Data Driven automated testing is a method in which the test data set is created in the excel sheet, and is then imported into automation testing tools to feed to the software under test.

What is Apache POI?

Apache POI (Poor Obfuscation Implementation) is an API written in Java to support read and write operations – modifying office files.

### Why data drive tests?

Often there might be may be a number of data sets that have to be used to test a feature of an application. Now running the same test with different data manually is time-consuming, error prone and a boring task.

**Let us understand this scenario with an example.**

Suppose we need to test the login/Register/ Any form with multiple input fields with 100 different data sets.

**To test this you have three different approaches:**

**1)** Create 100 scripts one for each dataset and execute each test one by one.  
**2)** Change the data in the script and execute it multiple times.  
**3)** Import the data from the excel sheet and execute the script multiple times with different data.

First two scenarios are laborious, time-consuming – implying low ROI. Hence, we must follow the third approach.

### What do we need to implement Data Driven Framework?

In order to follow this approach we must have Eclipse, TestNG properly configured.

***Once done, we will look at:***

* Various interfaces of Apache POI.
* Integration of Apache POI in the Eclipse.
* Read Data from the Excel Sheet.
* Write data to the Excel Sheet.
* Advantages of using Apache POI with Selenium.

### Interface in POI

One of the most remarkable features of **Apache POI** is that it supports read and write operations on both .xls and .xslx files.

Below mentioned are some of the **interfaces of POI**.

* **XSSFWorkbook:** Represents workbook in xlsx file.
* **HSSFWorkbook:** Represents workbook in xls file.
* **XSSFSheet:** Represents a sheet in XLSX file.
* **HSSFSheet:** Represents a sheet in XLS file.
* **XSSFRow:** Represents a row in a sheet of XLSX file.
* **HSSFRow:** Represents a row in a sheet of XLS file.
* **XSSFCell:** Represents a cell in a row of XLSX file.
* **HSSFCell:** Represents a cell in a row of XLS file.

**Fields available in a cell:**

* **CELL\_TYPE\_BLANK:** Represents a blank cell.
* **CELL\_TYPE\_BOOLEAN:** Represents a Boolean cell (true or false).
* **CELL\_TYPE\_ERROR:** Represents an error value in a cell.
* **CELL\_TYPE\_FORMULA:** Represents a formula result on a cell.
* **CELL\_TYPE\_NUMERIC:** Represents numeric data in a cell.
* **CELL\_TYPE\_STRING:** Represents string in a cell.

### The steps to use Selenium with Apache POI

Let us create an automation script to test the login process of a web -based applications.

**Here, I have taken LinkedIn** **as an example**.

We import data from an excel sheet and then use it to log into the application and after execution, we write the result in the excel sheet.

**Step #1)**

Firstly, we need to configure Eclipse with **Apache POI**.

[Download](https://archive.apache.org/dist/poi/release/bin/poi-bin-3.10-FINAL-20140208.zip) jar files for Apache POI.

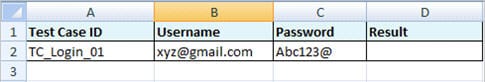
**Step #2)**

Unzip the jar file, and add the following jars to your project and configure them.

* dom4j-1.6.1.jar
* poi-3.10-FINAL-20140208.jar
* poi-ooxml-3.10-FINAL-20140208.jar
* poi-ooxml-schemas-3.10-FINAL-20140208.jar
* xmlbeans-2.3.0.jar

**Step #3)**

After configuring the respective jars, create an excel sheet enter some data in it and save it as *TestData.xlsx* at your preferred location.

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2017/04/excel-sheet.jpg)

**Step #4)**

Now let us follow the sample code to read data from the excel sheet and use it to login to [linkedin.com](https://www.linkedin.com/).

|  |  |  |
| --- | --- | --- |
| 1 | package automationFramework; | |
| 2 |  |

|  |  |
| --- | --- |
| 3 | import java.io.File; |
| 4 | import java.io.FileInputStream; | |

|  |  |  |
| --- | --- | --- |
| 5 | import java.io.FileOutputStream; | |
| 6 | import java.io.IOException; |

|  |  |  |
| --- | --- | --- |
| 7 | import java.util.concurrent.TimeUnit; | |
| 8 |  |

|  |  |  |
| --- | --- | --- |
| 9 | import org.apache.poi.hssf.usermodel.HSSFCell; | |
| 10 | | import org.apache.poi.hssf.usermodel.HSSFSheet; | |

|  |  |  |
| --- | --- | --- |
| 11 | import org.apache.poi.hssf.usermodel.HSSFWorkbook; | |
| 12 | import org.apache.poi.ss.usermodel.Cell; |

|  |  |
| --- | --- |
| 13 | import org.openqa.selenium.By; |
| 14 | import org.openqa.selenium.WebDriver; | |

|  |  |
| --- | --- |
| 15 | import org.openqa.selenium.firefox.FirefoxDriver; |
| 16 | import org.openqa.selenium.support.ui.WebDriverWait; | |

|  |  |  |
| --- | --- | --- |
| 17 | import org.testng.annotations.BeforeTest; | |
| 18 | import org.testng.annotations.Test; |

|  |  |
| --- | --- |
| 19 |  |
| 20 | /\*\* | |

|  |  |  |
| --- | --- | --- |
| 21 | \* @author Admin | |
| 22 | \* |

|  |  |
| --- | --- |
| 23 | \*/ |
| 24 | public class ReadWriteExcel | |

|  |  |
| --- | --- |
| 25 | { |
| 26 | WebDriver driver; | |

|  |  |
| --- | --- |
| 27 | WebDriverWait wait; |
| 28 | HSSFWorkbook workbook; | |

|  |  |  |
| --- | --- | --- |
| 29 | HSSFSheet sheet; | |
| 30 | HSSFCell cell; |

|  |  |
| --- | --- |
| 31 |  |
| 32 | @BeforeTest | |

|  |  |  |
| --- | --- | --- |
| 33 | public void TestSetup() | |
| 34 | { |

|  |  |  |
| --- | --- | --- |
| 35 | | // Set the path of the Firefox driver. |
| 36 | System.setProperty("webdriver.gecko.driver", "C:\\Users\\geckodriver.exe"); | | |

|  |  |  |
| --- | --- | --- |
| 37 | driver = new FirefoxDriver(); | |
| 38 |  |

|  |  |
| --- | --- |
| 39 | // Enter url. |
| 40 | driver.get("http://www.linkedin.com/"); | |

|  |  |  |
| --- | --- | --- |
| 41 | driver.manage().window().maximize(); | |
| 42 |  |

|  |  |
| --- | --- |
| 43 | wait = new WebDriverWait(driver,30); |
| 44 | driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS); | |

|  |  |
| --- | --- |
| 45 | } |
| 46 |  | |

|  |  |
| --- | --- |
| 47 | @Test |
| 48 | public void ReadData() throws IOException | |

|  |  |
| --- | --- |
| 49 | { |
| 50 | // Import excel sheet. | |

|  |  |  |
| --- | --- | --- |
| 51 | File src=new File("C:\\Users\\Admin\\Desktop\\TestData.xls"); | |
| 52 |  |

|  |  |
| --- | --- |
| 53 | // Load the file. |
| 54 | FileInputStream finput = new FileInputStream(src); | |

|  |  |
| --- | --- |
| 55 |  |
| 56 | // Load he workbook. | |

|  |  |  |
| --- | --- | --- |
| 57 | workbook = new HSSFWorkbook(finput); | |
| 58 |  |

|  |  |  |
| --- | --- | --- |
| 59 | // Load the sheet in which data is stored. | |
| 60 | sheet= workbook.getSheetAt(0); |

|  |  |
| --- | --- |
| 61 |  |
| 62 | for(int i=1; i&amp;lt;=sheet.getLastRowNum(); i++) | |

|  |  |
| --- | --- |
| 63 | { |
| 64 | // Import data for Email. | |

|  |  |
| --- | --- |
| 65 | cell = sheet.getRow(i).getCell(1); |
| 66 | cell.setCellType(Cell.CELL\_TYPE\_STRING); | |

|  |  |  |
| --- | --- | --- |
| 67 | driver.findElement(By.id("login-email")).sendKeys(cell.getStringCellValue()); | |
| 68 |  |

|  |  |
| --- | --- |
| 69 | // Import data for password. |
| 70 | cell = sheet.getRow(i).getCell(2); | |

|  |  |
| --- | --- |
| 71 | cell.setCellType(Cell.CELL\_TYPE\_STRING); |
| 72 | driver.findElement(By.id("login-password")).sendKeys(cell.getStringCellValue()); | |

|  |  |  |
| --- | --- | --- |
| 73 |  | |
| 74 | } |

|  |  |  |
| --- | --- | --- |
| 75 | } | |
| 76 |  |

|  |  |
| --- | --- |
| 77 | } |

**Step #5)**

Right click on the test case class and click on *Run as –> TestNG Test.*

**Apache POI** imports data from the excel sheet and uses it to log into our application. Now that we saw how to read data from the excel sheet, let’s look at how to write to the sheet.

|  |  |  |
| --- | --- | --- |
| 1 | package automationFramework; | |
| 2 |  |

|  |  |
| --- | --- |
| 3 | import java.io.File; |
| 4 | import java.io.FileInputStream; | |

|  |  |  |
| --- | --- | --- |
| 5 | import java.io.FileOutputStream; | |
| 6 | import java.io.IOException; |

|  |  |  |
| --- | --- | --- |
| 7 | import java.util.concurrent.TimeUnit; | |
| 8 |  |

|  |  |  |
| --- | --- | --- |
| 9 | import org.apache.poi.hssf.usermodel.HSSFCell; | |
| 10 | | import org.apache.poi.hssf.usermodel.HSSFSheet; | |

|  |  |  |
| --- | --- | --- |
| 11 | import org.apache.poi.hssf.usermodel.HSSFWorkbook; | |
| 12 | import org.apache.poi.ss.usermodel.Cell; |

|  |  |
| --- | --- |
| 13 | import org.openqa.selenium.By; |
| 14 | import org.openqa.selenium.WebDriver; | |

|  |  |
| --- | --- |
| 15 | import org.openqa.selenium.firefox.FirefoxDriver; |
| 16 | import org.openqa.selenium.support.ui.WebDriverWait; | |

|  |  |  |
| --- | --- | --- |
| 17 | import org.testng.annotations.BeforeTest; | |
| 18 | import org.testng.annotations.Test; |

|  |  |
| --- | --- |
| 19 |  |
| 20 | /\*\* | |

|  |  |  |
| --- | --- | --- |
| 21 | \* @author Admin | |
| 22 | \* |

|  |  |
| --- | --- |
| 23 | \*/ |
| 24 | public class ReadWriteExcel | |

|  |  |
| --- | --- |
| 25 | { |
| 26 | WebDriver driver; | |

|  |  |
| --- | --- |
| 27 | WebDriverWait wait; |
| 28 | HSSFWorkbook workbook; | |

|  |  |  |
| --- | --- | --- |
| 29 | HSSFSheet sheet; | |
| 30 | HSSFCell cell; |

|  |  |
| --- | --- |
| 31 |  |
| 32 | @BeforeTest | |

|  |  |  |
| --- | --- | --- |
| 33 | public void TestSetup() | |
| 34 | { |

|  |  |  |
| --- | --- | --- |
| 35 | | // Set the path of the Firefox driver. |
| 36 | System.setProperty("webdriver.gecko.driver", "C:\\Users\\geckodriver.exe"); | | |

|  |  |  |
| --- | --- | --- |
| 37 | driver = new FirefoxDriver(); | |
| 38 |  |

|  |  |
| --- | --- |
| 39 | // Enter url. |
| 40 | driver.get("http://www.linkedin.com/"); | |

|  |  |  |
| --- | --- | --- |
| 41 | driver.manage().window().maximize(); | |
| 42 |  |

|  |  |
| --- | --- |
| 43 | wait = new WebDriverWait(driver,30); |
| 44 | driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS); | |

|  |  |
| --- | --- |
| 45 | } |
| 46 |  | |

|  |  |
| --- | --- |
| 47 | @Test |
| 48 | public void ReadData() throws IOException | |

|  |  |
| --- | --- |
| 49 | { |
| 50 | // Import excel sheet. | |

|  |  |  |
| --- | --- | --- |
| 51 | File src=new File("C:\\Users\\Admin\\Desktop\\TestData.xls"); | |
| 52 |  |

|  |  |
| --- | --- |
| 53 | // Load the file. |
| 54 | FileInputStream finput = new FileInputStream(src); | |

|  |  |
| --- | --- |
| 55 |  |
| 56 | // Load he workbook. | |

|  |  |  |
| --- | --- | --- |
| 57 | workbook = new HSSFWorkbook(finput); | |
| 58 |  |

|  |  |  |
| --- | --- | --- |
| 59 | // Load the sheet in which data is stored. | |
| 60 | sheet= workbook.getSheetAt(0); |

|  |  |
| --- | --- |
| 61 |  |
| 62 | for(int i=1; i&amp;lt;=sheet.getLastRowNum(); i++) | |

|  |  |
| --- | --- |
| 63 | { |
| 64 | // Import data for Email. | |

|  |  |
| --- | --- |
| 65 | cell = sheet.getRow(i).getCell(1); |
| 66 | cell.setCellType(Cell.CELL\_TYPE\_STRING); | |

|  |  |  |
| --- | --- | --- |
| 67 | driver.findElement(By.id("login-email")).sendKeys(cell.getStringCellValue()); | |
| 68 |  |

|  |  |
| --- | --- |
| 69 | // Import data for password. |
| 70 | cell = sheet.getRow(i).getCell(2); | |

|  |  |
| --- | --- |
| 71 | cell.setCellType(Cell.CELL\_TYPE\_STRING); |
| 72 | driver.findElement(By.id("login-password")).sendKeys(cell.getStringCellValue()); | |

|  |  |
| --- | --- |
| 73 |  |
| 74 | // Write data in the excel. | |

|  |  |  |
| --- | --- | --- |
| 75 | FileOutputStream foutput=new FileOutputStream(src); | |
| 76 |  |

|  |  |
| --- | --- |
| 77 | // Specify the message needs to be written. |
| 78 | String message = "Data Imported Successfully."; | |

|  |  |
| --- | --- |
| 79 |  |
| 80 | // Create cell where data needs to be written. | |

|  |  |  |
| --- | --- | --- |
| 81 | sheet.getRow(i).createCell(3).setCellValue(message); | |
| 82 |  |

|  |  |
| --- | --- |
| 83 | // Specify the file in which data needs to be written. |
| 84 | FileOutputStream fileOutput = new FileOutputStream(src); | |

|  |  |
| --- | --- |
| 85 |  |
| 86 | // finally write content | |

|  |  |  |
| --- | --- | --- |
| 87 | workbook.write(fileOutput); | |
| 88 |  |

|  |  |
| --- | --- |
| 89 | // close the file |
| 90 | fileOutput.close(); | |

|  |  |  |
| --- | --- | --- |
| 91 |  | |
| 92 | } |

|  |  |  |
| --- | --- | --- |
| 93 | } | |
| 94 | } |

**Note*:****If you encounter any problems during this process, please check the following points.*

* Make sure all the mentioned jars are added to the project and are properly configured.
* Required software is correctly installed.
* Proper use of an interface with respect to excel file, like HSSF for .xls and XSSF for .xlsx.
* Valid row and column index is used.
* Excel file must be closed before execution.
* Proper classes used for the excel file like XSSF used for .xlsx files and HSSF used for .xls files.

Advantages of using Data Driven Framework

* Improves test coverage.
* Re-usability of code.
* Less maintenance.
* Faster Execution.
* Permits better error handling.