20-03-2017

FRAMEWORK EXECUTION:

To execute the framework run all test cases. We use suite file. To create it Right click on the java project >> testNG >> Select convert to testNG and click finish. It will generate testNG.xml >> Right click on xml file >> Go to Run as >> Select testNG suite >> which executes all the scripts present in framework.

TestNG.xml Content:

```
<suite name = "Suite" parallel = "none" >
<test name = "Test" >
<classes>
    <class name = "script.validLoginLogout"/>
    <class name = "script.InvalidLogin"/>
    <class name = "script.VerifyProductVersion"/>
</classes>
</test>
</suite>
```

After the execution refresh the java project, expand 'test-output' folder; Right click on emailable-report.html >> open with webbrowser

>> Right click on the page & select to export to MS Excel.

>> To convert Html report into excel report.

AUTOMATING FUNCTIONAL TESTCASES:

1) We automate all type of testcases, which are part of Regression such as smoke testing, functional integration system, valid scenarios, invalid scenarios etc.
2) While automating functional test cases, we need to take multiple inputs to perform thorough testing.
3) The test data will be provided by manual testing team & it may not be in the required format. The job of automation engineer is format the data suitable for executing automation.
4) If testdata is not ready, automation team should prepare using test case design techniques. (BVA, EC, EP)

5) To store the data we use Excel file.

6) In Selenium, there is no option to handle excel file, Hence we use 3rd party tool such as apache poi

It can be downloaded from:-

URL: https://poi.apache.org/download.html

File: poi-bin-3.15-20160924.zip

Unzip: poi-3.15

POI → poor obfuscation implentation.

Unzip folder 'poi-3.15' contains 13 files & copy paste all of them to jars folder of the framework & add all of them to build path.

To store the excel file in the framework, create a folder with the name "data" inside java project folder.

To view the extension of any, go to start → folder → select folder options → View Tab → Uncheck the checkbox (Hide Extensions for known file types) → Click OK.

Go to the location where newly created folder (data) is present (D:\BCSM13\actiTIME\data)

Right click Go to → new select → microsoft office Excel.

Specify the name as 'input'. Its extension will be "xlsx"

Open the file & open the sheet 1 and enter data in first cell of sheet1 → Save & close the file.

In eclipse refresh the java project. So that excel file will be visible.

Reading data from cell of an excel sheet

```
public class Excel {
    P S V M (String[] args) throws Exception {
  // open workbook (xl file)
    String xlpath = "./data/input.xlsx";
  FileInputStream fis = new FileInputStream(xlpath);
  Workbook wb = WorkbookFactory.create(fis)
  //go to sheet1
    sheet s = wb.getsheet("Sheet1");
  // go to Row 0
     Row r = s.getRow(0);
  // go to cell 0
     Cell c = r.getCell(0);
  // get cell data
    String v = c.getStringCellValue();
  // print
    s.o.p(v); }
}
```

Optimized:

```
FileInputStream fis = new FileInputstream("./data/input.xlsx");
Workbook wb = WorkBookFactory.create(fis);
String v = wb.getSheet("Sheet1").getRow(0).getCell(0).toString();
s.o.p(v)
```

In cell class toString() method is overridden which return address instead of value.

Writing Data into Excel Sheet

```
//open workbook
FileInputStream fis = new FileInputStream("./data/input.xlsx");
Workbook wb = WorkBookFactory.create(fis);
//write data
Cell c = wb.getSheet("Sheet1").getRow(0).getCell(0);
c.setCellValue("java");
//save as
FileOutputStream fos = new FileOutputStream("./data/input2.xlsx");
wb.write(fos);
```

21-03-2017

Write a code to print the content of excel sheet which has the data in 3 rows & 3 columns

```
public static void main(String[] args) throws Exception {
  FileInputStream fis = new FileInputStream("./data/Book1.xlsx");
  Workbook wb = WorkbookFactory.create(fis);
  for(int i=0; i<=2; i++){
    for(int j=0; j<=2; j++){
      Cell c = wb.getSheet("Sheet1").getRow(i).getCell(j);
      System.out.print(c+" ");
    }
    System.out.println();}
}
```

o/p

A1 B1 C1
A2 B2 C2
A3 B3 C3

The above code only works if no of rows & columns is 3.

7.9 ⁂ Write a code to print content of excel sheet where no of rows and columns are dynamically changing.

```
public static void main(String[] args) throws Exception {
FileInputStream fis = new FileInputStream("./data/book1.xlsx");
Workbook wb = WorkbookFactory.create(fis);
int rc = wb.getSheet("Sheet1").getLastRowNum();
  for (int i=0; i<=rc; i++) {
    int cc = wb.getSheet("Sheet1").getRow(i).getLastRowNum();
  for (int j=0; j<cc; j++) {
  Cell c = wb.getSheet("Sheet1").getRow(i).getCell(j);
    System.out.Print(c+" ");
  }
  System.out.Println(); }
}
```

Using Excel in Framework:

To handle the excel file in the framework, we develop generic methods as shown below.

```
package generic;

import java.io.FileInputStream;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;
```

```
public class Excel {
    Public static String getCellValue(String path, String sheet,
        int r, int c) {
    String v="";
     try {
         FileInputStream fis = new FileInputStream(path);
         Workbook wb = WorkbookFactory.create(fis);
         v = wb.getSheet(sheet).getRow(r).getCell(c).toString();
       }
    catch (Exception e){
     }
    return v;
}

public static int getRowCount(String path, String sheet){
  int rc=0;
     try{
         FileInputStream fis = new FileInputStream(path);
       Workbook wb = WorkbookFactory.create(fis);
        rc = wb.getSheet(sheet).getLastRowNum();
       }
  catch(Exception e){
    }
  return rc;}
}
```

Steps to get data from Excel Sheet:-

Go to input.xlsx file present in data folder & create a sheet with the name same as test class name. Enter the data as shown below and save & close.

UserName	PassWord	HomeTitle	LoginTitle
admin	manager	actiTIME - Enter Time-Track	actiTIME - Login

validLoginLogout ⊕

Update testClass as shown below:

```
public class ValidLoginLogout extends BaseTest {
@Test
public void testValidLoginLogout() {
String UN = Excel.getCellValue(INPUT_PATH, "ValidLoginLogout",
                               1, 0);
String PW = Excel.getCellValue(INPUT_PATH, "ValidLoginLogout",
                               1, 1);
String HomeTitle = Excel.getCellValue(INPUT_PATH, "ValidLoginLogout",
                               1, 2);
String loginTitle = Excel.getCellValue(INPUT_PATH, "ValidLoginLogout",
                               1, 3);
LoginPage l = new LoginPage(driver);
l.setUsername(UN);
l.setPassword(PW);
l.clickLogin();
EnterTimeTrackPage e = new EnterTimeTrackPage(driver);
e.verifyTitle(homeTitle);
```

e.clicklogout();
k.verifyTitle(loginTitle); }
}

In the above code, we are using constant INPUT_PATH. declared inside the interface AutoConst as shown below

⇒ String INPUT_PATH = "./data/input.xlsx";

TestData for InvalidLogin:

UserName	Password
adam	damager
admin	damager
-	-
-	manager
admin	-

ValidloginLogout | InValidLogin | ⊕

public class Invalidlogin extends BaseTest {
@Test
public void testInvalidlogin() {
int rc = Excel.getRowCount(INPUT_PATH, "InvalidLogin");
for(int i=1; i<=rc; i++){
String un= Excel.getCellValue(INPUTPATH, "InvalidLogin", i, 0);
String pw= Excel.getCellValue(INPUT_PATH, "InvalidLogin", i, 1);
LoginPage l= new LoginPage(driver);

```
l.setUsesName(un);
l.setPassWord(pw);
l.clickLogin();
l.verifyErrMsgIsPresent();}
 }
}
```