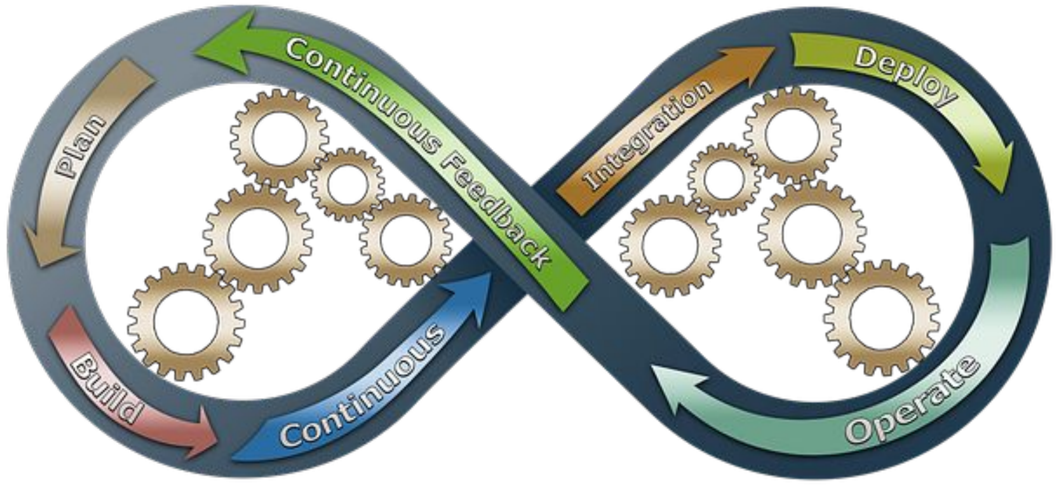


Session 4.10

FrameWorks

AN INITIATIVE BY

UNICAL ACADEMY

Session 4.10

Introduction

UNICAL ACADEMY

Let's go!!!

FrameWorks

Selenium Framework

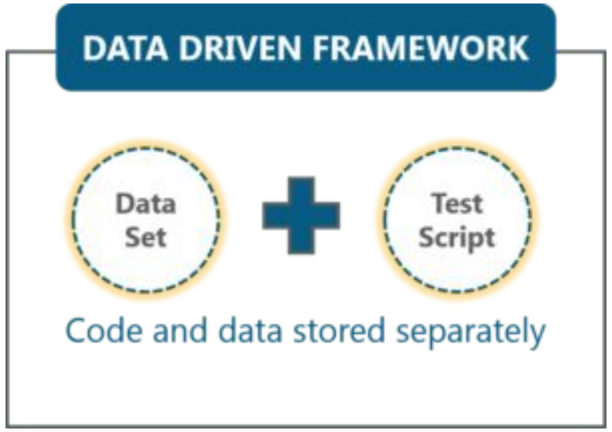
- Selenium framework is a code structure for making code maintenance simpler, and code readability better. A framework involves breaking the entire code into smaller pieces of code, which test a particular functionality.
- The code is structured such that, the “data set” is separated from the actual “test case” which will test the functionality of the web application.
- There are a number of frameworks out there, but 3 commonly used Selenium framework (s) are:

❖ Data Driven framework
❖ Keyword Driven framework
❖ Hybrid framework

UNICAL ACADEMY

Data Driven Framework

- A Data Driven framework in Selenium is the technique of separating the “data set” from the actual “test case” (code).
- This framework completely depends on the input test data. The test data is fed from external sources such as an excel file, .CSV file or any database.



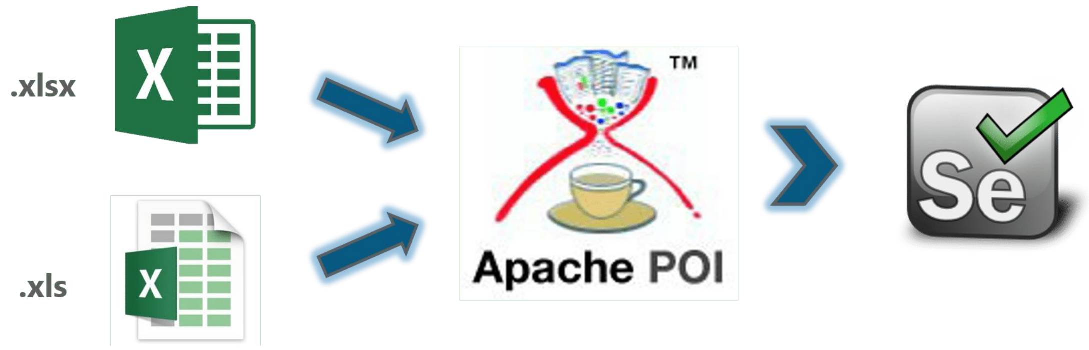
- Since the test case is separated from the data set, we can easily modify the test case of a particular functionality without making wholesale changes to your code.
- For example, if you want to modify the code for login functionality, then you can modify just that instead of having to also modify any other dependent portion in the same code.

UNICAL ACADEMY

Data Driven Framework

Using Apache POI With Selenium WebDriver:

- WebDriver does not directly support reading of excel files. Hence we use **Apache POI** for reading/ writing to any Microsoft office document.
- You can download Apache POI (set of JAR files) zip file or tar file as per your requirement and place them along with the set of Selenium JARs.

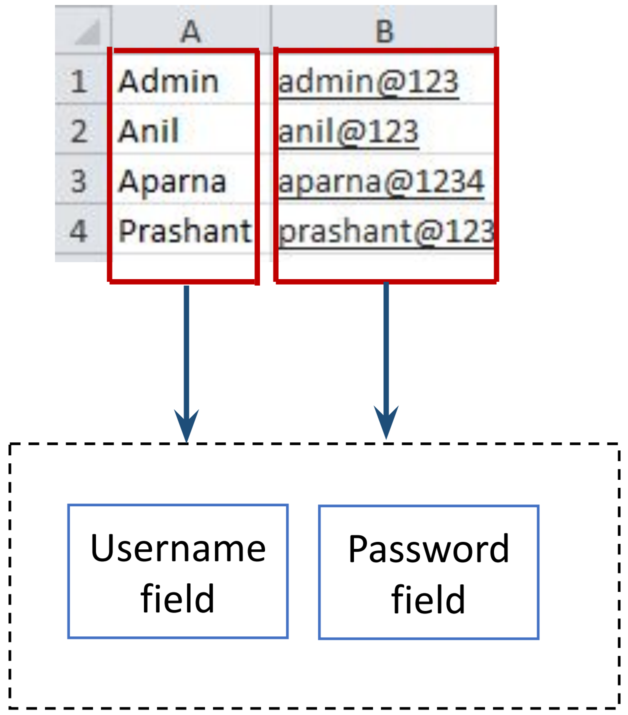


- The co-ordination between the main code and data set will be taken care by *TestNG Data Providers,* which is a library that comes as a part of the Apache POI JAR files.

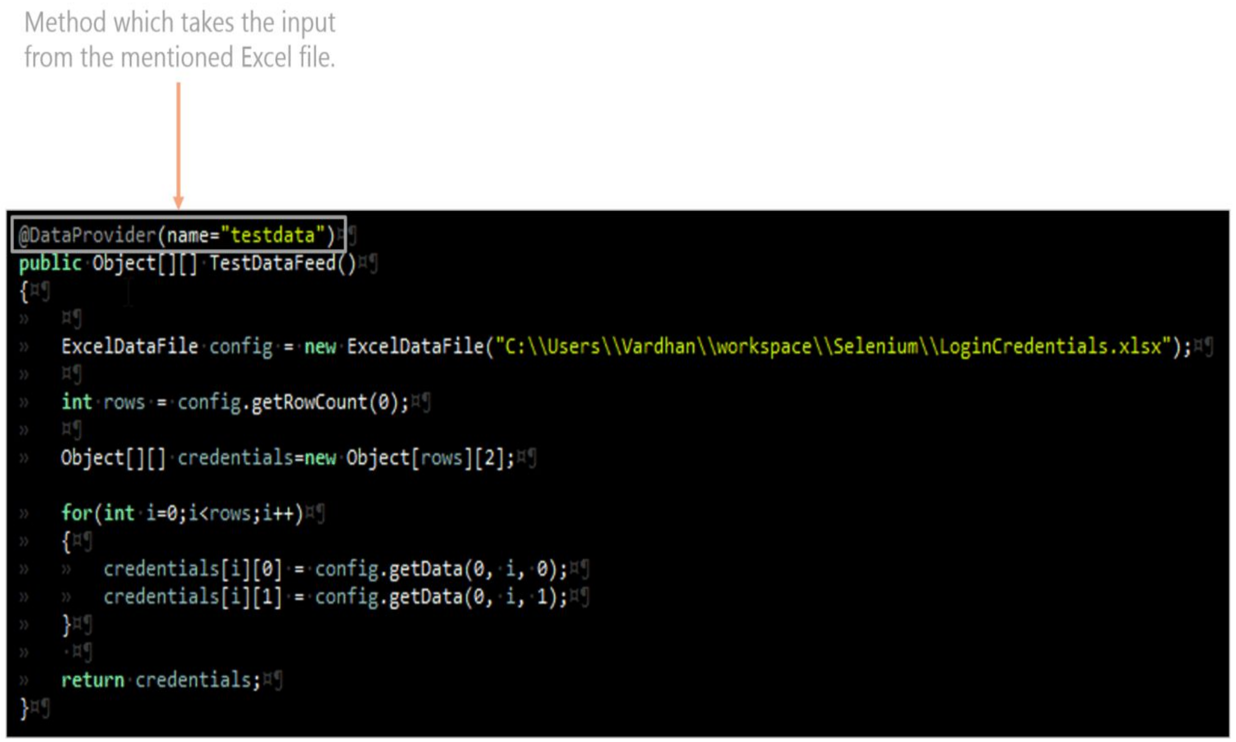
Data Driven Framework

Example:

- created an excel file called “LoginCredentials” in which the usernames and passwords have been stored in different columns.



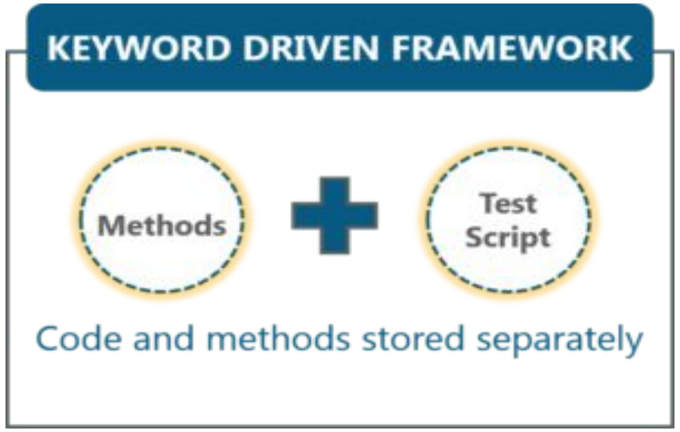
	A	B
1	Admin	admin@123
2	Anil	anil@123
3	Aparna	aparna@1234
4	Prashant	prashant@123



```
@DataProvider(name="testdata")
public Object[][] TestDataFeed()
{

    ExcelDataFile config = new ExcelDataFile("C:\\Users\\Vardhan\\workspace\\Selenium\\LoginCredentials.xlsx");

    int rows = config.getRowCount(0);

    Object[][] credentials=new Object[rows][2];

    for(int i=0;i<rows;i++)
    {
        credentials[i][0] = config.getData(0, i, 0);
        credentials[i][1] = config.getData(0, i, 1);
    }

    return credentials;
}
```

Keyword Driven Framework

- Keyword Driven framework is a technique in which all the operations & instructions to be performed are written separately from the actual test case.
- The similarity it has with Data Driven framework is that, the operations to be performed is again stored in an external file like Excel sheet.



- The benefit with Keyword Driven framework is that you can easily control the functionalities you want to test. You can specify the methods which test the functionality of the application in the excel file.

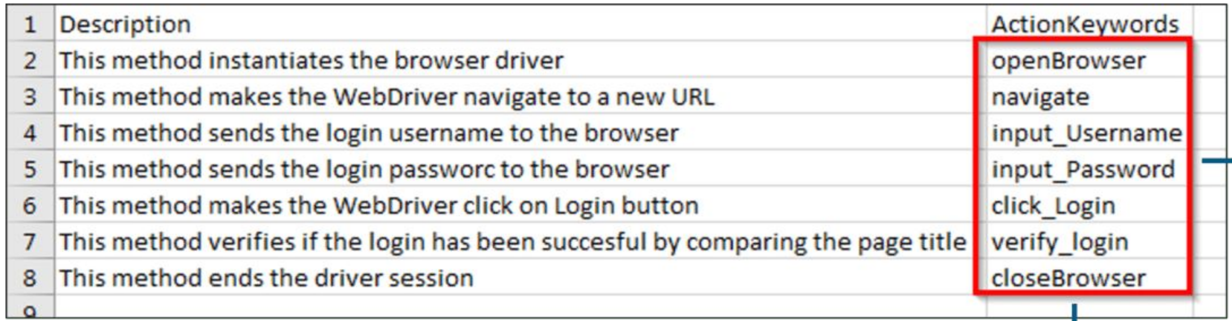
UNICAL ACADEMY

Example:

- For logging into the web application, we can write multiple methods in the main test case, in which each test case will test certain functionality.
- For instantiating the browser driver there could be one method, for finding the username & password fields, there could be methods, for navigating to a web page there could be another method, etc.
- The different functionalities which need to be tested are present in separate methods waiting to be called. Now, these methods will be called from another Class, based on the presence of the method name in the excel file.

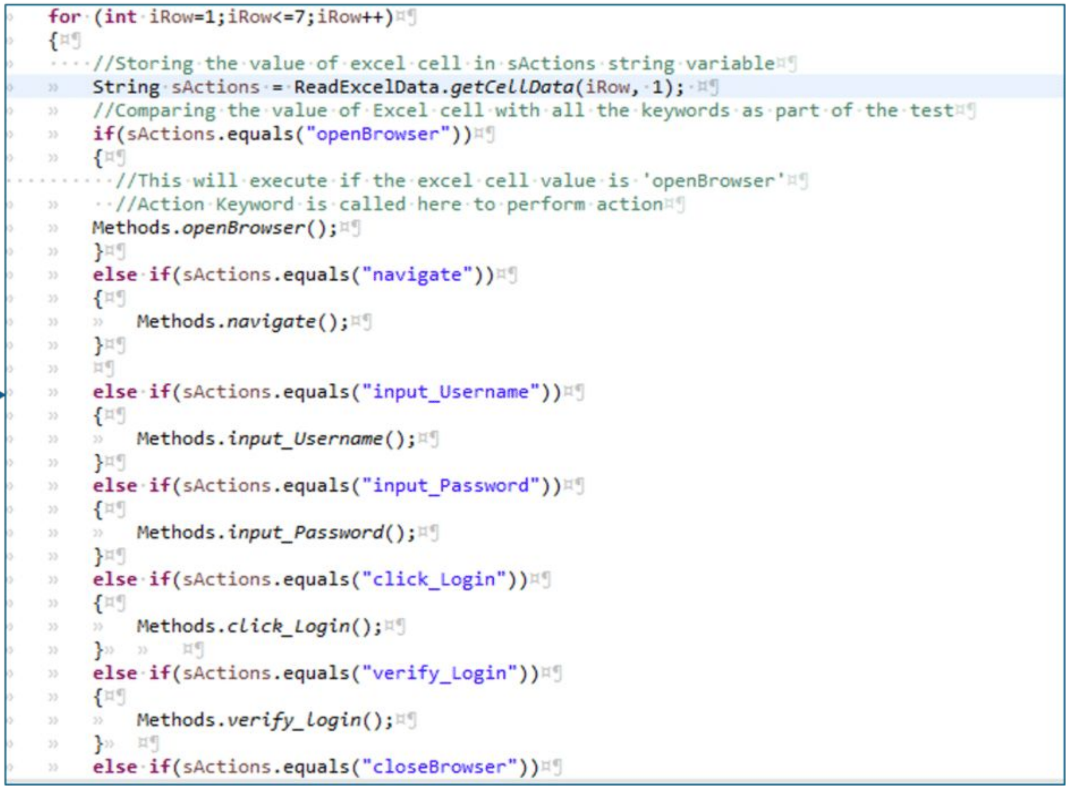
Session 4.10

UNICAL ACADEMY



1	Description	ActionKeywords
2	This method instantiates the browser driver	openBrowser
3	This method makes the WebDriver navigate to a new URL	navigate
4	This method sends the login username to the browser	input_Username
5	This method sends the login passworc to the browser	input_Password
6	This method makes the WebDriver click on Login button	click_Login
7	This method verifies if the login has been succesful by comparing the page title	verify_login
8	This method ends the driver session	closeBrowser

Method names present here will be executed as part of the test case.

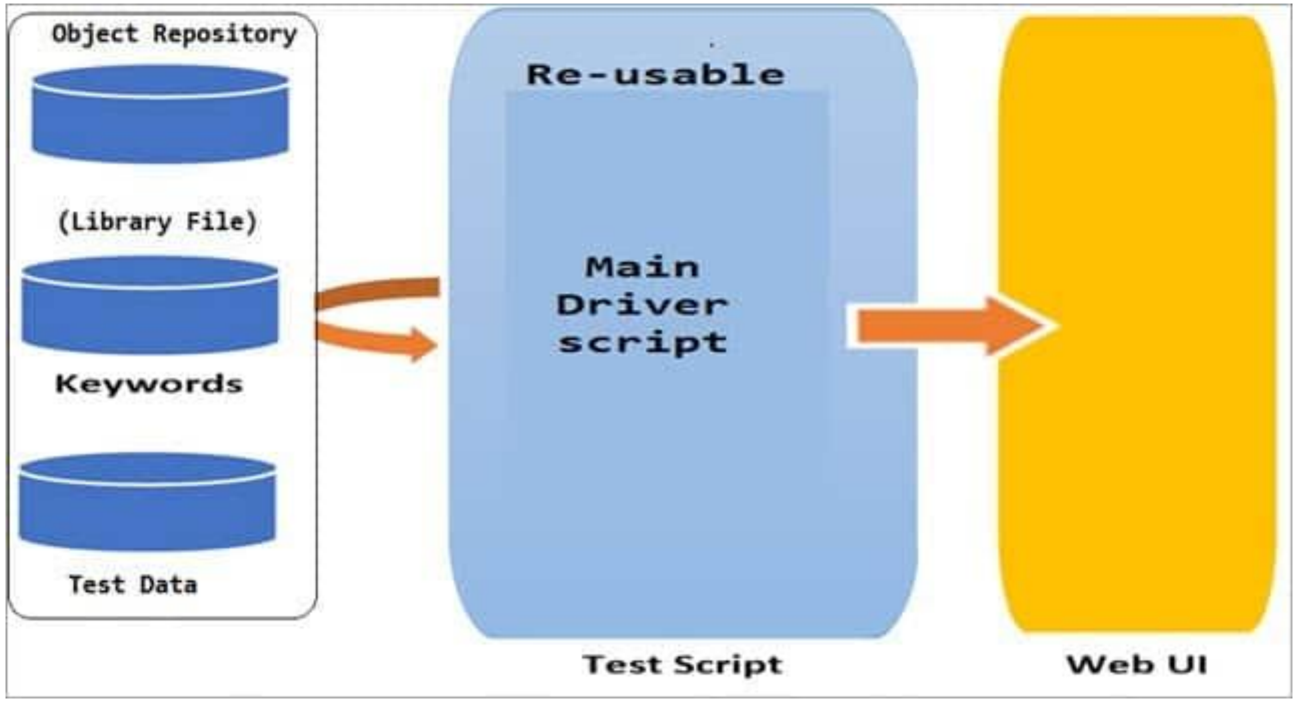


```
for (int iRow=1;iRow<=7;iRow++)
{
    //Storing the value of excel cell in sActions string variable
    String sActions = ReadExcelData.getCellData(iRow, 1);
    //Comparing the value of Excel cell with all the keywords as part of the test
    if(sActions.equals("openBrowser"))
    {
      //This will execute if the excel cell value is 'openBrowser'
      //Action Keyword is called here to perform action
    Methods.openBrowser();
    }
    else if(sActions.equals("navigate"))
    {
        Methods.navigate();
    }

    else if(sActions.equals("input_Username"))
    {
        Methods.input_Username();
    }
    else if(sActions.equals("input_Password"))
    {
        Methods.input_Password();
    }
    else if(sActions.equals("click_Login"))
    {
        Methods.click_Login();
    }
    else if(sActions.equals("verify_Login"))
    {
        Methods.verify_login();
    }
    else if(sActions.equals("closeBrowser"))
```

Hybrid framework

UNICAL ACADEMY

- we can build a Hybrid framework by storing the methods to execute in an excel file (keyword driven approach) and passing these method names to the *Java Reflection Class* (data driven approach) instead of creating an **If/Else** loop in the "DriverScript" class.



Hybrid Framework

UNICAL ACADEMY

Components of the Hybrid Framework:

1) Function Library

- User-defined methods are created for each user action. In other words, Keywords are created in the library file.

Example:

Let us take an instance to automate the below test cases.

Test Case No	Description	Test Steps	Expected Result
1	Verify Unical logo present	1. Enter URL - http://183.82.4.93:5887/moss/	Unical logo should be displayed in home page
2	Verify valid login	1. Enter URL - http://183.82.4.93:5887/moss/ 2. Click on 'login' link 3. Enter valid Username 4. Click on continue 5. Enter Valid Password 6. Click on LogInButton	User Icon should be present in Homepage
3	Invalid login	1. Enter URL - http://183.82.4.93:5887/moss/ 2. Click on 'login' link 3. Enter invalid username and password 4. Click on continue	This error message should contain 'Invalid Username or Password '

UNICAL ACADEMY

Hybrid Framework

- **TC 01:** Verify Unical logo present- the user actions will be: Enter URL
- **TC 02:** Verify Valid LogIn- the user actions are Enter URL, Click, TypeIn
- **TC03:** Verify Invalid Login- the user actions are Enter URL, Click, TypeIn

2) Excel Sheet To Store Keywords

- Keywords that are created in the library file are stored in an excel sheet with its description for anyone who uses this framework to understand.

3) Design Test Case Template

- As per the Hybrid Framework, both Test data and Keywords should be externalized. **So, a template is created accordingly. For example:**

Hybrid Framework

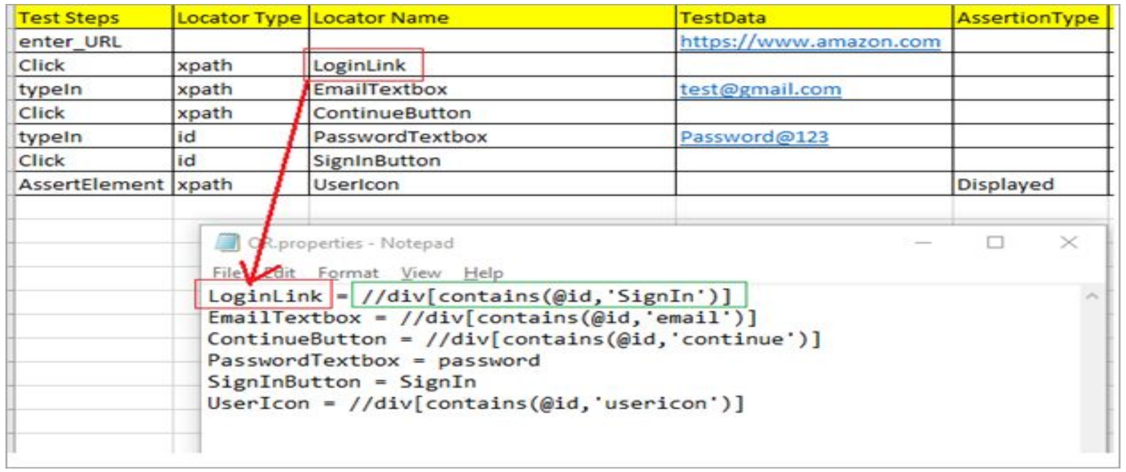
For Test case 2 – Verify valid LogIn

Test Steps	Locator Type	Locator Value	TestData	AssertionType	ExpectedValue
enter_URL			http://183.82.4.93:5887/moss/		
Click	xpath	//div[contains(@id,'LogIn')]			
typeIn	xpath	//div[contains(@id,'UserID')]	Admin		
typeIn	id	password	Aparna@1234		
Click	id	LogIn			

4) Object Repository For Elements

- A separate Repository is maintained for all elements on the webpage. Each WebElement is referred with a name followed by its value in an Object Repository.

UNICAL ACADEMY

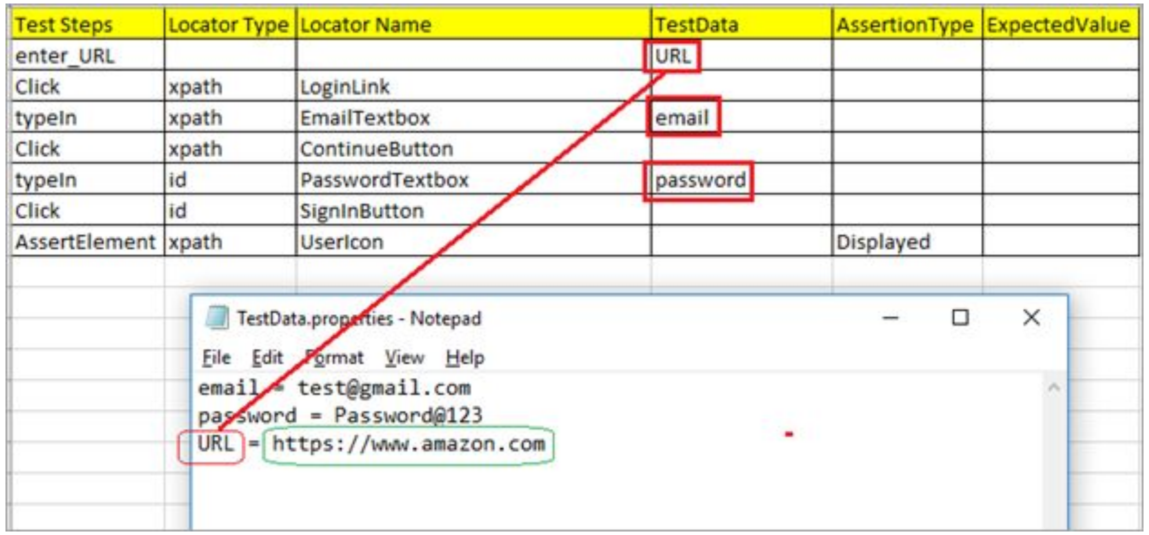


Test Steps	Locator Type	Locator Name	TestData	AssertionType
enter_URL			https://www.amazon.com	
Click	xpath	LoginLink		
typeIn	xpath	EmailTextbox	test@gmail.com	
Click	xpath	ContinueButton		
typeIn	id	PasswordTextbox	Password@123	
Click	id	SignInButton		
AssertElement	xpath	UserIcon		Displayed

```
LoginLink = //div[contains(@id,'SignIn')]
EmailTextbox = //div[contains(@id,'email')]
ContinueButton = //div[contains(@id,'continue')]
PasswordTextbox = password
SignInButton = SignIn
UserIcon = //div[contains(@id,'usericon')]
```

Object Repository For Test Data In Test Cases:

- Externalizing Test Data from Test case Template:

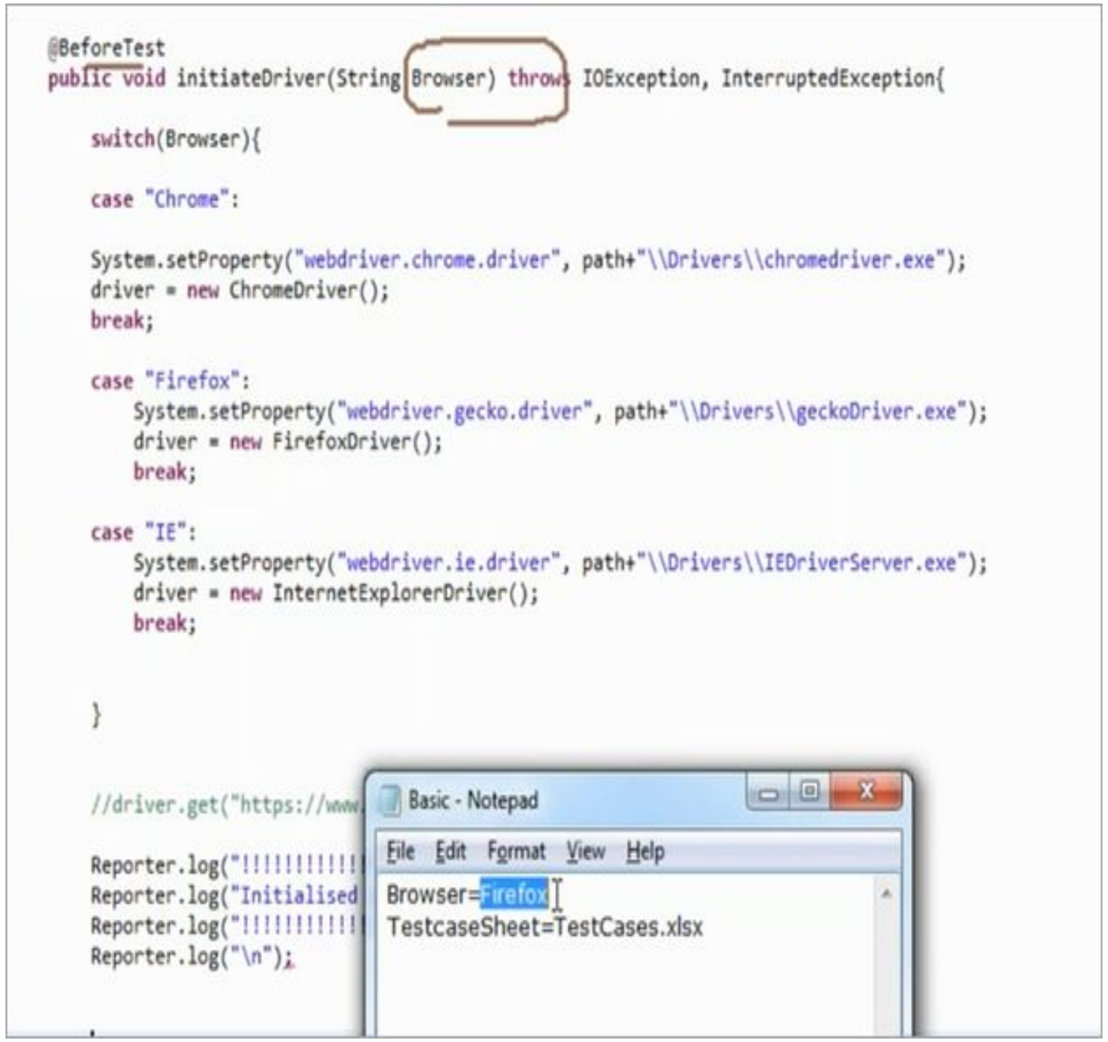


Test Steps	Locator Type	Locator Name	TestData	AssertionType	ExpectedValue
enter_URL			URL		
Click	xpath	LoginLink			
typeIn	xpath	EmailTextbox	email		
Click	xpath	ContinueButton			
typeIn	id	PasswordTextbox	password		
Click	id	SignInButton			
AssertElement	xpath	UserIcon		Displayed	

```
email = test@gmail.com
password = Password@123
URL = https://www.amazon.com
```

UNICAL ACADEMY

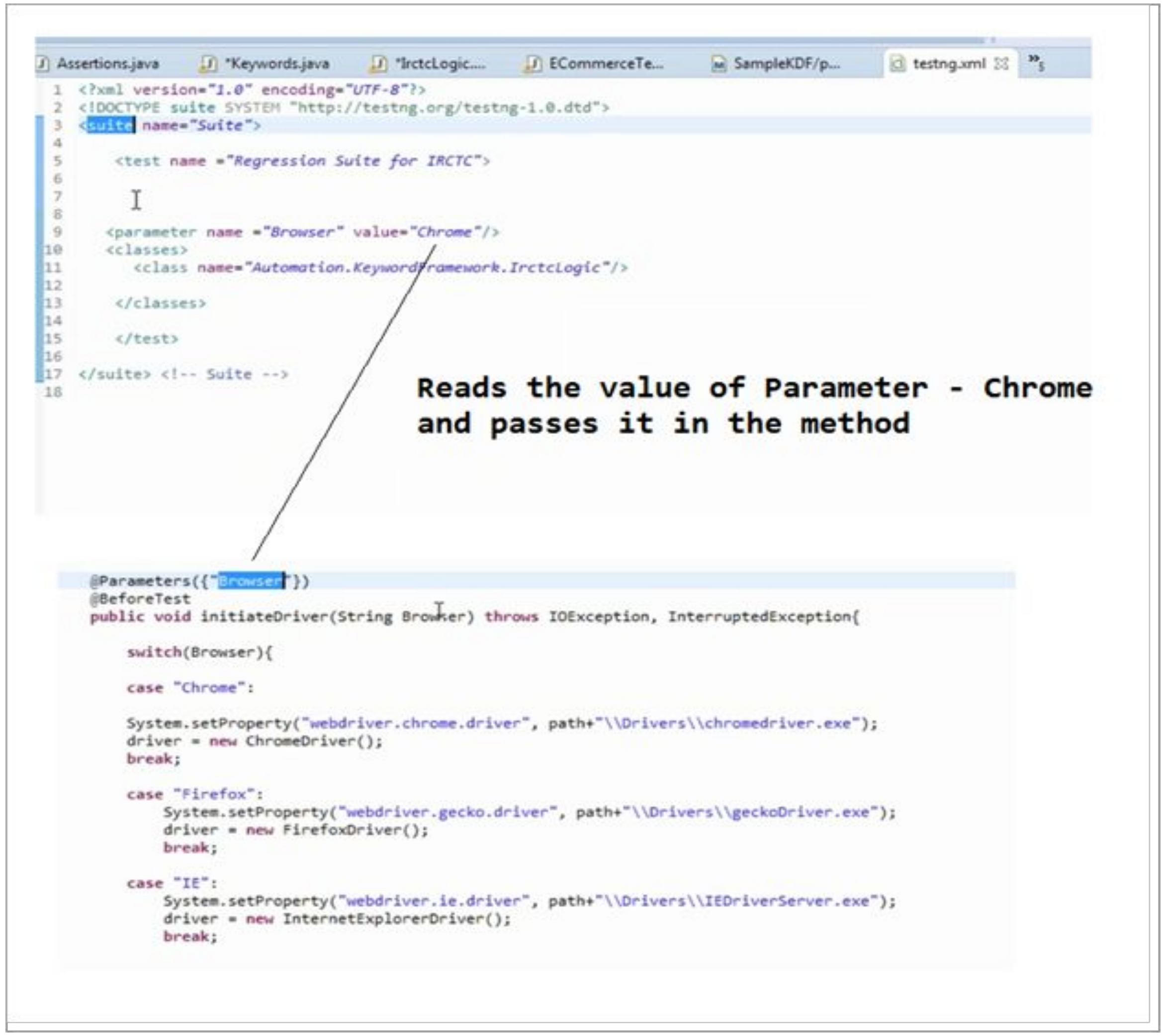
Object Repository for test data in general script:



```
@BeforeTest
public void initiateDriver(String Browser) throws IOException, InterruptedException{

    switch(Browser){

    case "Chrome":

    System.setProperty("webdriver.chrome.driver", path+"\\Drivers\\chromedriver.exe");
    driver = new ChromeDriver();
    break;

    case "Firefox":
        System.setProperty("webdriver.gecko.driver", path+"\\Drivers\\geckoDriver.exe");
        driver = new FirefoxDriver();
        break;

    case "IE":
        System.setProperty("webdriver.ie.driver", path+"\\Drivers\\IEDriverServer.exe");
        driver = new InternetExplorerDriver();
        break;


    }


    //driver.get("https://www

    Reporter.log("!!!!!!!!!!!
    Reporter.log("Initialised
    Reporter.log("!!!!!!!!!!!
    Reporter.log("\n");
```

```
Browser=Firefox
TestcaseSheet=TestCases.xlsx
```

Passing Test Data from TestNG Suite:



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Suite">

  <test name ="Regression Suite for IRCTC">



 <parameter name ="Browser" value="Chrome"/>
 <classes>
   <class name="Automation.KeywordFramework.IrctcLogic"/>

  </classes>

  </test>

</suite> <!-- Suite -->
```

Reads the value of Parameter - Chrome and passes it in the method

```
@Parameters({"Browser"})
@BeforeTest
public void initiateDriver(String Browser) throws IOException, InterruptedException{

    switch(Browser){

    case "Chrome":

    System.setProperty("webdriver.chrome.driver", path+"\\Drivers\\chromedriver.exe");
    driver = new ChromeDriver();
    break;

    case "Firefox":
        System.setProperty("webdriver.gecko.driver", path+"\\Drivers\\geckoDriver.exe");
        driver = new FirefoxDriver();
        break;

    case "IE":
        System.setProperty("webdriver.ie.driver", path+"\\Drivers\\IEDriverServer.exe");
        driver = new InternetExplorerDriver();
        break;
```

UNICAL ACADEMY

Multiple parameters can also be passed to a method as:

Syntax:

```
<parameter name = “Browser” value=”Chrome”/>
<parameter name = “SheetName” value=”TestCase.xls”/>
 <parameter name = “DriverLocation” value=”chromedriver.exe”/>
          <classes>
             <class name= “com.automation.Amazon”/>
         </classes>
   @Parameters({“Chrome”}, {“ TestCase.xls”}, {“ chromedriver.exe”})
   public void init(String Browser, String SheetName, String DriverLocation){
…..
……
……….
}
```

5) Driver Script

- This contains the main logic to read all the test cases from the test case template excel sheet and performs the corresponding action by reading from the library file. The script is designed based on the test case template created.

Page Object Model (POM)

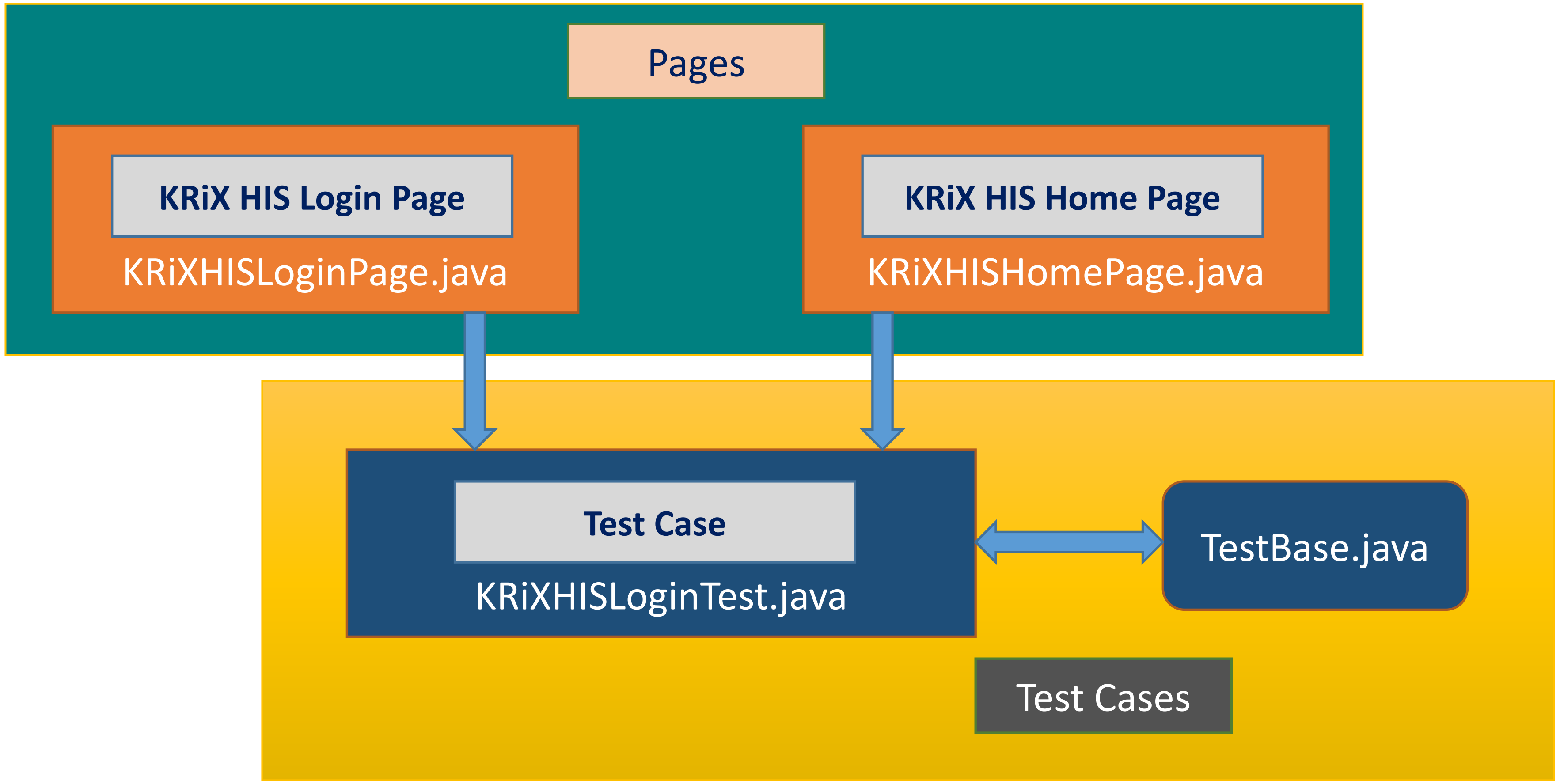
- Page object model is an object design pattern in Selenium test automation used to make a repository of Web UI elements.

- POM improves test readability and reduces code duplication, by acting as an interface for the page under test. In the Page object model, all the webpages have individual page classes. These page classes find the web elements on the page, and contain all the relevant page methods.

```
clickLoginButton();
enterCredentials(user_name,user_password);
checkIfImageIsDisplayed();
```

UNICAL ACADEMY

Page Object Model Design Pattern:



UNICAL ACADEMY

Implementation of Page Object Model:

Step 1: Creating TestBase class. Here we create an object of WebDriver, maximize browser, implementing waits, launching URL and etc.,

Step 2: Creating classes for each page (Eg., *KRiX HIS Login* Page, *KRiX HIS Home* Page) to hold element locators and their methods. Usually, we create page objects for all available pages in the AUT. For each page, we create a separate class with a constructor. Identify all the locators and keep them in one class. It allows us to reuse the locators in multiple methods. It allows us to do easy maintenance, if there is any change in the UI, we can simply change on one Page.

Step 3: Creating Test (Eg., *LMSLoginTest*) based on above pages.

- Launch browser and open http://183.82.4.93:5887/moss/
- Enter user credentials and do login
- Verify the loggedIn user name and do logout

Step 4: Creating testng.xml file

UNICAL ACADEMY

Execute the scripts from Frameworks

Develop a Custom Framework

UNICAL ACADEMY

Below is the outline of the major steps in building a maintainable Selenium framework:

Step 1: Choose a programming language:

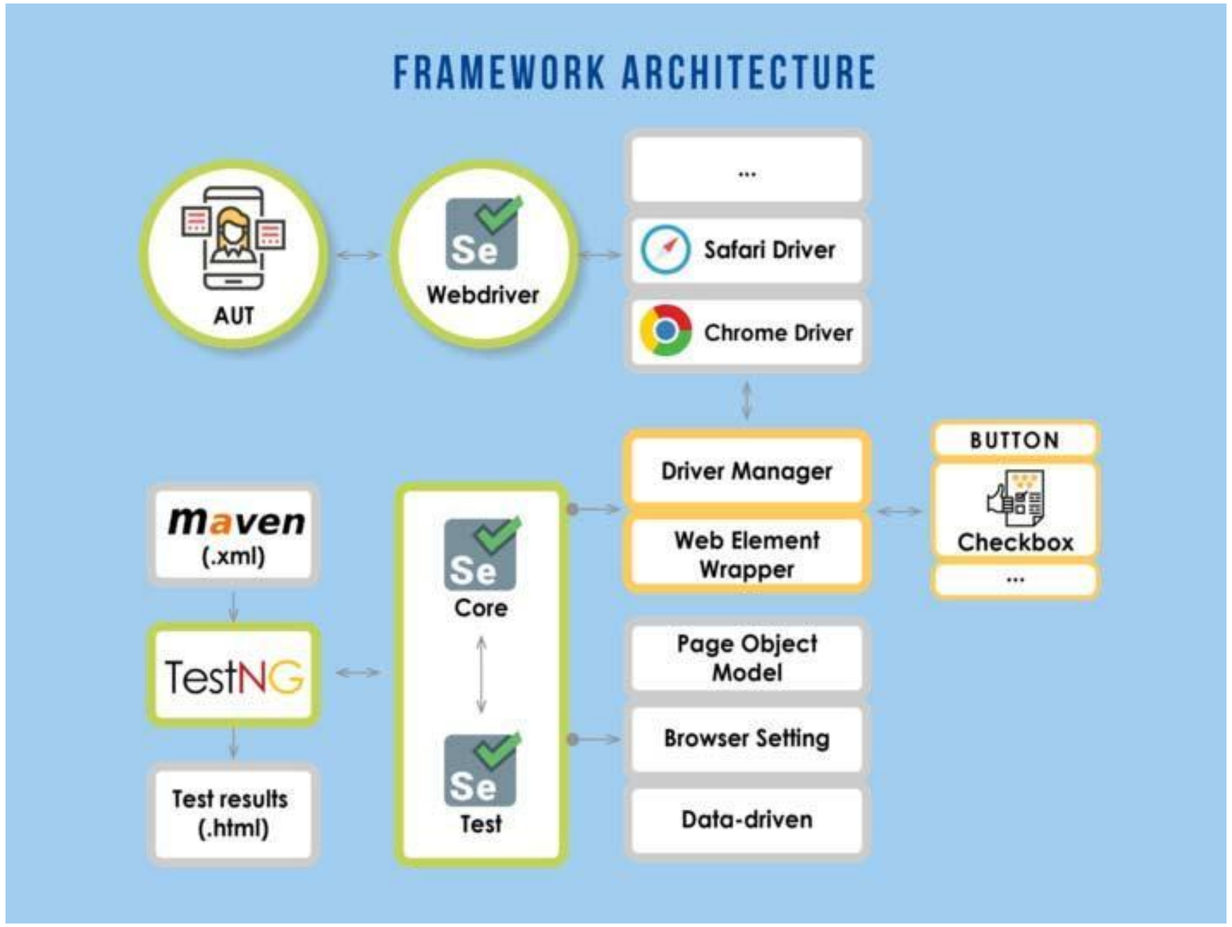
- Programming language of choice has a colossal impact to your framework design & productivity.
- **Java** is the safest choice if you start a new project from scratch since it is widely adopted by the community due to the fact that it works across platforms.

Step 2: Choose a unit test framework:

- Now we've selected the most suitable programming language, we now need to pick a unit test framework that we will build our framework upon.
- **TestNG** is similar to JUnit, but it is much more powerful than JUnit—especially in terms of testing integrated classes.
- **TestNG** eliminates most of the limitations of the older frameworks and gives you the ability to write more flexible and powerful tests. Some of the highlight features are: easy annotations, grouping, sequencing, and parameterizing.

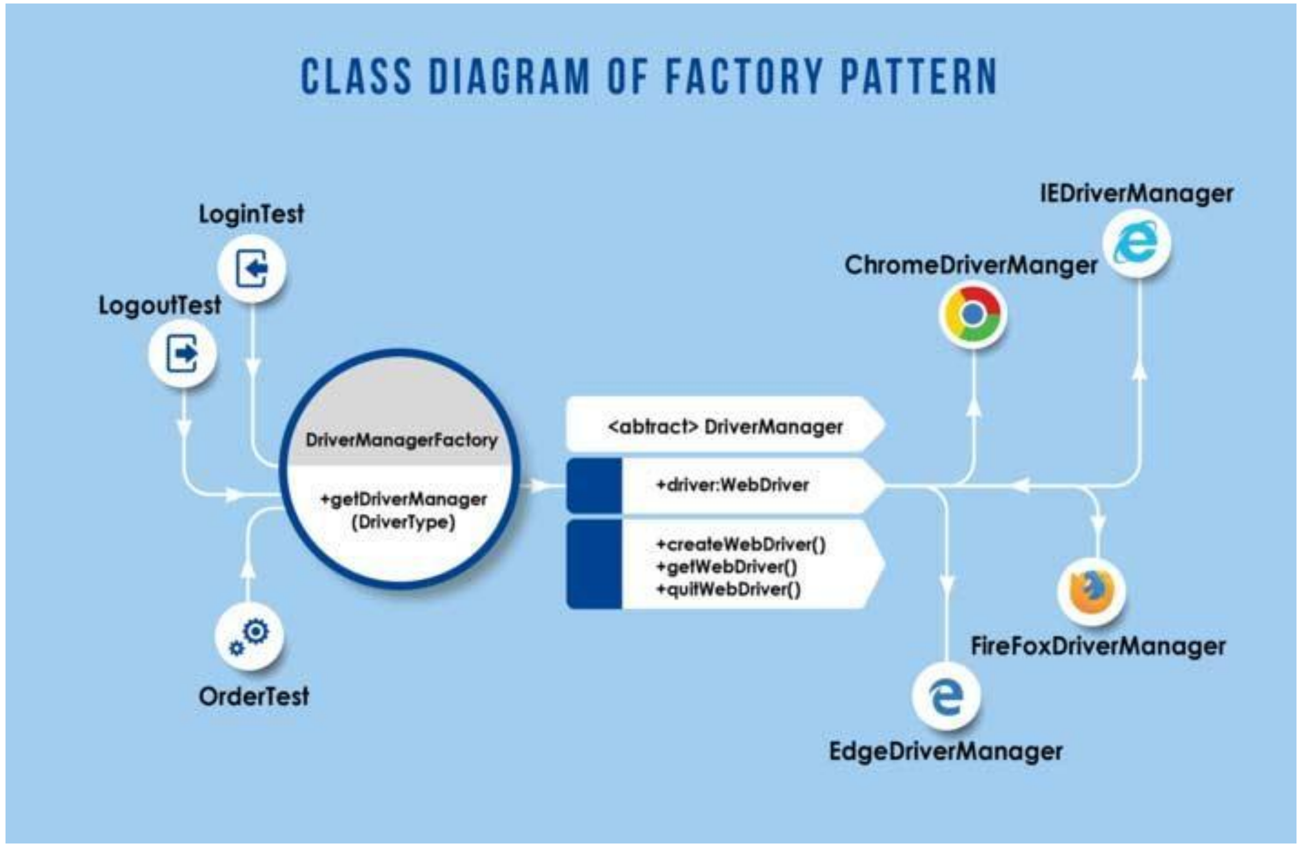
UNICAL ACADEMY

Step 3: Design the framework architecture:



Step 4: Build the SeleniumCore component:

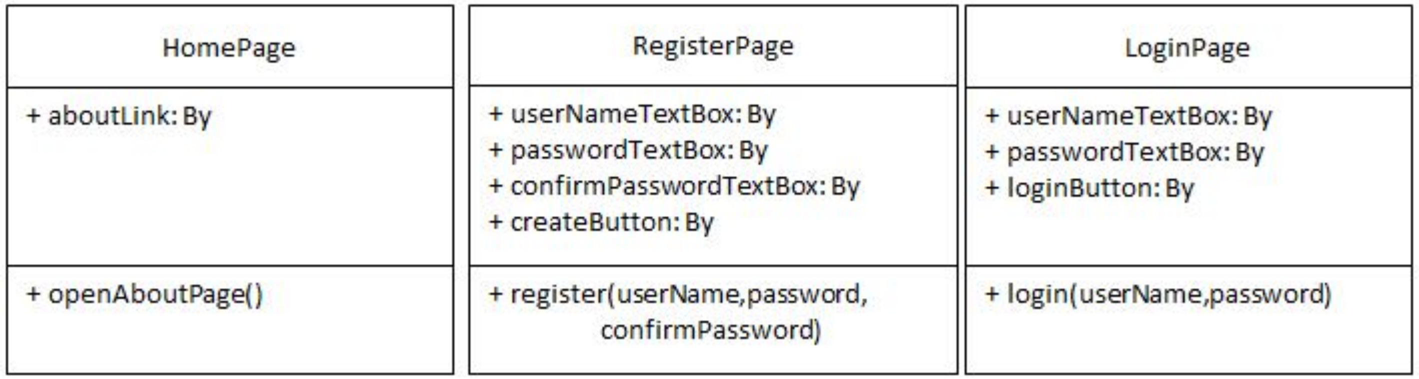
- **SeleniumCore** is designed to manage the browser instances as well as element interactions. This component helps you to create, and destroy WebDriver objects.
- Class diagram explaining how we use the Factory design pattern in our framework.



UNICAL ACADEMY

Step 5: Build the SeleniumTest component:

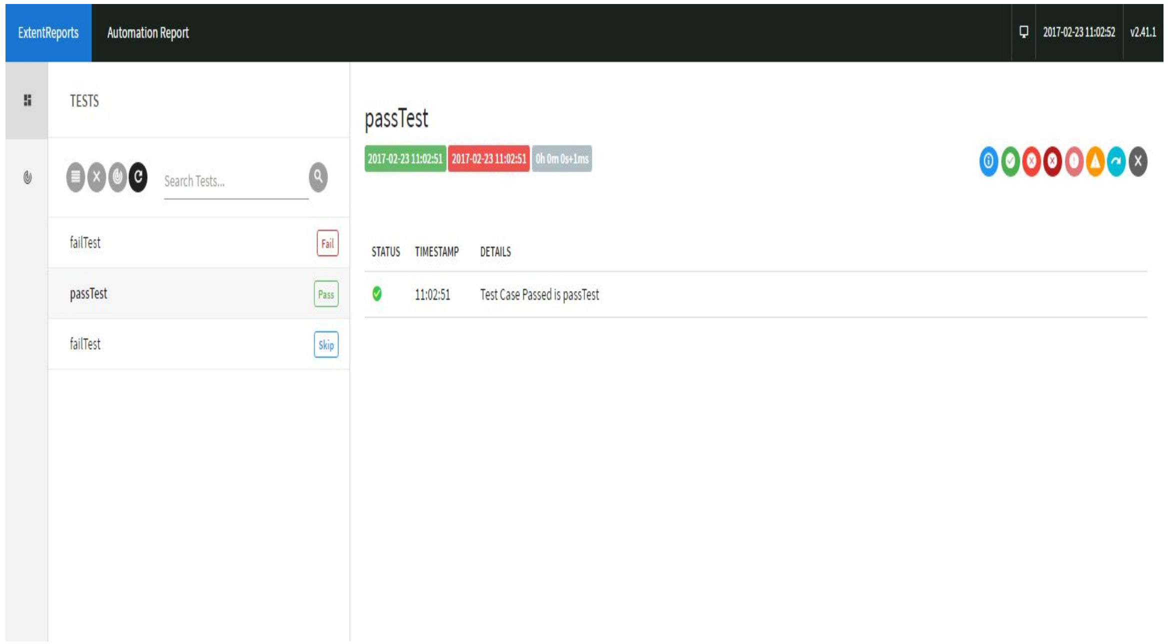
- **SeleniumTest** component contains all test cases that use the classes provided by **SeleniumCore**. As we mentioned earlier, the design pattern we'll apply here is called **Page Object** pattern (POM).
- Page Object Model (POM) has become the de-facto pattern used in test automation frameworks because it reduces duplication of code thus reduces the test maintenance cost.
- If web app includes several pages called the Login page, Home page, Register page, etc., we'll create the corresponding PageObjects for them such as **LoginPage**, **HomePage**, **RegisterPage**, etc.
- The picture below demonstrates how we usually structure PageObjects, their element locators as well as action methods.



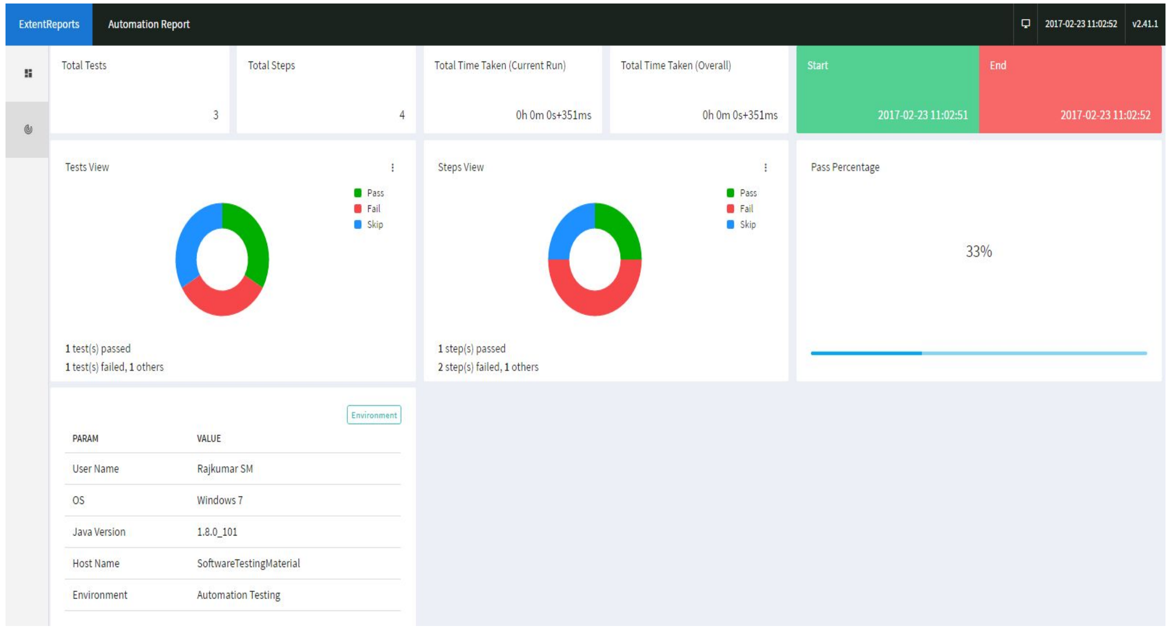
UNICAL ACADEMY

Step 6: Choose a reporting mechanism:

- Reporting mechanisms provided by testing frameworks such as Junit and TestNG are often generated in XML format, which can easily be interpreted by other software like CI/CD tools (Jenkins).
- Third party libraries such as **ExtentReport** and **Allure** can help you create test result reports that are human-readable. They also include visuals like pie charts and screenshots.



STATUS	TIMESTAMP	DETAILS
✓	11:02:51	Test Case Passed is passTest



Total Tests	Total Steps	Total Time Taken (Current Run)	Total Time Taken (Overall)	Start	End
3	4	0h 0m 0s+351ms	0h 0m 0s+351ms	2017-02-23 11:02:51	2017-02-23 11:02:52

PARAM	VALUE
User Name	Rajkumar SM
OS	Windows 7
Java Version	1.8.0_101
Host Name	SoftwareTestingMaterial
Environment	Automation Testing

Step 7: Integrate your framework with other tools:

Consider integrating with the following tools to add more value to your framework:

- **Jira** is a famous eco-system for software development and testing. Thus, consider integrating with Jira in some common scenarios such as automatically posting and closing Jira bugs according to Selenium test results.
- **TestRail** is a test case management (TCM) system that proves useful when your project has a large number of tests and related work items such as bugs and technical tasks. It’s best if our Selenium framework can automatically upload test results to TestRail after execution.
- **AutoIt** is a freeware BASIC-like scripting language designed for automating the Windows GUI and general scripting. It will help you in case you want to work with desktop GUI, like the download dialog of the browser.

UNICAL ACADEMY

Session Recap