Page object module (POM Class):-

* POM is one of design pattern which help proper way of writing automation test script.
* POM Follows Encapsulation principal while designing class

Encapsulation:

1. Data member should be declared globally and whose access specifer should be private.
2. Initialize within constructor whose access scope should be public
3. Utilize data members inside methods with access scope public.

* POM class should not contain main method.
* To run POM class there is need of another test class which constist of main method.
* Total number data members in POM class is depends on number of elements which needs to be handle on webpage.

**Page Factory:**

Page factory is class which is having static method like initElemet() method.

* To initialize data member we need to call initElemet method within the class inside the constructor.
* Ex: PageFactory.*initElements*(dri, **this**);
* InitElement method will initialize data member by identifying each component present on web page by using @FindBy annotation which takes input as locator type.
* Ex: @FindBy(xpath="x-path expression") **private** WebElement dataMember;

Working:

When we call POM class initElements method will convert all data members @FindBy annotation to findElemets this is called as basic or early initialization.

Ex:

@FindBy(xpath="x-path expression") **private** WebElement dataMember;

to

WebElement dataMember = driver.findElement(By.xpath(“x-path expression”));

* Instead of perform action on element directly, element on web page is identified and then action will be performed this is known as late initialization.

Framework Type:

2. Keyword driven:

* In keyword driven framework all actions are control through separate sheet apart from test data.
* Actions are methods which needs to be handle through another sheet.
* Similarity between data driven and keyword driven is that it depends on external data sheet

1. Hybrid Framework:

This is combination of both framework

**Data Driven Advantages:**

1. We can test functionality with multiple test data.
2. As test data is separate from actual automation script, any one can change test data who don’t have automation knowledge.
3. Any change in automation code will not impact on Test data.

Disadvantages:

1. Skill person is required to design and work.
2. For large amount of data care should be taken to put data in proper place.
3. Documentation is needed to learn the framework.

**package** com.test.signup;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** org.apache.poi.EncryptedDocumentException;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.ss.usermodel.WorkbookFactory;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** com.vel.config.Configurations;

**public** **class** TestCase02 {

**public** **static** **void** main(String[] args) **throws** InterruptedException, EncryptedDocumentException, IOException {

System.*setProperty*("webdriver.chrome.driver", Configurations.***driverPath***);

WebDriver driver = **new** ChromeDriver();

driver.get(Configurations.*appURL*);

driver.manage().window().maximize();

SignUpPOM signup = **new** SignUpPOM(driver);

Utility util = **new** Utility();

//TC01

signup.enterFirstName(util.readExcelSheet(1, 1));

signup.enterLastName(util.readExcelSheet(2, 1));

signup.enterEmail(util.readExcelSheet(3, 1));

signup.selectGender();

signup.selectMonth(util.readExcelSheet(4, 1));

Thread.*sleep*(4000);

driver.get(Configurations.*appURL*);

//TC02

signup.enterFirstName(util.readExcelSheet(1, 2));

signup.enterLastName(util.readExcelSheet(2, 2));

signup.enterEmail(util.readExcelSheet(3, 2));

signup.selectGender();

signup.selectMonth(util.readExcelSheet(4, 2));

Thread.*sleep*(4000);

//TC03

driver.get(Configurations.*appURL*);

signup.enterFirstName(util.readExcelSheet(1, 3));

signup.enterLastName(util.readExcelSheet(2, 3));

signup.enterEmail(util.readExcelSheet(3, 3));

signup.selectGender();

signup.selectMonth(util.readExcelSheet(4, 3));

}

}

--

**package** com.test.signup;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**import** org.apache.poi.EncryptedDocumentException;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.ss.usermodel.WorkbookFactory;

**import** com.vel.config.Configurations;

**public** **class** Utility {

**public** String readExcelSheet(**int** row, **int** col) **throws** EncryptedDocumentException, IOException {

FileInputStream file = **new** FileInputStream(Configurations.***excelSheetPath***);

Sheet sh = WorkbookFactory.*create*(file).getSheet("TC01");

String data = sh.getRow(row).getCell(col).getStringCellValue();

**return** data;

}

}

**package** com.test.signup;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.support.FindBy;

**import** org.openqa.selenium.support.PageFactory;

**import** org.openqa.selenium.support.ui.Select;

**public** **class** SignUpPOM {

//Declaration

@FindBy(xpath="//input[@name='firstname']") **private** WebElement userName;

@FindBy(xpath="//input[@name='lastname']") **private** WebElement lastName;

@FindBy(xpath="//input[@name='reg\_email\_\_']") **private** WebElement email;

@FindBy(xpath="(//input[@type='radio'])[2]") **private** WebElement gender;

@FindBy(xpath="//select[@name='birthday\_month']") **private** WebElement month;

//Initilization

SignUpPOM(WebDriver dri)

{

PageFactory.*initElements*(dri, **this**);

}

//Usage

**public** **void** enterFirstName(String name) {

userName.sendKeys(name);

}

**public** **void** enterLastName(String lName) {

lastName.sendKeys(lName);

}

**public** **void** enterEmail(String email1) {

email.sendKeys(email1);

}

**public** **void** selectGender() {

gender.click();

}

**public** **void** selectMonth(String value) {

Select s1 = **new** Select(month);

s1.selectByVisibleText(value);

}

}

Tomorrow and Sunday 9 to 11.

1St march 9 to 11