Framework

Class 3

Agenda

Data Driven Framework

Excel File in Java

What is Data Driven Framework

Data Driven Framework is focused on **separating the test scripts** logic and the **test data** from each other.

The test data set is kept in the external files **Excel Files, CSV Files, Database**) and are loaded into the variables inside the Test Script. Variables are used both for **Input values and for Verification values.**

The test scripts connect to the external resources to get the test data.

In Data Driven Framework Single Test case iterate over the Test Data.

For example, We have Login test Case and we want to test it with Combination of Username and Password then, in this case, we will keep our Test Data in some external file and read this data and then pass it to our Test Script.

Data driven framework using Excel files

Selenium support only Web browser automation so for Read and Write excel files we use the Apache POI library, which allows us to read, create and edit Microsoft Office-documents using Java.

Apache POI is an open source java library to create and manipulate various file formats based on Microsoft Office (doc, docx, ppt, pptx, xls, xlsx)

Apache POI is mostly used for reading and writing excel documents.It has many predefined methods, classes, and interfaces that we can use to read and write into Excel files

Reading Excel file

```
String xlFile="./TestData/OrangeHRMData.xlsx";
// create file input stream object and load file
FileInputStream fis =new FileInputStream(xlFile);
// create object for workbook and load file
XSSFWorkbook workbook = new XSSFWorkbook(fis);
// get the sheet which you want to modify or create
XSSFSheet sheet = workbook.getSheetAt(0);
//create object for row
XSSFRow row = sheet.getRow(0);
//create object for cell
XSSFCell cell = row.getCell(0);
//get and print the value from specific cell
String value = cell.getStringCellValue();
System.out.println(value);
```

Reading Excel file

```java
@Test
public void readExcel() throws IOException {
    //specify xl file path
    String xlPath="src/test/resources/testdata/OrangeHrmData.xlsx";
    //open stream and pass file
    FileInputStream fis=new FileInputStream(xlPath);
    //open workbook
    XSSFWorkbook workbook=new XSSFWorkbook(fis);
    //access worksheet;
    XSSFSheet sheet=workbook.getSheet("EmployeesData");
    //access specified data
    String cellData=sheet.getRow(1).getCell(1).toString();
    System.out.println(cellData);
    //get number of rows
    int rows=sheet.getPhysicalNumberOfRows();
    System.out.println("Number of rows= "+rows);
    //get number of cols
    int cols=sheet.getRow(0).getLastCellNum();
    System.out.println("Number of cols= "+cols);
    //get all data
    String data;
    for (int i=0; i<rows; i++) {
        for (int y=0; y<cols; y++) {
            data=sheet.getRow(i).getCell(y).toString();
            System.out.println(data);
        }
    }
    workbook.close();
    fis.close();
}
```

Writing to Excel file

```
String xlFile="./TestData/OrangeHRMData.xlsx";
    FileInputStream fis =new FileInputStream(xlFile);
    // create object for workbook and load file
    XSSFWorkbook workbook = new XSSFWorkbook(fis);
    // get the sheet which you want to modify or create
    XSSFSheet sheet = workbook.getSheetAt(0);
    //create 3 row and create 1 column
    sheet.createRow(2).createCell(0).setCellValue("user");
    //get 3 row and create 2 column
    sheet.getRow(2).createCell(1).setCellValue("user123");
    // create file output stream object and load file
    FileOutputStream fos=new FileOutputStream(xlFile);
    //write content
    workbook.write(fos);
    //close the stream
    fos.close();
```

Writing to Excel file

```
@Test
public void writeToExcel() throws IOException {
    // specify xl file path
    String xlPath = "src/test/resources/testdata/OrangeHrmData.xlsx";
    // open stream and pass file
    FileInputStream fis = new FileInputStream(xlPath);
    // open workbook
    XSSFWorkbook workbook = new XSSFWorkbook(fis);
    // access worksheet;
    XSSFSheet sheet = workbook.getSheet("EmployeeDetails");
    // get all rows
    int rows = sheet.getPhysicalNumberOfRows();
    // get all cols
    int cols = sheet.getRow(0).getLastCellNum();
    // create cell
    sheet.getRow(0).createCell(cols).setCellValue("Result");
    sheet.getRow(1).createCell(cols).setCellValue("Pass");
    //write to Excel
    FileOutputStream fos = new FileOutputStream(xlPath);
    workbook.write(fos);
    //close workbook and stream
    fos.close();
    workbook.close();
}
```

Test Case

TC 1: Add Location

1. Login to OrangeHRM
2. From Admin --> Organization select Locations
3. Add 5 different locations by providing complete required details
4. Save location and verify it has been successfully saved.

Note: store data in excel

Creating Xl Utility

```java
public class ExcelUtility {
    public FileInputStream fis;
    public XSSFWorkbook workbook;
    public XSSFSheet sheet;

    public void openWorkbook(String filePath) {
        try {
            fis=new FileInputStream(filePath);
            workbook=new XSSFWorkbook(fis);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void openSheet( String sheetName) {
        sheet=workbook.getSheet(sheetName);
    }

    public String getCellData(int row, int cell) {
        String cellValue = sheet.getRow(row).getCell(cell).toString();
        return cellValue;
    }

    public int rowCount() {
        return sheet.getPhysicalNumberOfRows();
    }

    public int colsCount() {
        return sheet.getRow(0).getLastCellNum();
    }
}
```

Xl with DataProvider

```java
@Test(dataProvider = "login credentials")
public void negativeLogin(String uname, String pwd) throws IOException {
    LoginPage login = new LoginPage();
    login.login(uname, pwd);
    String errorText = login.message.getText();
    Assert.assertEquals(errorText, "Invalid Credentials");
}

@DataProvider(name = "login credentials")
public Object[][] getData() {
    ExcelUtility excel=new ExcelUtility();
    excel.openWorkbook(Constants.TESTDATA_PATH);
    excel.openSheet("LoginCredentials");
    int rows = excel.rowCount();
    int cols = excel.colsCount();

    System.out.println(rows);
    System.out.println(cols);
    Object[][] data = new Object[rows - 1][cols];

    for (int i = 1; i < rows; i++) {
        for (int y = 0; y < cols; y++) {
            String cellValue = excel.getCellData(i, y);
            data[i - 1][y] = cellValue;
            System.out.println(i - 1);
        }
    }
    return data;
}
```

Listeners in TestNG

- Listener is defined as interface that modifies the default TestNG behavior. Listeners implement the interface “org.testng.ITestListener”.
- Listeners "listen" to the event defined in the selenium script and behave accordingly.
- It is used in selenium by implementing Listeners Interface.
- It allows customizing TestNG reports or logs. There are many types of TestNG listeners available.
- **Listeners provide us the flexibility to change and modify the TestNG default behavior.**
- TestNG manages everything through Suite, Test and Methods and the Listeners gives us the ability to act before and after of every Suite, Test and Methods.
-

Types of Listeners in TestNG

- IAnnotationTransformer ,
- IAnnotationTransformer2 ,
- IConfigurable ,
- IConfigurationListener ,
- IExecutionListener,
- IHookable ,
- IInvokedMethodListener ,
- IInvokedMethodListener2 ,
- IMethodInterceptor ,
- IReporter,
- ISuiteListener,
- ITestListener .

Methods of ITestListener

onStart() – This Invoked after the Test Class is instantiated but before any test method is called.

onFinish() – This invoked after all methods done and before the class finishes execution.

onTestStart() – Invoked each time before a test will be invoked.

onTestFailure() – Invoked each time a test fails.

onTestSkipped() – Invoked each time a test is skipped.

onTestSuccess() – Invoked each time a test succeeds.

Listeners in TestNG

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Smoke Suite">

    <groups>
        <run>
            <include name="smoke"></include>
        </run>
    </groups>

    <listeners>
        <listener class-name="com.syntax.utils.Listeners" />
    </listeners>


    <test thread-count="5" name="Smoke Test">
        <packages>
            <package name="com.syntax.testcases" />
        </packages>

    </test> <!-- Test -->
</suite> <!-- Suite -->
```