**Keyword driven automation testing framework** is the most popular testing framework that is easy to design in selenium.

It is a technique in which we represent the test scenario in terms of keywords and corresponding parameters.

A keyword-driven framework in Selenium is a collection of keywords that are used to write test scripts step-wise in the form of table and functions are called based on keywords to complete an end-to-end flow.

Each keyword represents a user action or function that we would like to test in the application.

We write keywords one after another in the table (generally in an excel sheet or database) in a simple English style to automate a test case.

For each keyword, we write a particular code that will be executed when the specified keyword is called to perform a particular action.

Look at the block diagram of keyword driven framework design that uses keyword library to perform a particular action of application.

In the keyword-driven automation framework, we mainly focus on actions/functions, not on test data. Therefore, if you want to test multiple functionalities of application under test, you can go for the best option keyword driven framework in Selenium.

This framework is a type of functional automation testing framework which is also known as table-driven testing or action based testing because a sequence of test steps and their corresponding inputs are implemented by using a set of keywords in tabular format.

In the keyword driven framework, we keep separate all the operations and instructions to be performed from the actual test case.

**Main Components of Keyword Driven Framework in Selenium**

There are some common components for each keyword driven automation framework that you need in order to create a Selenium keyword driven framework. They are as follows:

**1. Excel Sheet:** It is used to store keywords and data required for test cases, test steps, test objects, and test actions

**2. Object Repository:** These are property files that are used to store locator values of web elements present in the application.

A property file is a text file where data is stored in the form of key-value pairs. These property files act as an object repository in Selenium WebDriver.

**3. Function Library:** Functions library plays an important role that is used to perform actions. These functions that we create, should be tied up with keywords.

**4. Test Data Sheet (Generally in Excel format):** This is an excel file that is used to store data values within objects to perform actions on them.

**5. Test Scripts:** Test scripts perform operations on AUT (Application under test) with the help of Library functions and Test Data.

**6. Driver Script:** Driver script is the heart of the keyword driven framework in selenium. This script is also known as execution engine.

This is the main script that interacts with all test scripts. The driver script does so with the help of Library functions.

The main task that is accomplished by driver script, is to read data from environment variables, read excel file, call functions mapped to keywords, and log the result.

**Test Data Sheet in Keyword Driven Framework in Selenium**

List of common columns in Data Sheet are as follows:

**1. Teststeps:** Name of the test Cases/Scenario

**2. LocatorType:** Name of locators like XPath, CSS id, name, etc

**3. Selector:**  Name of the selector for web UI Element.

**4. Action:** Name of the action that has to be performed on any Object such as open browser, enter username, enter password, click, etc.

**5. Value:** Actual test data that has to be entered into objects.

In the above sample data sheet, we have created one test case which contains various keywords that represent the functionality of an application in step by step.

**Why do we perform Keyword Driven Testing ?**

There are the following reasons to perform keyword-driven testing. They are as follows:

1. Keyword-driven framework helps to separate test scrip and test data individually which minimizes script modification and maintenance efforts.

2. By separating it, a non-technical person or manual tester can also easily understand automation process and write automation script to automate the functionality of the application.

3. Test execution can be performed automatically by non-technical person or manual tester based on the keyword lists.

4. Using this approach, we can make all the common components files individually which help to handle easily and increase the reusability of common components.

5. In keyword driven testing approach, the bug report and its graphical representation are generated automatically which reduces the effort and time of testers.

So, these are some common reasons for performing keyword driven testing framework.

**Steps to Set up Keyword Driven Framework in Selenium**

Let’s create a simple design of keyword driven framework with the help of a scenario. In this scenario, we will automate an end-to-end flow of a web application.

**Scenario to Automate:**

1. Open a Browser.

2. Navigate to URL “https://opensource-demo.orangehrmlive.com/web/index.php/auth/login“.

3. Enter username

4. Enter password

5. Click on the login button

6. Close the browser

Now, follow all the below steps to design a simple keyword driven framework:

**Step1:** To design a keyword-driven framework, the first step is to identify all the actions that have to be performed for test automation of an application. In the above scenario, there are eight actions that have to be performed.

**Step 2:** Once all actions are identified, the next step is to create a Keyword map table in the excel sheet. Keyword map table is a table in an excel sheet that defines all the keywords available for test automation project.

Table shows keyword map table for the above scenario:

As you can observe that action keywords have been defined one after another in a table to automate a test case. The table so formed is called step table. Now, save this excel sheet as “ExcelSheet” on your computer.

**Step 3:** Once all keywords have been identified, the next step is to place the excel sheet in a package of your project. Create a package named ‘excelData’ by right click on the Project. For this, go to New > Package.

Place newly created excel file in the package directory locally on your computer. Now, refresh your project folder in Eclipse. The package structure of the keyword-driven framework can be seen in the below screenshot.

It is always recommended to use this structure, as it is easy to understand, use, and maintain.

**Step 5:** Next step is to write a code for each action keyword. Create a new package named “keywordDriven”. Create a new class file named “ActionKeywords”.

using KeywordDrivenFramework.KeywordDriven;

using NPOI.SS.UserModel;

using NPOI.Util;

using NPOI.XSSF.UserModel;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace KeywordDrivenFramework.ExcelUtility

{

public class ReadExcel

{

//XSSF

public void ReadData()

{

string path = @"D:\Trupti\DemoFile.xlsx";

FileStream fs = null;

try

{

fs = new FileStream(path, FileMode.Open, FileAccess.Read);

}

catch(FileNotFoundException e)

{

Console.WriteLine(e.StackTrace);

}

XSSFWorkbook wb = new XSSFWorkbook(fs);

XSSFSheet sh = (XSSFSheet)(wb.GetSheet("DemoData"));// get sheet from workbook

DataEngine engine = new DataEngine();

for(int i=1; i<= sh.LastRowNum; i++) {

IRow row = sh.GetRow(i);// single row

string teststeps = row.GetCell(0).StringCellValue.Trim();

string locatorType = row.GetCell(1).StringCellValue.Trim();

string selector = row.GetCell(2).StringCellValue.Trim();

string action = row.GetCell(3).StringCellValue.Trim();

string value = row.GetCell(4).StringCellValue.Trim();

Console.WriteLine(teststeps+ ":"+locatorType+":" +selector+":"+ action+":" + value);

engine.execute(teststeps,locatorType,selector,action,value);

}

}

}

}

**Step 6:** In this step, we have to write code to read data from excel sheet. For this, we use NPO library which allows us to read, create and edit Microsoft Office documents using Java.

Create a new package named “excelUtility” and then create a new class file named “ReadExcelSheet”. Look at the following source code to read all keywords from the step table.

**ReadExcelSheet class code:**

using KeywordDrivenFramework.KeywordDriven;

using NPOI.SS.UserModel;

using NPOI.Util;

using NPOI.XSSF.UserModel;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace KeywordDrivenFramework.ExcelUtility

{

public class ReadExcel

{

//XSSF

public void ReadData()

{

string path = @"D:\Trupti\DemoFile.xlsx";

FileStream fs = null;

try

{

fs = new FileStream(path, FileMode.Open, FileAccess.Read);

}

catch(FileNotFoundException e)

{

Console.WriteLine(e.StackTrace);

}

XSSFWorkbook wb = new XSSFWorkbook(fs);

XSSFSheet sh = (XSSFSheet)(wb.GetSheet("DemoData"));// get sheet from workbook

DataEngine engine = new DataEngine();

for(int i=1; i<= sh.LastRowNum; i++) {

IRow row = sh.GetRow(i);// single row

string teststeps = row.GetCell(0).StringCellValue.Trim();

string locatorType = row.GetCell(1).StringCellValue.Trim();

string selector = row.GetCell(2).StringCellValue.Trim();

string action = row.GetCell(3).StringCellValue.Trim();

string value = row.GetCell(4).StringCellValue.Trim();

Console.WriteLine(teststeps+ ":"+locatorType+":" +selector+":"+ action+":" + value);

engine.execute(teststeps,locatorType,selector,action,value);

}

}

}

}

**Advantage of Keyword Driven Framework**

Some of the advantages of keyword-driven framework are as follows:

1. We don’t need expertise in automation to maintain or create new test scripts. Once the keyword functionalities of application are ready, minimal specific scripting skills are needed for automating test cases.

2. Keywords can be reused across multiple test scripts or even different software.

3. Easy maintenance of test scripts.

4. Keyword-driven framework is not dependent on a specific programming language or tool

5. It is compatible with any automation tools available in the market.

**Disadvantage of Keyword Driven Framework**

Some of the disadvantages of keyword-driven framework are as follows:

1. To create keyword driven automation framework, we need a higher knowledge of scripting knowledge.

2. Test scripts in keyword driven framework are longer and more complex, taking more time to create and maintain.

**Step 7:** Now the next step is to write code for calling readExcelData() method of ReadExcelSheet class and methods of ActionKeywords class one by one.

Create a new package named “executionEngine” and write the following source code by creating a class named “ExecutionTest”.

**ExecutionTest class code:**

using KeywordDrivenFramework.ExcelUtility;

using KeywordDrivenFramework.KeywordDriven;

using NUnit.Framework;

namespace KeywordDrivenFramework.TestExecution

{

public class TestExecution

{

[Test]

public void execution()

{

DataEngine engine = new DataEngine();

ReadExcel re = new ReadExcel();

re.ReadData();

}

}

}