Selenium Automation Framework: Data Driven, Keyword Driven & Hybrid

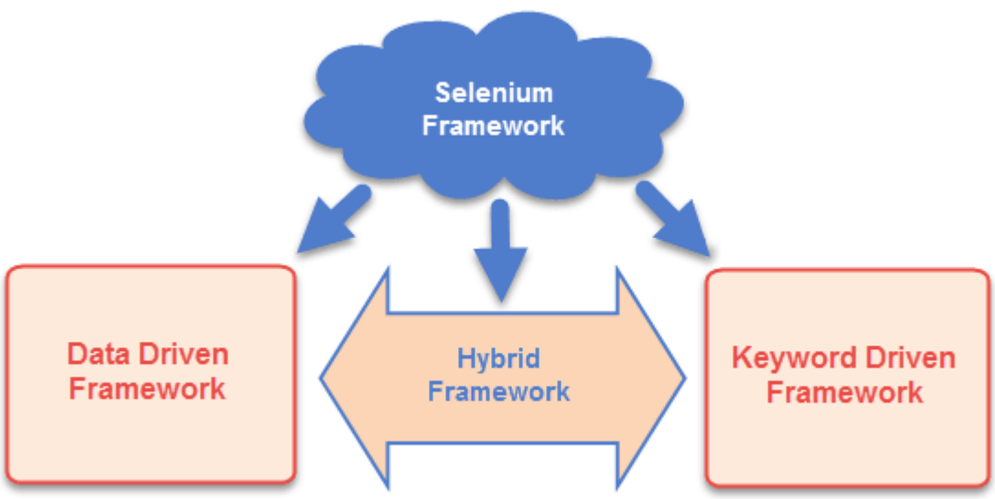
By Krishna Rungta Updated October 7, 2021

What is Selenium Framework?

The **Selenium Framework** is a code structure that makes code maintenance easy and efficient. Without frameworks, users may place the “code” and “data” at the same location which is neither reusable nor readable. Frameworks produce beneficial outcomes like increased code reusability, higher portability, reduced cost of script maintenance, better code readability, etc.

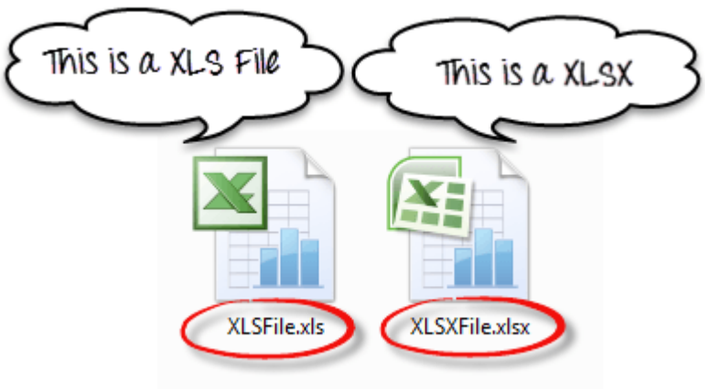
There are mainly three type of frameworks created by Selenium WebDriver to automate manual test cases

- Data Driven Test Framework
- Keyword Driven Test Framework
- Hybrid Test Framework



Data Driven Framework in Selenium

Data Driven Framework in Selenium is a method of separating data sets from the test case. Once the data sets are separated from the test case, it can be easily modified for a specific functionality without changing the code. It is used to fetch test cases and suites from external files like Excel, .csv, .xml or some database tables.



To read or write an Excel, Apache provides 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.

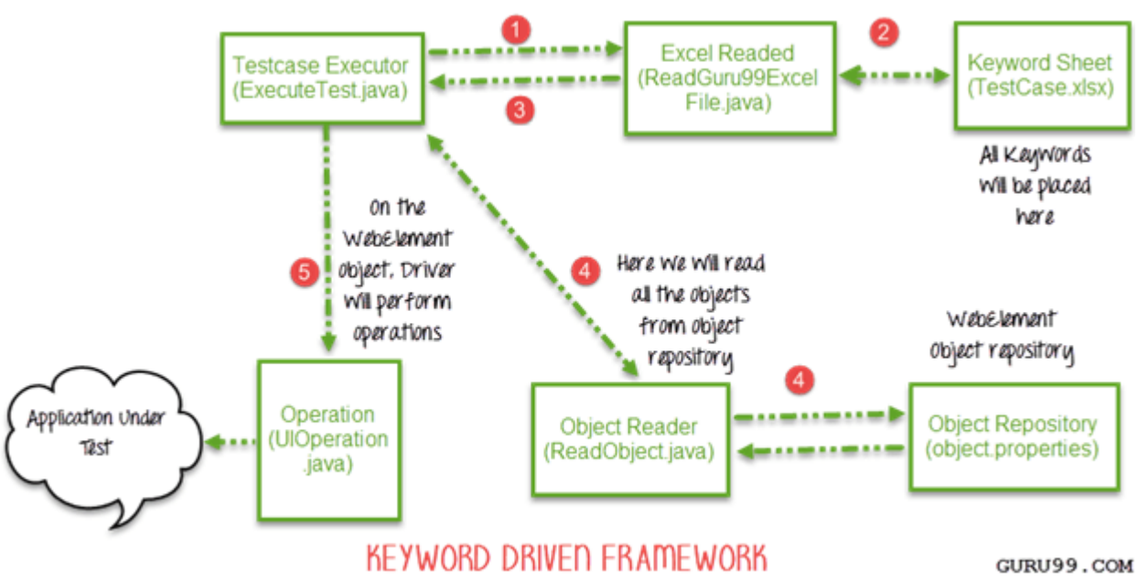
To read **XLSX, XSSF** implementation of **POI library** will be the choice. Let's study these implementations in detail.

We already learned about Data Driven Testing in our **previous tutorial**

Keyword Driven Framework in Selenium

Keyword Driven Framework in Selenium is a method used for speeding up automated testing by separating keywords for common set of functions and instructions. All the operations and instructions to be performed are written in some external file like an Excel sheet. Users can easily control and specify the functionalities they want to test.

Here is how the complete framework looks like



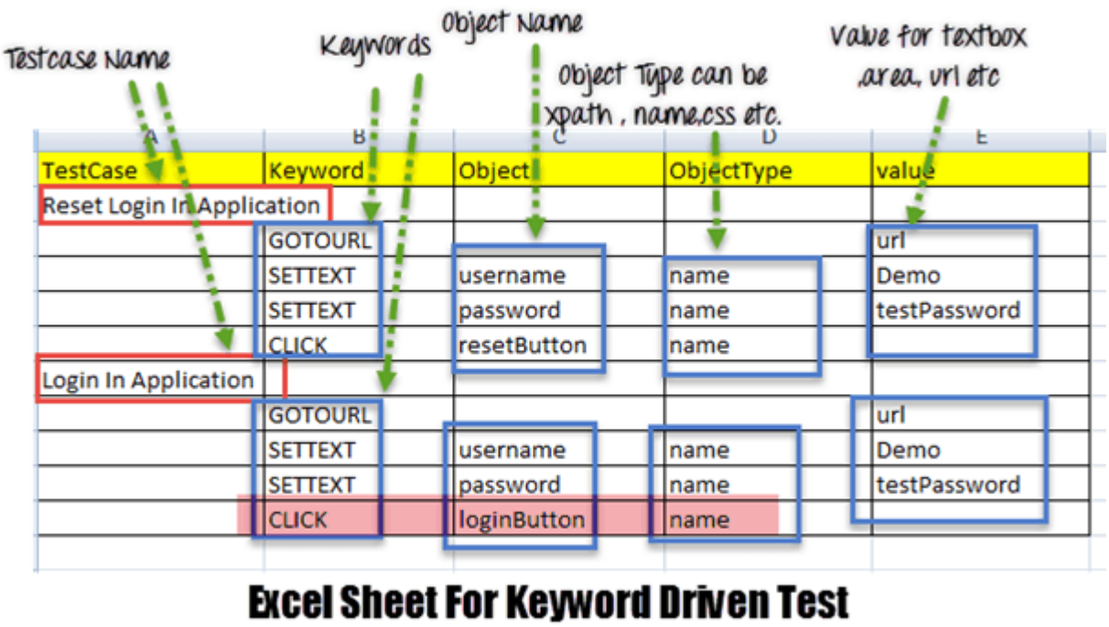
As you can see it's a 5 step framework. Let's study it stepwise in detail

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-



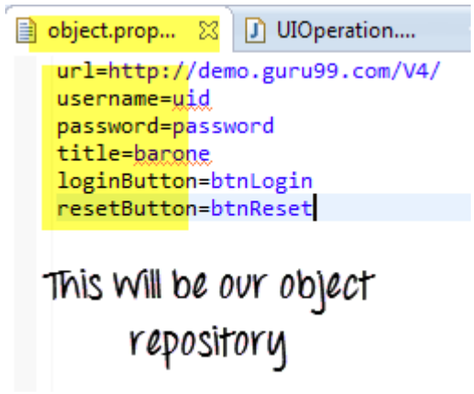
Excel Sheet For Keyword Driven Test

- 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 a name (see highlighted the row in above image). The Object Type could be Xpath, name CSS or any other value

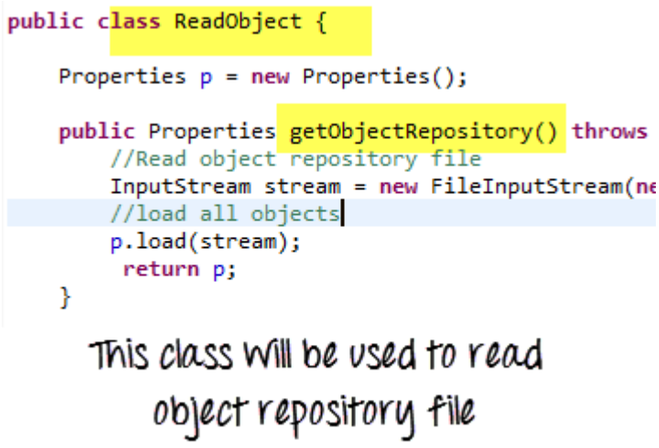
Step 3) ReadGuru99ExcelFile.java will pass this data to the driver script Execute.java

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.)



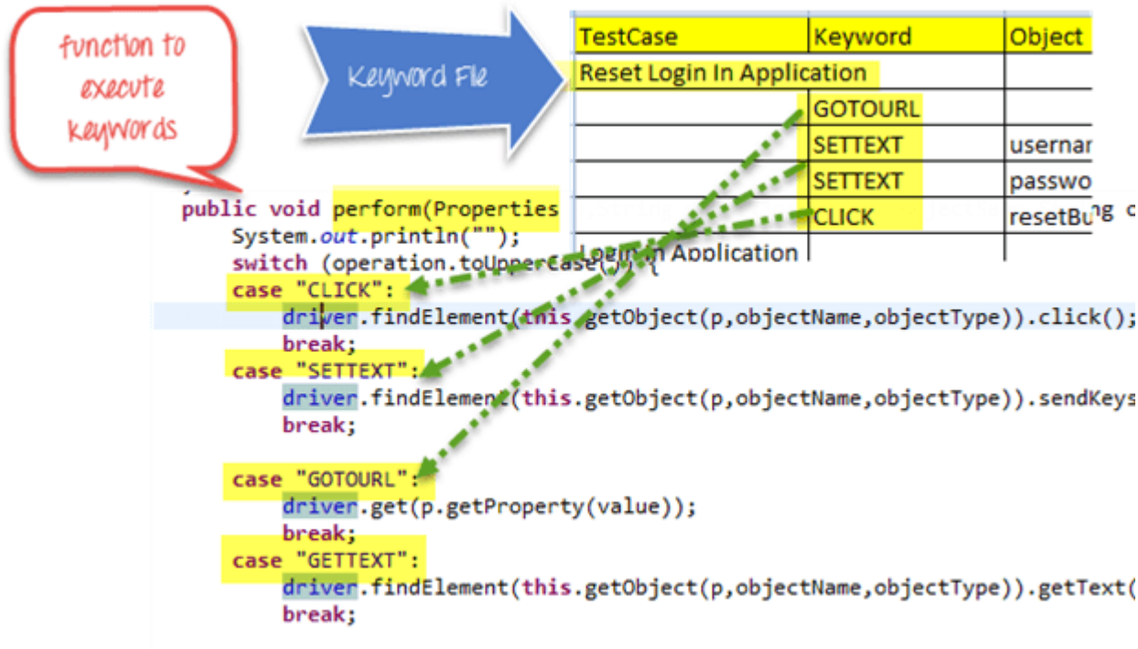
- Execute.java (our driver script) will read the entire Object Repository and store it in a variable
- To read this object repository, we need a ReadObject class which has a getObjectRepository method to read it.



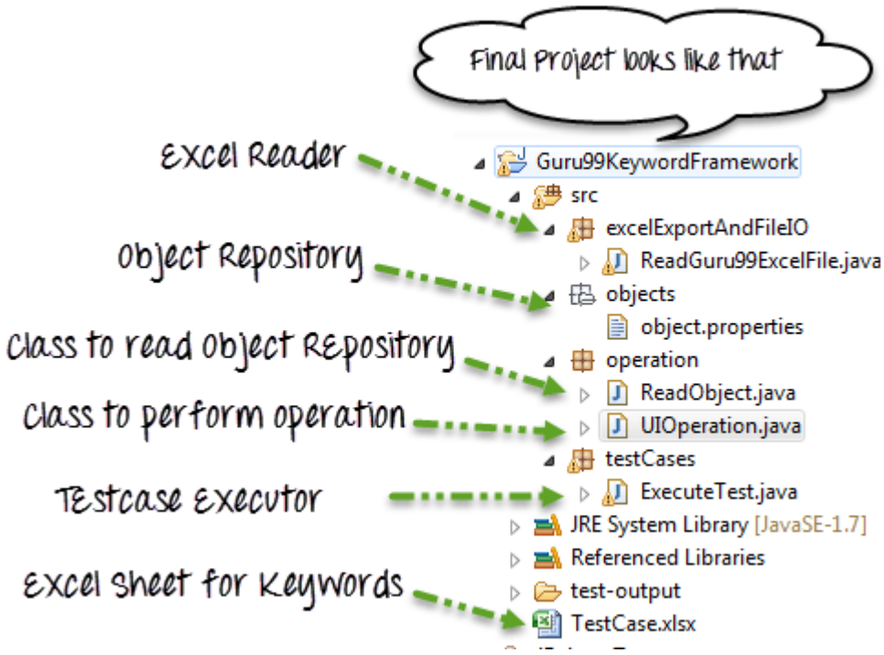
NOTE: You may think why do we need to create an object repository. The answer helps in code maintenance. For example, we are using the button with name = btnlogin in 10 different test cases. In future, the developer decides to change the name from btnlogin to submit. You will have to make a change in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

Step 5)

- The driver will pass the data from Excel & Object Repository to UIOperation class
- UIOperation class has functions to perform 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



The complete project will look like-



Let's look into an example:

Test Scenario: We are executing 2 test cases

- **Test Case 1:**
  - Goto `http://demo.guru99.com/V4/`
  - Enter User ID
  - Enter Password
  - Click Reset
- **Test Case 2:**
  - Goto `http://demo.guru99.com/V4/`
  - Enter User ID
  - Enter Password
  - Click Login

object.properties

url= `http://demo.guru99.com/V4/`

url=http://demo.guru99.com/V4/

username=uid

password=password

title=barone

loginButton=btnLogin

resetButton=btnReset

ReadGuru99ExcelFile.java

```
package excelExportAndFileIO;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
public class ReadGuru99ExcelFile {

    public Sheet readExcel(String filePath,String fileName,String sheetName)
throws IOException{
    //Create a object of File class to open xlsx file
    File file =    new File(filePath+"\\"+fileName);
    //Create an object of FileInputStream class to read excel file
    FileInputStream inputStream = new FileInputStream(file);
    Workbook guru99Workbook = null;
    //Find the file extension by spliting file name in substing and getting only
extension name
    String fileExtensionName = fileName.substring(fileName.indexOf("."));
    //Check condition if the file is xlsx file
    if(fileExtensionName.equals(".xlsx")){
    //If it is xlsx file then create object of XSSFWorkbook class
    guru99Workbook = new XSSFWorkbook(inputStream);
    }
    //Check condition if the file is xls file
    else if(fileExtensionName.equals(".xls")){
        //If it is xls file then create object of XSSFWorkbook class
        guru99Workbook = new HSSFWorkbook(inputStream);
    }
    //Read sheet inside the workbook by its name
    Sheet guru99Sheet = guru99Workbook.getSheet(sheetName);
     return guru99Sheet;
    }
}
```

ReadObject.java

```
package operation;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;
public class ReadObject {
    Properties p = new Properties();
    public Properties getObjectRepository() throws IOException{
        //Read object repository file
        InputStream stream = new FileInputStream(new
File(System.getProperty("user.dir")+"\\src\\objects\\object.properties"));
        //load all objects
        p.load(stream);
         return p;
    }

}
```

UIOperation.java

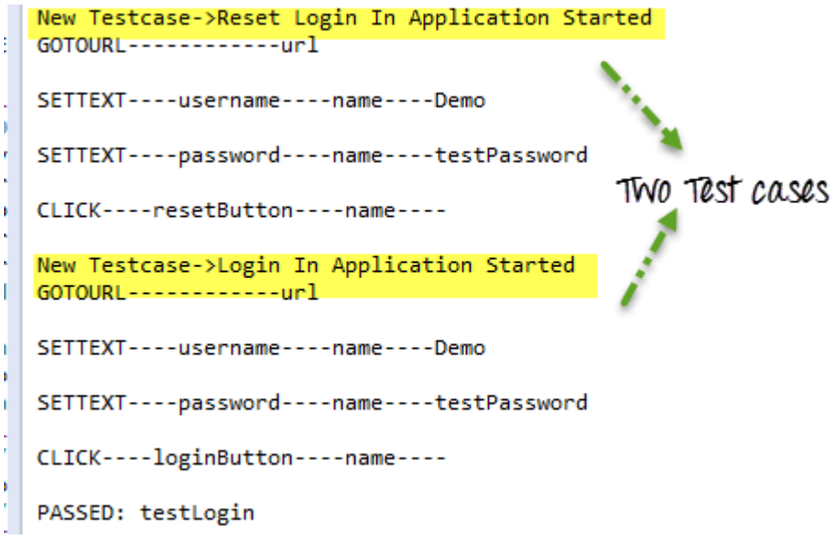
```
package operation;
import java.util.Properties;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
public class UIOperation {
    WebDriver driver;
    public UIOperation(WebDriver driver){
        this.driver = driver;
    }
    public void perform(Properties p,String operation,String objectName,String
objectType,String value) throws Exception{
        System.out.println("");
        switch (operation.toUpperCase()) {
        case "CLICK":
            //Perform click
            driver.findElement(this.getObject(p,objectName,objectType)).click();
            break;
        case "SETTEXT":
            //Set text on control

driver.findElement(this.getObject(p,objectName,objectType)).sendKeys(value);
            break;

        case "GOTOURL":
            //Get url of application
            driver.get(p.getProperty(value));
            break;
        case "GETTEXT":
            //Get text of an element

driver.findElement(this.getObject(p,objectName,objectType)).getText();
            break;
        default:
            break;
```

```
            break;
        }

    }

    /**
     * Find element BY using object type and value
     * @param p
     * @param objectName
     * @param objectType
     * @return
     * @throws Exception
     */
    private By getObject(Properties p,String objectName,String objectType)
throws Exception{
        //Find by xpath
        if(objectType.equalsIgnoreCase("XPATH")){

            return By.xpath(p.getProperty(objectName));
        }
        //find by class
        else if(objectType.equalsIgnoreCase("CLASSNAME")){

            return By.className(p.getProperty(objectName));

        }
        //find by name
        else if(objectType.equalsIgnoreCase("NAME")){

            return By.name(p.getProperty(objectName));

        }
        //Find by css
        else if(objectType.equalsIgnoreCase("CSS")){

            return By.cssSelector(p.getProperty(objectName));

        }
        //find by link
        else if(objectType.equalsIgnoreCase("LINK")){

            return By.linkText(p.getProperty(objectName));

        }
        //find by partial link
        else if(objectType.equalsIgnoreCase("PARTIALLINK")){

            return By.partialLinkText(p.getProperty(objectName));

        }else
        {
            throw new Exception("Wrong object type");
        }
    }
}
```

ExecuteTest.java

```
package testCases;
import java.util.Properties;
import operation.ReadObject;
import operation.UIOperation;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.Test;
import excelExportAndFileIO.ReadGuru99ExcelFile;
public class ExecuteTest {
@Test
    public void testLogin() throws Exception {
        // TODO Auto-generated method stub
WebDriver webdriver = new FirefoxDriver();
ReadGuru99ExcelFile file = new ReadGuru99ExcelFile();
ReadObject object = new ReadObject();
Properties allObjects = object.getObjectRepository();
UIOperation operation = new UIOperation(webdriver);
//Read keyword sheet
Sheet guru99Sheet =
file.readExcel(System.getProperty("user.dir")+"\\","TestCase.xlsx" ,
"KeywordFramework");
//Find number of rows in excel file
    int rowCount = guru99Sheet.getLastRowNum()-guru99Sheet.getFirstRowNum();
    //Create a loop over all the rows of excel file to read it
    for (int i = 1; i < rowCount+1; i++) {
        //Loop over all the rows
        Row row = guru99Sheet.getRow(i);
        //Check if the first cell contain a value, if yes, That means it is the
new testcase name
        if(row.getCell(0).toString().length()==0){
        //Print testcase detail on console
            System.out.println(row.getCell(1).toString()+"----"+
row.getCell(2).toString()+"----"+
            row.getCell(3).toString()+"----"+ row.getCell(4).toString());
        //Call perform function to perform operation on UI
            operation.perform(allObjects, row.getCell(1).toString(),
row.getCell(2).toString(),
                row.getCell(3).toString(), row.getCell(4).toString());
     }
        else{
            //Print the new testcase name when it started
                System.out.println("New Testcase->"+row.getCell(0).toString() +"
Started");
            }
        }
    }
}
```

After execution, output will look like –

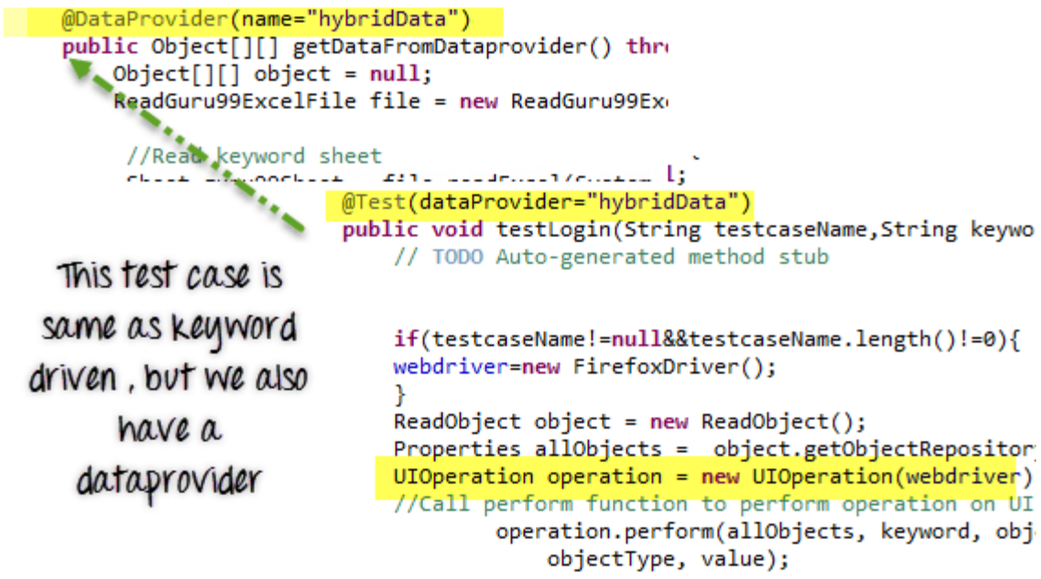


Download the Selenium Project Files for the Demo in this Tutorial

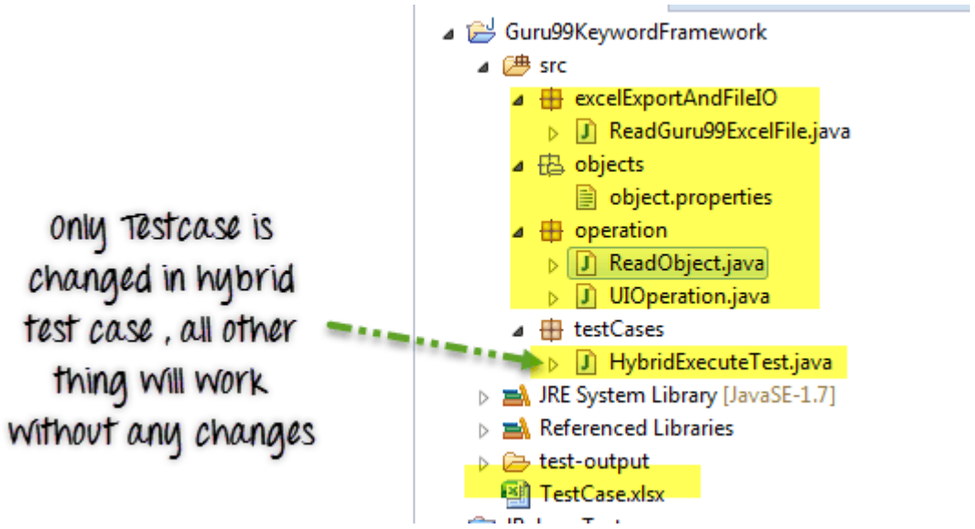
Hybrid Framework

Hybrid Framework in Selenium is a concept where we are using the advantage of both Keyword driven framework as well as Data driven framework. It is an easy to use framework which allows manual testers to create test cases by just looking at the keywords, test data and object repository without coding in the 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.

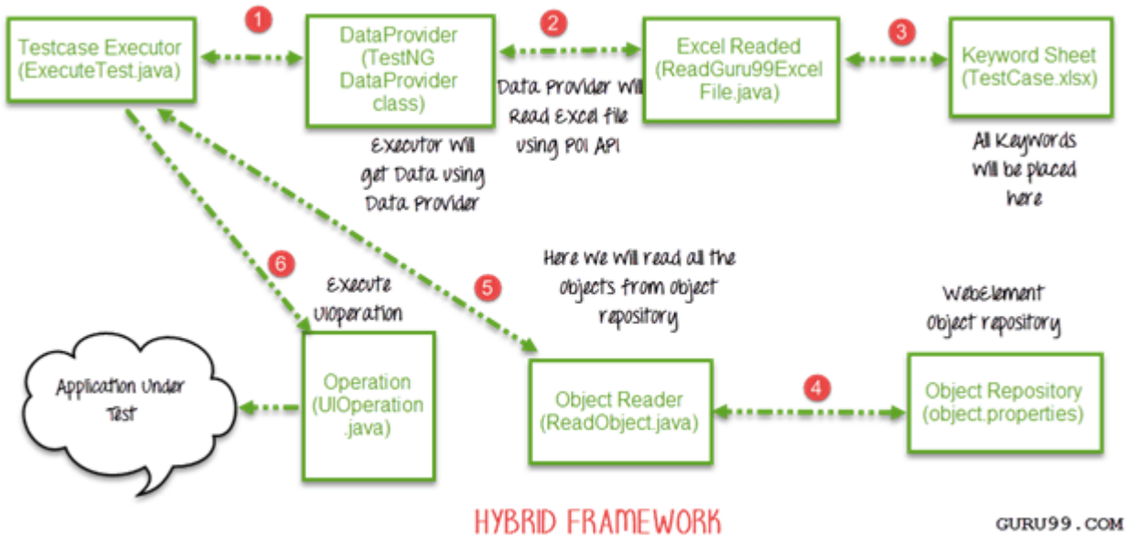


Here in our hybrid framework, we don't need to change anything in Keyword driven framework, here we just need to replace ExecuteTest.java file with HybridExecuteTest.java file.



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

The complete pictorial representation of hybrid framework will look like



HybridExecuteTest.java

```
package testCases;
import java.io.IOException;
import java.util.Properties;
import operation.ReadObject;
import operation.UIOperation;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;
import excelExportAndFileIO.ReadGuru99ExcelFile;
public class HybridExecuteTest {
    WebDriver webdriver = null;
@Test(dataProvider="hybridData")
    public void testLogin(String testcaseName,String keyword,String
objectName,String objectType,String value) throws Exception {
        // TODO Auto-generated method stub

    if(testcaseName!=null&&testcaseName.length()!=0){
```

```
    webdriver=new FirefoxDriver();
    }
ReadObject object = new ReadObject();
Properties allObjects = object.getObjectRepository();
UIOperation operation = new UIOperation(webdriver);
    //Call perform function to perform operation on UI
            operation.perform(allObjects, keyword, objectName,
                objectType, value);

    }
@DataProvider(name="hybridData")
    public Object[][] getDataFromDataprovider() throws IOException{
    Object[][] object = null;
    ReadGuru99ExcelFile file = new ReadGuru99ExcelFile();
//Read keyword sheet
Sheet guru99Sheet =
file.readExcel(System.getProperty("user.dir")+"\\","TestCase.xlsx" ,
"KeywordFramework");
//Find number of rows in excel file
    int rowCount = guru99Sheet.getLastRowNum()-guru99Sheet.getFirstRowNum();
    object = new Object[rowCount][5];
    for (int i = 0; i < rowCount; i++) {
        //Loop over all the rows
        Row row = guru99Sheet.getRow(i+1);
        //Create a loop to print cell values in a row
        for (int j = 0; j < row.getLastCellNum(); j++) {
            //Print excel data in console
            object[i][j] = row.getCell(j).toString();
        }
    }
    System.out.println("");
     return object;
    }
}
```

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.
- The hybrid framework is a mix of keyword driven and data driven framework.

Download the Selenium Project Files for the Demo in this Tutorial

You Might Like:

- How to use Selenium IDE with Scripts & Commands (Assert, Verify)
- First Selenium Webdriver Script: JAVA Sample Code Example
- XPath Contains, Following Sibling, Ancestor & Selenium AND/OR

- XPath Contains, Following Sibling, Ancestor & Selenium AND/OR
- Database Testing using Selenium: Step by Step Guide
- TestNG @Test Priority in Selenium

Prev Report a Bug Next