TestNG is a testing framework developed in the lines of JUnit and NUnit, however it introduces some new functionalities that make it more powerful and easier to use. TestNG is designed to cover all categories of tests − unit, functional, end-to-end, integration, etc., and it requires JDK 5 or higher. TestNG is a [Testing](http://www.guru99.com/software-testing.html) framework that overcomes the limitations of another popular testing framework called [JUnit](http://www.guru99.com/junit-tutorial.html). The "NG" means "Next Generation". Most Selenium users use this more than JUnit because of its advantages

There are three major advantages of TestNG over JUnit:

* Annotations are easier to understand
* Test cases can be grouped more easily
* Parallel testing is possible

**Why do we need TestNG in Selenium?**

TestNG can generate reports based on our Selenium test results.

1)WebDriver has no native mechanism for generating reports. TestNG can generate the report in a readable format like the one shown below.2) TestNG simplifies the way the tests are coded. There is no more need for a static main method in our tests. The sequence of actions is regulated by easy-to-understand annotations that do not require methods to be static. 3)Uncaught exceptions are automatically handled by TestNG without terminating the test prematurely. These exceptions are reported as failed steps in the report.

|  |  |
| --- | --- |
| **notation** | **Description** |
| **@BeforeSuite** | The annotated method will be run only once before all tests in this suite have run. |
| **@AfterSuite** | The annotated method will be run only once after all tests in this suite have run. |
| **@BeforeClass** | The annotated method will be run only once before the first test method in the current class is invoked. |
| **@AfterClass** | The annotated method will be run only once after all the test methods in the current class have run. |
| **@BeforeTest** | The annotated method will be run before any test method belonging to the classes inside the <test> tag is run. |
| **@AfterTest** | The annotated method will be run after all the test methods belonging to the classes inside the <test> tag have run. |
| **@BeforeGroups** | The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked. |
| **@AfterGroups** | The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked. |
| **@BeforeMethod** | The annotated method will be run before each test method. |
| **@AfterMethod** | The annotated method will be run after each test method. |
| **@DataProvider** | Marks a method as supplying data for a test method. The annotated method must return an Object[ ][ ], where each Object[ ] can be assigned the parameter list of the test method. The @Test method that wants to receive data from this DataProvider needs to use a data Provider name equals to the name of this annotation. |
| **@Factory** | Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[ ]. |
| **@Listeners** | Defines listeners on a test class. |
| **@Parameters** | Describes how to pass parameters to a @Test method. |
| **@Test** | Marks a class or a method as a part of the test. |

**Following are some of the benefits of using annotations:**

1)TestNG identifies the methods it is interested in, by looking up annotations. Hence, method names are not restricted to any pattern or format.2) We can pass additional parameters to annotations.3) Annotations are strongly typed, so the compiler will flag any mistakes right away.4)Test classes no longer need to extend anything (such as Test Case, for JUnit

**Based on the above output, the execution procedure is as follows:**

1)First of all, beforeSuite() method is executed only once.20Lastly, the afterSuite() method executes only once.3)Even the methods beforeTest(), beforeClass(), afterClass(), and afterTest() methods are executed only once.4)eforeMethod() method executes for each test case but before executing the test case.5)afterMethod() method executes for each test case but after executing the test case.6)In between beforeMethod() and afterMethod(), each test case executes.

## Passing Parameters with *Dataproviders*

When you need to pass complex parameters or parameters that need to be created from Java (complex objects, objects read from a property file or a database, etc...), in such cases parameters can be passed using Dataproviders. A Data Provider is a method annotated with *@DataProvider*. This annotation has only one string attribute: its name. If the name is not supplied, the Data Provider’s name automatically defaults to the method’s name. A Data Provider returns an array of objects.

Let us check out examples below of using Dataproviders. The first example is about @DataProvider using Vector, String or Integer as parameter and the second example is about @DataProvider using object as parameter

## Create testng.xml

## Next, let's create testng.xml file in C:\ > TestNG\_WORKSPACE to execute Test case(s). This file captures your entire testing in XML. This file makes it easy to describe all your test suites and their parameters in one file, which you can check in your code repository or email to coworkers. It also makes it easy to extract subsets of your tests or split several runtime configurations (e.g., testng-database.xml would run only tests that exercise your database).

## PAGE OBJECT MODEL:

### Why POM?

Starting a UI Automation in Selenium WebDriver is NOT a tough task. You just need to find elements, perform operations on it.

This is a small script. Script maintenance looks easy. But with time test suite will grow. As you add more and more lines to your code, things become tough.

The chief problem with script maintenance is that if 10 different scripts are using the same page element, with any change in that element, you need to change all 10 scripts. This is time consuming and error prone.

A better approach to script maintenance is to create a separate class file which would find web elements, fill them or verify them. This class can be reused in all the scripts using that element. In future if there is change in the web element, we need to make change in just 1 class file and not 10 different scripts.

This approach is called **Page Object Model(POM)**. It helps make code **more readable, maintainable**, and **reusable.**

### **What is POM?**

* **Page Object Model** is a design pattern to create **Object Repository** for web UI elements.
* Under this model, for each web page in the application there should be corresponding page class.
* This Page class will find the Web Elements of that web page and also contains Page methods which perform operations on those WebElements.
* Name of these methods should be given as per the task they are performing i.e., if a loader is waiting for payment gateway to be appear, POM method name can be waitForPaymentScreenDisplay ().

### **Advantages of POM**

1. Page Object Patten says operations and flows in the UI should be separated from verification. This concept makes our code cleaner and easy to understand.
2. Second benefit is the **object repository is independent of test cases**, so we can use the same object repository for a different purpose with different tools. For example, we can integrate POM with TestNG/JUnit for functional testing and at the same time with JBehave/Cucumber for acceptance testing.
3. Code becomes less and optimized because of the reusable page methods in the POM classes.
4. **Methods** get **more realistic names** which can be easily mapped with the operation happening in UI. i.e. if after clicking on the button we land on the home page, the method name will be like 'gotoHomePage()'.

Simple POM:How to implement POM

It's the basic structure of Page object model (POM) where all Web Elements of the **AUT** and the method that operate on these Web Elements are maintained inside a class file. Task like **verification** should be **separate** as part of Test methods.

### What is Page Factory?

Page Factory is an inbuilt page object model concept for Selenium WebDriver but it is very optimized.

Here as well we follow the concept of separation of Page Object repository and Test methods. Additionally, with the help of PageFactory class we use annotations **@FindBy** to find WebElement. We use initElements method to initialize web elements

**@FindBy** can accept **tagName, partialLinkText, name, linkText, id, css, className, xpath**as attributes.

### AjaxElementLocatorFactory

    One of the key advantage of using Page Factory pattern is AjaxElementLocatorFactory Class.

It is working on lazy loading concept, i.e. a timeout for a Web Element will be assigned to the Object page class with the help of AjaxElementLocatorFactory.

Here, when an operation is performed on an element the wait for its visibility starts from that moment only. If the element is not found in the given time interval, test case execution will throw 'NoSuchElementException' exception.

### Summary

1. Page Object Model is an Object repository design pattern in Selenium WebDriver.
2. POM creates our testing code maintainable, reusable.
3. Page Factory is an optimized way to create object repository in POM concept.
4. AjaxElementLocatorFactory is a lazy load concept in Page Factory pattern to identify WebElements only when they are used in any operation.

# Keyword Driven Framework Testing - Complete Tutorial

A keyword-driven framework is a table-driven testing or action word based testing. It is a software testing method suitable for both manual and automated testing.

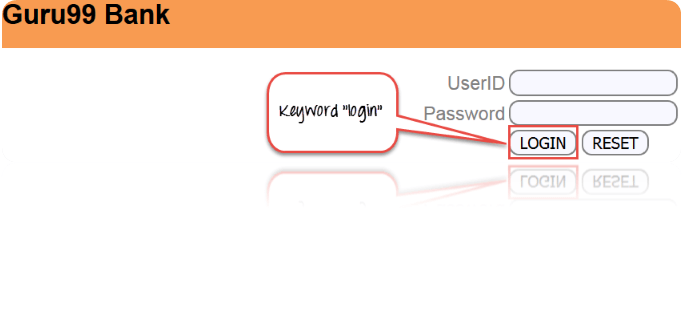
#### What is Keyword Driven Testing?

A keyword-driven testing is a scripting technique that uses data files to contain the keywords related to the application being tested. These keywords describe the set of actions that is required to perform a specific step.

A keyword-driven test consists of high and low-level keywords, including the keyword arguments, which is composed to describe the action of a test case.

In Keyword Driven Testing, you first identify a set of keywords and then associate an action (or function) related to these keywords. Here, every testing action like opening or closing of browser, mouse click, keystrokes, etc. is described by a keyword such as **openbrowser, click, Typtext**and so on.

**For Example**:

[](http://cdn.guru99.com/images/4-2016/042616_0457_WhatisKeywo1.png)

1. **login** to "guru99" website – Keyword "login" will be used in our automation framework, to the test the login function or action associated with it.
2. **logout** to "guru99" website— Keyword "logout" will be used in our automation framework, to test the logout function or action associated with it.

We will see some more example further in the article.

**Example of keywords**

|  |  |
| --- | --- |
| Keywords | Description |
| Login | Login to guru99 bank demo site |
| Emails | Send Email |
| logouts | Log out from guru99 bank demo site |
| Notifications | Find unread notifications |

In order to create a Keyword driven framework, you need following things

1. **Excel Sheet**- Identify the keywords and store them in an excel sheet
2. **Function Library**- Function library consist of the function for the business flows ( login button for any website).So when test is executed, it will read the keyword from the excel sheet and call the functions accordingly
3. **Data Sheets**- Data sheets is used to store the test data that will be used in the application
4. **Object Repository**- based on your keyword driven framework you can use an object repository
5. **Test Scripts**- Based on the design of your framework, you can have test scripts for each manual test case or a single driver script

### Why do Keyword Driven Testing

Keyword Driven Testing is done due to following reason

* Common components handled by standard library
* Using this approach tests can be written in a more abstract manner
* High degree of reusability
* The detail of the script is hidden from the users
* Users don't have to use the scripting languages
* The test is concise, maintainable and flexible

### How to perform Keyword Driven Testing

Keyword based testing can be done in both ways, manually as well as automated. But usually, it is used with automated testing.

The objective behind automating Keyword based testing is

* It helps to reduce maintenance cost
* Avoids duplicated specifications
* Greater reuse of function scripting
* Better testing support and portability
* Achieve more testing with less or same effort

With keyword driven testing, you can create a simple functional tests in the earlier stages of development, testing the application piece-by-piece. The simplest way to compose keyword driven test is to record them. After recording, the test can be modified and customized as per the requirement.

Each keyword needs to be linked with atleast one command, test scripts or function, which implement the actions related to that keyword.

When test cases are executed, keywords are interpreted by a test library, which is called by a test automation framework.

The major activities involved in keyword driven testing are

**Step 1**. Identifying low level as well as high-level keywords

**Step 2**. Implementing the keywords as executable

**Step 3**. Creating test cases

**Step 4**. Creating the driver scripts

**Step 5**. Executing the automation test scripts

### Tools used for Keyword Driven Testing

Few tools which are extensively used for Keyword driven testing.

* [HP QTP](http://www.guru99.com/quick-test-professional-qtp-tutorial.html)
* [Selenium](http://www.guru99.com/selenium-tutorial.html)

### Benefits of Keyword Driven Testing

* It allows functional testers to plan test automation before the application is ready
* Tests can be developed without programming knowledge
* It is not dependent on a specific programming language or tool
* Compatible with any automation tools available in the market

### Sample test cases

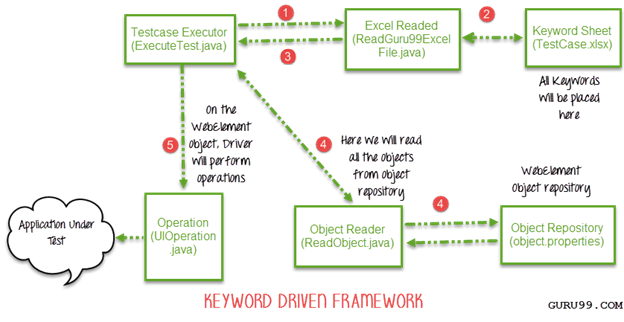
1. TC\_01: Login to guru99 demo site, find out how many transactions is carried out today
2. TC\_02: Login to guru99 demo site, send an email to one of your customer and then logout
3. TC\_03: Login to guru99 demo site and check for any notification received

**Summary:**

* A keyword-driven testing is a scripting technique that uses data files to contain the keywords related to the application being tested
* A keyword-driven testing usually performed by automated testing.
* Tests can be developed without programming knowledge
* Tests are compatible with any automation tools available in the market

### **Keyword Driven Test Framework:**

In keyword driven test framework, all the operations and instructions are written in some external file like excel worksheet. Here is how the complete framework looks like

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0715_CreatingKey2.png)

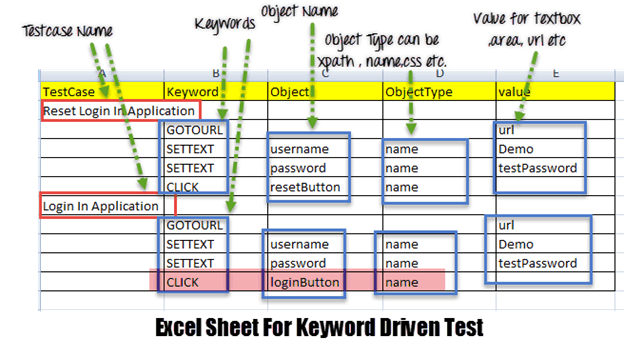
As you can see it's a 5 step framework. Let's study it stepwise in detail

Step 1)

* The driver script Execute.java will call ReadGuru99ExcelFile.java
* ReadGuru99ExcelFile.java has POI script to read data from an Excel

Step 2)

* ReadGuru99ExcelFile.java will read data from TestCase.xlsx
* Here is how the sheet looks like-

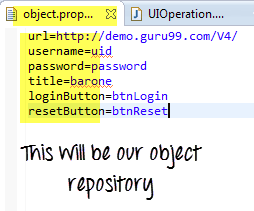
[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0715_CreatingKey3.png)

* According to the keywords written in excel file, the framework will perform the operation on UI.
* For example, we need to click a button 'Login'. Correspondingly, our excel will have a keyword 'Click'. Now the AUT can have hundreds of button on a page, to identify a Login button, in excel we will input Object Name as loginButton & object type as name (see highlighted row in above image). The Object Type could be xpath,name css or any other value

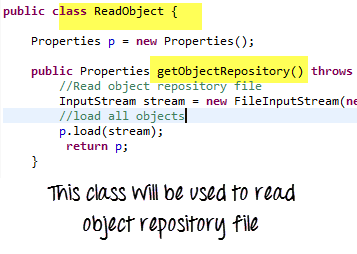
Step 3) ReadGuru99ExcelFile.java will pass this data to the driver script Execute.java

Step 4)

* For all of our UI web elements we need to create an object repository where we will place their element locator (like xpath, name, css path ,class name etc.)

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0715_CreatingKey4.png)

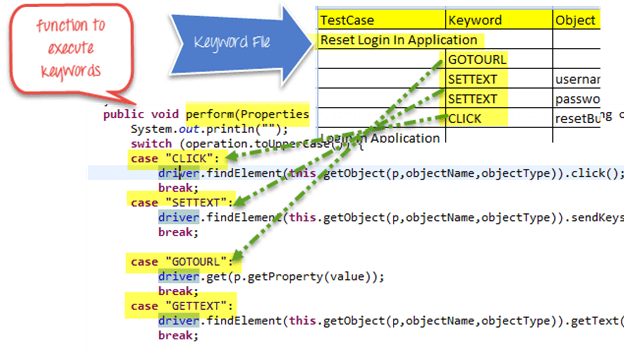
* Execute.java (our driver script) will read the entire Object Repository and store it in a variable
* To read this object repository we need a ReadObject class which has a getObjectRepository method to read it.

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0715_CreatingKey5.png)

NOTE: You may think why do we need to create an object repository. The answer it helps in code maintainence. For example, we are using the a button with name = btnlogin in 10 different test cases. In future , the developer decides to change the name from btnlogin to submit. You will have to make change in all the 10 test cases. In case of an object repository, you will make change just once in the repository.

Step 5)

* The driver will pass the data from Excel & Object Repositoy to UIOperation class
* UIOperation class has functions to perfom actions corresponding to keywords like CLICK, SETTEXT etc… mentioned in the excel
* UIOperation class is a java class which has the actual implementation of the code to perform operations on web elements

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0715_CreatingKey6.png)

The complete project will looks like-

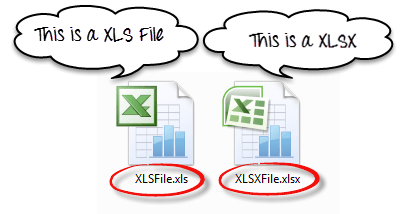
# **All About Excel in Selenium: POI & JXL:**

File IO is a critical part for any software process. We frequently create a file, open it & update something or delete it in our Computers. Same is the case with Selenium Automation. We need a process to manipulate files with Selenium.

Java provides us different classes for File Manipulation with Selenium. In this tutorial we are going to learn how can we read and write on excel file with the help of Java IO package and Apache POI library.

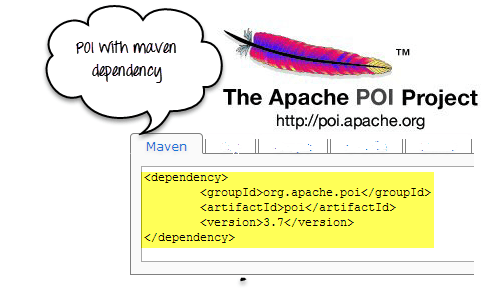
File IO is a critical part for any software process. We frequently create a file, open it & update something or delete it in our Computers. Same is the case with Selenium Automation. We need a process to manipulate files with Selenium.

Java provides us different classes for File Manipulation with Selenium. In this tutorial we are going to learn how can we read and write on excel file with the help of Java IO package and Apache POI library.

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc1.png)

### **Exporting Excel**

* **How to handle excel file using POI (Maven POM Dependency)**

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc2.png)

To read or write an Excel, Apache provide a very famous library POI. This library is capable enough to read and write both **XLS** and **XLSX** file format of excel.

To read **XLS** files an **HSSF** implementation is provided by POI library.

To read **XLSX** , **XSSF** implementation of **POI** **library** will be the choice. Let's study these implementations in detail.

If you are using maven in your project the maven dependency will be

**<dependency>**

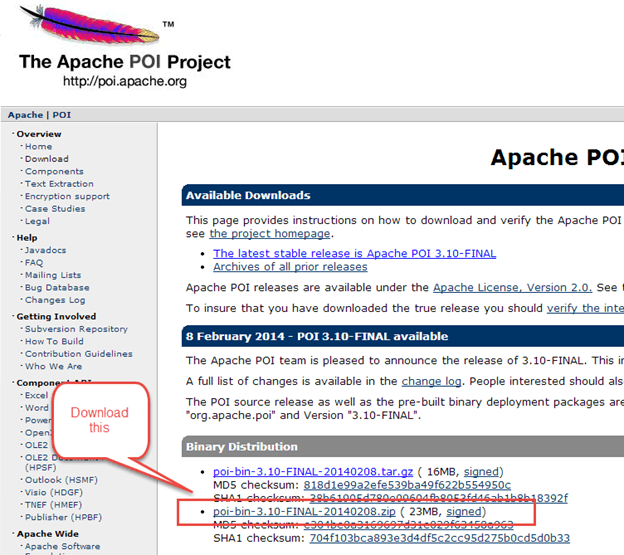
**<groupId>org.apache.poi</groupId>**

**<artifactId>poi</artifactId>**

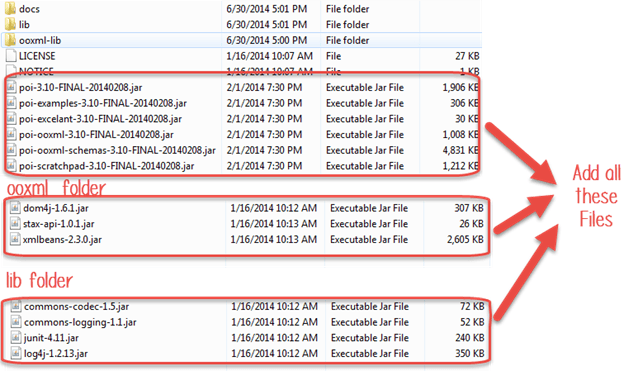
**<version>3.9</version>**

**</dependency>**

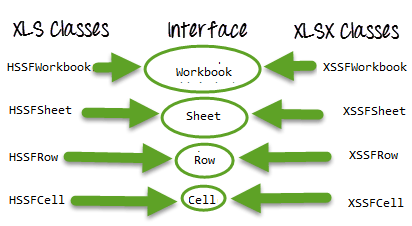
Or you can simply download the latest version POI jars from <http://poi.apache.org/download.html> & download [poi-bin-3.10-FINAL-20140208.zip](http://www.apache.org/dyn/closer.cgi/poi/release/bin/poi-bin-3.10-FINAL-20140208.zip)

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc3.png)

When you download the zip file for this jar , you need to unzip it and add these all jars to the class path of your project.

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc4.png)

### **Classes and Interfaces in POI:**

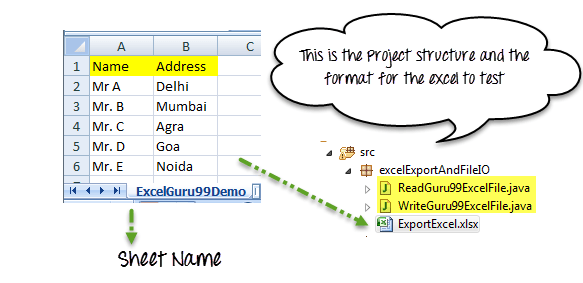
[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc5.png)

Following is a list of different Java Interfaces and classes in **POI** for reading **XLS** and **XLSX** file-

* **Workbook**: XSSFWorkbook and HSSFWorkbook classes implement this interface.
* **XSSFWorkbook**: Is a class representation of XLSX file.
* **HSSFWorkbook**: Is a class representation of XLS file.
* **Sheet**: XSSFSheet and HSSFSheet classes implement this interface.
* **XSSFSheet**: Is a class representing a sheet in a XLSX file.
* **HSSFSheet**: Is a class representing a sheet in a XLS file.
* **Row**: XSSFRow and HSSFRow classes implement this interface.
* **XSSFRow**: Is a class representing a row in sheet of XLSX file.
* **HSSFRow**: Is a class representing a row in sheet of XLS file.
* **Cell**: XSSFCell and HSSFCell classes implement this interface.
* **XSSFCell**: Is a class representing a cell in a row of XLSX file.
* **HSSFCell:** Is a class representing a cell in a row of XLS file.

### **Read/Write operation-**

For our example we will consider below given excel file format

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc6.png)

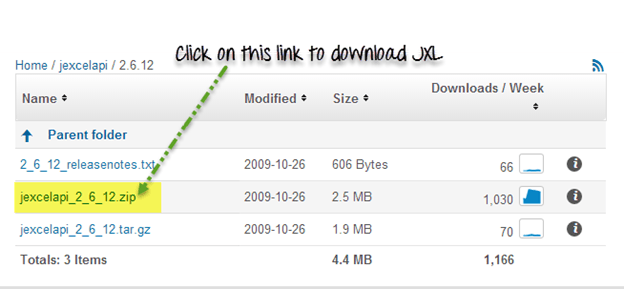
JXL is also another famous jar for reading writing Excel files. Now a day's POI is used in most of the projects but before POI, JXL was only Java API for excel manipulation. It is a very small and simple API.

TIPS: *My suggestion is not to use JXL in any new project because the library is not in active development from 2010 and lack in feature in compare to POI API.*

Download JXL:

If you want to work with JXL you can download it from this link

http://sourceforge.net/projects/jexcelapi/files/jexcelapi/2.6.12/

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0711_AllAboutExc10.png)

You can also get demo example inside this zipped file for JXL.

Some of the features:

* JXL is able to read Excel 95, 97, 2000,XP , 2003 workbook.
* We can work with English, French, Spanish, German.
* Copying a Chart and image insertion in excel is possible

Drawback:

* We can write excel 97 and later only (writing in Excel 95 is not supported).
* JXL does not support XLSX format of excel file.
* It Generate spreadsheet in Excel 2000 format.

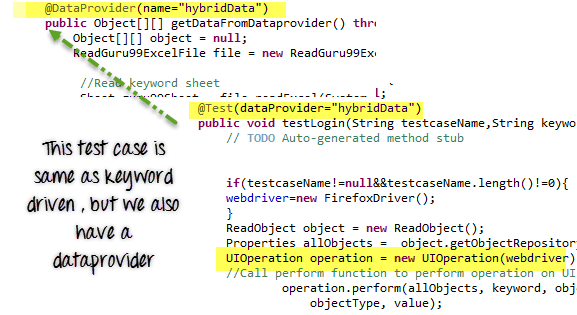
**Summary:**

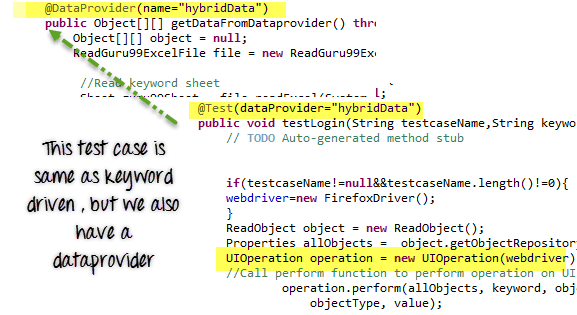
* Excel file can be read by Java IO operation. For that, we need to use **Apache POI Jar**.
* There are two kind of workbook in excel file, **XLSX** and **XLS** files.
* POI has different Interfaces Workbook, Sheet, Row, Cell .
* These interfaces are implemented by corresponding **XLS**(**HSSFWorkbook, HSSFSheet, HSSFRow, HSSFCell** ) and **XLSX**(**XSSFWorkbook, XSSFSheet, XSSFRow, XSSFCell**) file manipulation classes.
* JXL is another API for excel manipulation.
* JXL cannot work with XLSX format of excel.

### Hybrid Test Framework

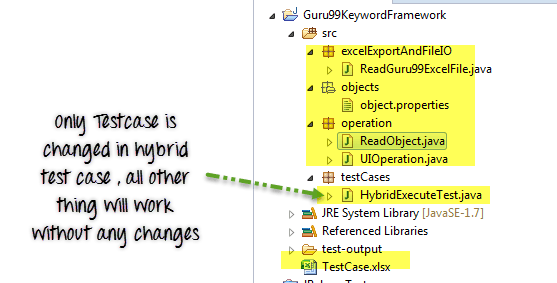
Hybrid Test framework is a concept where we are using advantage of both Keyword and Data driven framework.

Here for keywords we will use excel files to maintain test cases and for test data we can use data provider of TestNG framework.

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0715_CreatingKey9.png)

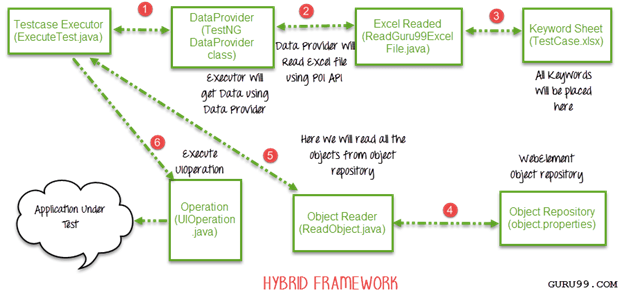
[[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0715_CreatingKey9.png)](http://cdn.guru99.com/images/AdvanceSelenium/071514_0715_CreatingKey9.png)

Here in our hybrid framework we don't need to change anything in Keyword driven framework , here we just need to replace ExecuteTest.java file with HybridExecuteTest.java file.

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0715_CreatingKey10.png)

This HybridExecuteTest file has all the code for keyword driven with data provider concept.

The complete pictorial representation of hybrid framework will look like

[](http://cdn.guru99.com/images/AdvanceSelenium/071514_0715_CreatingKey11.png)

Summary:

* We can create three types of test framework using selenium WebDriver.
* These are Data Driven, Keyword driven and Hybrid test framework.
* We can achieve Data driven framework using TestNG's data provider.
* In Keyword driven framework , keywords are written in some external files like excel file and java code will call this file and execute test cases.
* Hybrid framework is a mix of keyword driven and data driven framework.

BDD

Behavior Driven Development (BDD) is a rising methodology to test and check your code. Cucumber is a flagship BDD tool and this course is geared to make you Cucumber pro!

Before we learn about Cucumber , lets understand BDD -

What is Behaviors Driven Development?

Consider you are assigned to create Funds Transfer module in a Net Banking application.

There are multiple ways to test it

1. Fund Transfer should take place if there is enough balance in source account
2. Fund Transfer should take place if the destination a/c details are correct
3. Fund Transfer should take place if transaction password / rsa code / security authentication for the transaction entered by user us correct
4. Fund Transfer should take place even if it's a Bank Holiday
5. Fund Transfer should take place on a future date as set by the account holder

The test scenario become more elaborate and complex as we consider additional features like transfer amount X for an interval Y days/months , stop schedule transfer when the total amount reaches Z , and so on

The general tendency of developers is to develop features and write test code later. As, evident in above case, test case development for this case is complex and developer will put off testing till release , at which point he will do quick but ineffective testing.

To overcome this issue (Behavior Driven Development) BDD was conceived. It makes the entire testing process easy for a developer

In BDD, whatever you write must go into ***Given-When-Then*** steps. Lets consider the same example above in BDD

***Given****that a fund transfer module in net banking application has been developed   
And I am accessing it with proper authentication*

***When****I shall transfer with enough balance in my source account  
Or I shall transfer on a Bank Holiday  
Or I shall transfer on a future date  
And destination a/c details are correct  
And transaction password/rsa code / security authentication for the transaction is correct  
And press or click send button*

***Then****amount must be transferred  
And the event will be logged in log file*

Isn't it easy to write and read and understand? It covers all possible test cases for the fund transfer module and can be easily modified to accommodate more. Also, it more like writing documentation for the fund transfer module.

**What is Cucumber?**

Cucumber tools also support Behavior Driven Development (BDD).It offers a way to write tests that anybody can understand, regardless of their technical knowledge.

In BDD, users (business analysts, product owners) first write scenarios or acceptance tests that describes the behavior of the system from the customer's perspective, for review and sign-off by the product owners before developers write their codes.

Cucumber use Ruby programming language.

**What are the benefits?**

1. It is helpful to involve business stakeholders who can't easily read code
2. Cucumber focuses on end-user experience
3. Style of writing tests allow for easier reuse of code in the tests
4. Quick and easy set up and execution
5. Efficient tool for testing

**Advantages of Cucumber over other tools?**

|  |  |  |
| --- | --- | --- |
| **Cucumber** | **HP ALM (QTP)** | **Selenium** |
| * It is free | * QTP is expensive | * It is free |
| * It's a behavior driven development tool | * It's a Functional Automation Tool | * It's a Functional and Performance ( Selenium Grid) test tool |
| * Plugin in cucumber works faster | * Plugin are slower compare to Cucumber and Selenium | * Plugins are slower than cucumber |
| * Cucumber supports other language as well beyond Ruby like Java, Scala, Groovy etc. | * QTP supports only VB script | * Selenium supports Java, .Net and many other languages |
| * Writing automation steps are joint effort of testers and developer | * In QTP only tester writes automation steps | * Like Cucumber, writing automation steps are joint effort of testers and developer |
| * Cucumber supports only web environment | * Support web, desktop and any client server application | * Supports only web environment |

For every cucumber project there is a single directory at the root of the project named "**features**". This is where all of your cucumber features will reside. In this directory you will find additional directories, which is **step\_definition** and **support directories**

### **What is "Feature File"?**

Feature File consist of following components -

* **Feature**: A feature would describe the current test script which has to be executed.
* **Scenario**: Scenario describes the steps and expected outcome for a particular test case.
* **Scenario Outline**: Same scenario can be executed for multiple sets of data using scenario outline. The data is provided by a tabular structure separated by (I I).
* **Given**: It specifies the context of the text to be executed. By using datatables "Given", step can also be parameterized.
* **When**: "When" specifies the test action that has to performed
* **Then**: The expected outcome of the test can be represented by "Then"

**Example:**

**Feature:**Visit **career guide** page in career.guru99.com

**Scenario :**Visit career.guru99.com

**Given:** I am on career.guru99.com

**When:**I click on career guide menu

**Then:**I should see career guide page

[](http://cdn.guru99.com/images/cucumber/CucumberLatest.gif)

### **What is "Step Definition"**

Step definition maps the test case steps in the feature files(introduced by Given/When/Then) to code, which executes and checks the outcomes from the system under test. For a step definition to be executed, it must match the given compoent in a feature. Step definition is defined in ruby files under "features/step\_definitions/\*\_steps.rb".

**Example for Step Definition**: Here we will above example of browsing career.guru99.com do We will use features like "When, Then, Given "

Step 1:

Given (/^ I am on career.guru99.com$/) do

Browser.goto "[http://career.guru99.com](http://career.guru99.com/)" -This will visit career.guru99 on browser

end

Step 2:

When (/^ click on career guide menu$/) do

Browser.text (:name, " career guide" ).click – This will click "career guide menu"

end

Step 3:

Then (/^ I should see career guide page$/) do

Browser.goto "<http://career.guru99.com/category/career-guide/>" - It will visit "career guide page"

end

Summary:

* You need 2 Files – Features and Step Definition to execute a Cucmber test scenario
* Features file contain high level description of the test scenario in simple language
* Steps Definition file contains the actual code to execute the test scenario in the Features file.