getData(*self*, *request*):**

***return* *request*.param**

1. **Organizing Data from separate Data Files and injecting into fixture at run time –**
   1. This is the continuation of step 8, in this we will get the passing argument as dictionary from the **testData** package’s **HomePageData.py** file, and then we will access this dictionary data using key value, the code will look like this-

***from* utilities.BaseClass *import* BaseClass**

***from* pageObjects.HomePage *import* HomePage**

***from* testData.HomePageData *import* HomePageData**

***import* pytest**

**class TestHomepage(BaseClass):**

**def test\_homepageForm(*self*, *getData*):**

***#passing browser/driver to homepage.py***

**homepage = HomePage(*self*.browser)**

***#fill name in name field in the homepage form***

**homepage.getName().send\_keys(*getData*["firstname"])**

***#fill email in email field in the homepage form***

**homepage.getEmail().send\_keys(*getData*["email"])**

***#click on checkbox***

**homepage.getCheck().click()**

***#we are getting this from the Baseclass.py***

***self*.selectOptionByText(homepage.getGender(), *getData*["gender"])**

***#click on the submit button***

**homepage.getSubmit().click()**

***#checking if success text is present in homepage or not after we are done***

***# with the all the operations***

**alertText = homepage.getSuccessMsg().text**

***assert*("Success" *in* alertText)**

***self*.browser.refresh()**

**@pytest.fixture(*params*=HomePageData.test\_homepageData)**

**def getData(*self*, *request*):**

***return* *request*.param**

* 1. And the HomePageData.py will look like this-

**class HomePageData:**

**test\_homepageData = [{"firstname":"sandeep", "email":"test@gmail.com", "gender":"Male"}, {"firstname":"sweety", "email":"test1@gmail.com", "gender":"Female"}]**

1. **Implementing Logging feature to Web-driver tests–**
   1. To implement the logging feature first we will define the logger method in **BaseClass.py** and then use that method in whichever test we want. Code of baseclass will look like this-

***from* selenium.webdriver.support.wait *import* WebDriverWait**

***from* selenium.webdriver.support *import* expected\_conditions *as* EC**

***from* selenium.webdriver.common.by *import* By**

***from* selenium.webdriver.support.select *import* Select**

***import* inspect**

***import* logging**

***import* pytest**

***#setup method comes from conftest.py***

**@pytest.mark.usefixtures('setup')**

**class BaseClass:**

**def getLogger(*self*):**

**loggerName = inspect.stack()[1][3]**

**logger = logging.getLogger(loggerName)**

**fileHandler = logging.FileHandler('logfile.log')**

**formatter = logging.Formatter("%(asctime)s :%(levelname)s : %(name)s :%(message)s")**

**fileHandler.setFormatter(formatter)**

**logger.addHandler(fileHandler)  *# filehandler object***

**logger.setLevel(logging.DEBUG)**

***return* logger**

***#to verify is the certain element is present or not***

**def verifyLinkPresence(*self*, *link\_text*):**

**element = WebDriverWait(*self*.browser, 10).until(EC.presence\_of\_element\_located((By.LINK\_TEXT, *link\_text*)))**

**def selectOptionByText(*self*, *locator*, *text*):**

**sel = Select(*locator*)**

**sel.select\_by\_visible\_text(*text*)**

* 1. Then after using the logger method in **test\_e2e.py** it will look like this-

***from* utilities.BaseClass *import* BaseClass**

***from* pageObjects.HomePage *import* HomePage**

**class TestOne(BaseClass):**

**def test\_e2e(*self*):**

**'''We defined browser opening inside the 'setup method' in 'conftest.py'**

**file we will use it with the help of fixture.'''**

***#logger object***

**log = *self*.getLogger()**

***#passing browser/driver to homepage.py***

**homePage = HomePage(*self*.browser)**

***#click shop button and return and pass the browser/driver to CheckoutPage.py***

**checkOutPage = homePage.getShopItems()**

***#logging into log file***

**log.info("getting all the card titles")**

***#getting all the products/cards in page***

**products = checkOutPage.getCardTitles()**

**i = -1**

***for* product *in* products:**

**i = i + 1**

**productTitle = product.text**

***#printing title of the product in log file***

**log.info(productTitle)**

***if* (productTitle ==  'Blackberry'):**

**checkOutPage.getAddToCart()[i].click()**

***#click checkout button***

**checkOutPage.getCheckoutButton().click()**

***#checkout button of next page***

***#click checkout button on next page and return and pass the browser/driver to***

***# ConfirmPage.py***

**confirmPage = checkOutPage.getCheckoutButtonNext()**

***#logging into log file***

**log.info("Entering country name as ind")**

***#input box - entering "ind"***

**confirmPage.getInputBox().send\_keys('ind')**

***#verify the link presence***

***self*.verifyLinkPresence("India")**

***#select/click india***

**confirmPage.getSelectInd().click()**

***#click on t&c checkbox***

**confirmPage.getCheckBox().click()**

***#click on purchase button***

**confirmPage.getPurchaseButton().click()**

***#get the success msg printed on webpage verify it if it's matching***

**success\_text = confirmPage.getSuccessMsg().text**

***#logging the success msg into log file***

**log.info("Text received from application is: " + success\_text)**

***assert* "Success! Thank you!" *in* success\_text**

***#taking a screenshot of the page***

***self*.browser.get\_screenshot\_as\_file('ss.png')**

* 1. Then after using the logger method in **test\_homepage.py** it will look like this-

***from* utilities.BaseClass *import* BaseClass**

***from* pageObjects.HomePage *import* HomePage**

***from* testData.HomePageData *import* HomePageData**

***import* pytest**

**class TestHomepage(BaseClass):**

**def test\_homepageForm(*self*, *getData*):**

***#logger object***

**log = *self*.getLogger()**

***#passing browser/driver to homepage.py***

**homepage = HomePage(*self*.browser)**

***#logging firstname into log file***

**log.info("Firstname: " + *getData*["firstname"])**

***#fill name in name field in the homepage form***

**homepage.getName().send\_keys(*getData*["firstname"])**

***#fill email in email field in the homepage form***

**homepage.getEmail().send\_keys(*getData*["email"])**

***#click on checkbox***

**homepage.getCheck().click()**

***#we are getting this from the Baseclass.py***

***self*.selectOptionByText(homepage.getGender(), *getData*["gender"])**

***#click on the submit button***

**homepage.getSubmit().click()**

***#checking if success text is present in homepage or not after we are done***

***# with the all the operations***

**alertText = homepage.getSuccessMsg().text**

***assert*("Success" *in* alertText)**

***self*.browser.refresh()**

**@pytest.fixture(*params*=HomePageData.test\_homepageData)**

**def getData(*self*, *request*):**

***return* *request*.param**

1. **Test Execution HTML Reporting or Generating HTML Reports with logging integration–**
   1. First of all, we need to download and install the pytest-html plugin, to do that use command **pip install pytest-html** and after that we can generate the reports while running our code, we simply have to write **–html=report.html** before the **py.test** and it will generate the report for us.
2. **Automatic Screenshot capture on Test Failures–**
   1. For this we have to put the code in **conftest.py** file and we are using pre-made code of the tutor after using that out code will look like this-

***from* selenium *import* webdriver**

***from* selenium.webdriver.chrome.service *import* Service**

***import* pytest**

**browser=None**

***#add a command line option of '--browser name\_of\_browser'***

**def pytest\_addoption(*parser*):**

***parser*.addoption(**

**"--browser", *action*='store', *default*='chrome'**

**)**

**@pytest.fixture(*scope*='class')**

**def setup(*request*):**

**global browser**

***#this is how we can get the value of '--browser' key***

**browser\_name = *request*.config.getoption("browser")**

***if* browser\_name == "chrome":**

**browser = webdriver.Chrome(*service*=Service("C:\\Users\\sande\\Desktop\\Programs\\chromedriver.exe"))**

***elif* browser\_name == "firefox":**

**browser = webdriver.Firefox(*service*=Service("C:\\Users\\sande\\Desktop\\Programs\\geckodriver.exe"))**

***elif* browser\_name == 'edge':**

**browser = webdriver.Edge(*service*=Service("C:\\Users\\sande\\Desktop\\Programs\\msedgedriver.exe"))**

**browser.get('https://rahulshettyacademy.com/angularpractice/')**

**browser.maximize\_window()**

***#whichever class request browser pass this browser***

***request*.cls.browser = browser**

***yield***

**browser.close()**

***#capture the screenshot whenever the test fails***

**@pytest.mark.hookwrapper**

**def pytest\_runtest\_makereport(*item*):**

**"""**

**Extends the PyTest Plugin to take and embed screenshot in html report, whenever test fails.**

**:param item:**

**"""**

**pytest\_html = *item*.config.pluginmanager.getplugin('html')**

**outcome = *yield***

**report = outcome.get\_result()**

**extra = getattr(report, 'extra', [])**

***if* report.when == 'call' *or* report.when == "setup":**

**xfail = hasattr(report, 'wasxfail')**

***if* (report.skipped *and* xfail) *or* (report.failed *and* *not* xfail):**

**file\_name = report.nodeid.replace("::", "\_") + ".png"**

**\_capture\_screenshot(file\_name)**

***if* file\_name:**

**html = '<div><img src="%s" alt="screenshot" style="width:304px;height:228px;" ' \**

**'onclick="window.open(this.src)" align="right"/></div>' % file\_name**

**extra.append(pytest\_html.extras.html(html))**

**report.extra = extra**

**def \_capture\_screenshot(*name*):**

**browser.get\_screenshot\_as\_file(*name*)**

**Jenkins**

1. **Integrating Framework to Jenkins and setting up various Job Parameters–**
   1. First of all, we will download and install Jenkins, to do that we will google Jenkins download and then we will download the jenkins.war file.
   2. After that we have to put that **jenkins.war** file to whichever folder we like and then we will open command prompt there and we have to use command **“java -jar jenkins.war”** to open and setup it for the first time and after the second time we will just fire the command and run that jenkins.war.
   3. If we want to run that server in a specific port, we can do that by modifying the command and it will look like this -> **“java -jar jenkins.war -httpPort=8080”** port value can be anything, whatever you like.
   4. Then visit [**http://localhost:8080/**](http://localhost:8080/)and click on **new item** on left side then put the name **SeleniumPythonTesting** then first option **Freestyle Project.**
   5. Then we have to set it up with our ‘end-to-end selenium framework’ so after the freestyle project we will get the description screen and, on that screen, we have to setup some things.
   6. To do that we will copy the path of your **‘python-selenium-framework’** folder and then click **advanced** option in general tab in the page of jenkins localhost.
   7. Click on **custom workspace** there and put the path of python-selenium-framework (C:\Users\sande\Desktop\Programs\Projects\python-selenium-framework) there.
   8. Then click on **build** and select the **Execute windows batch command option** and then put the commands first one will be for changing directory to test folder and second one for running/executing our tests, it will look like this-

**cd test**

**py.test --browser chrome --html=$WORKSPACE/reports/reports.html**

* 1. Here workspace means the python-selenium-framework directory, put these commands in the box and then click on save.
  2. Now open the project and click on build now and it will build for us then we can see the logs in console for that we have to click on **latest build** then **console output**.
  3. And to open the reports.html we have to go to **project** then **workspace** then **reports** then **reports.html**.
  4. Now let’s make it like this that we don’t have to provide the browser name in our command for that we will click on **This project is parameterized** and from the dropdown select the **choice parameter** on the name box fill **browserName** and in choices fill **firefox** then enter to change the line and **chrome** then enter for line change then **edge** and then click on save.
  5. Now we can see we have the option to build with parameters click on that and then from the dropdown we can select the browser.
  6. We can also create Junit reports and to do that we need to open **configure** scroll down and then on **build** section we have to change the commands to these-

**cd test**

**py.test --browser "%browserName%" --html=$WORKSPACE/reports/reports.html -v --junitxml="result.xml"**

* 1. Then scroll down to **post build actions** and on **test report xmls** box write **test/result.xml** and save and after this whenever we will use the build option it will generate the **result.xml** for us and we can access that file from **workspace** section.