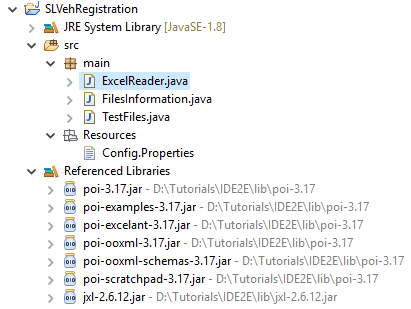
**IDENTITY E2E – Test 4 - Java Exercise**

**Part 1: Write a Service layer bean to do the following:**

I have created a Java project for it in Eclipse named “**SLVehRegistration**” structure of the project is as shown below:



1. Scan configured directory in file system which will return this information --> filename, file mime type, file size, file extension

**FileInformation.Java**

**package** main;

**import** java.io.File;

**public** **class** FilesInformation {

/\*

\* Function: getFilesInfo

\* I/P Parameter: Folder location to start searching file

\* O/P Parameter: List of exists

\*/

**public** File[] getFilesInfo(String filePath) {

//String filePath = System.getProperty("user.dir") + "\\Data";

//System.out.println(filePath);

File folder = **new** File(filePath);

File[] listOfFiles = folder.listFiles();

**return** listOfFiles;

}

}

**TestFile.Java**

package main;

import java.io.File;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.util.Properties;

import javax.activation.MimetypesFileTypeMap;  
import jxl.read.biff.BiffException;

public class TestFiles {

public static void main(String[] args) throws IOException, BiffException {

String path = System.getProperty("user.dir") + "\\src\\Resources\\Config.Properties";

Properties config = new Properties();

FileInputStream fis = new FileInputStream(path);

config.load(fis);

FilesInformation fi = new FilesInformation();

File[] listOfFiles = fi.getFilesInfo(config.getProperty("filelocation"));

for (int i = 0; i < listOfFiles.length; i++) {

if (listOfFiles[i].isFile()) {

System.out.println("File :" + i);

System.out.println("\tFile name: " + listOfFiles[i].getName());

System.out.println("\tFile mime type: " + new MimetypesFileTypeMap().getContentType(listOfFiles[i]));

System.out.println("\tFile size: " + listOfFiles[i].length());

System.out.println("\tFile extension: " + listOfFiles[i].getAbsolutePath());

}

}

String vrFile = config.getProperty("VehRegFile");  
 String sheetName = config.getProperty("sheetName");

ExcelReader DT = new ExcelReader();  
 System.out.println(DT.returnRowCount(vrFile, sheetName));

for (int j = 1; j < 3; j++) {  
 for (int k = 0; k < 3; k++) {  
 System.out.println(DT.returnCellData(vrFile, sheetName, j, k));  
 }

}

// DT.readExcel(vrFile, sheetName);

}

}

**Config.Properties:** stored all configuration parameters

filelocation=D:\\Tutorials\\IDE2E\\SLVehRegistration\\Data

VehRegFile=D:\\Tutorials\\IDE2E\\SLVehRegistration\\Data\\VR.xls

sheetName=RegDetails

Output:

File :0

File name: ImgGif.gif

File mime type: image/gif

File size: 2340

File extension: D:\Tutorials\IDE2E\SLVehRegistration\Data\ImgGif.gif

File :1

File name: ImgJpg.jpg

File mime type: image/jpeg

File size: 4458

File extension: D:\Tutorials\IDE2E\SLVehRegistration\Data\ImgJpg.jpg

File :2

File name: ImgPng.png

File mime type: application/octet-stream

File size: 1238

File extension: D:\Tutorials\IDE2E\SLVehRegistration\Data\ImgPng.png

File :3

File name: TestPpt.pptx

File mime type: application/octet-stream

File size: 29852

File extension: D:\Tutorials\IDE2E\SLVehRegistration\Data\TestPpt.pptx

File :4

File name: TestVR.txt

File mime type: text/plain

File size: 14

File extension: D:\Tutorials\IDE2E\SLVehRegistration\Data\TestVR.txt

File :5

File name: VR.csv

File mime type: application/octet-stream

File size: 2

File extension: D:\Tutorials\IDE2E\SLVehRegistration\Data\VR.csv

File :6

File name: VR.xls

File mime type: application/octet-stream

File size: 22528

File extension: D:\Tutorials\IDE2E\SLVehRegistration\Data\VR.xls

File :7

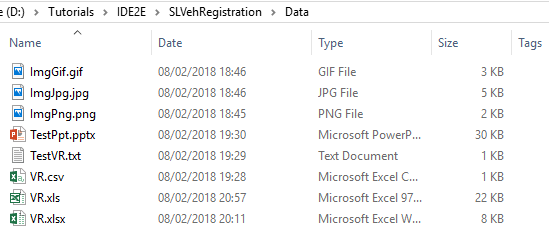
File name: VR.xlsx

File mime type: application/octet-stream

File size: 7939

File extension: D:\Tutorials\IDE2E\SLVehRegistration\Data\VR.xlsx

1. Use a directory containing a reasonably large number of files, minimum 10.



 3.      Provide a way to retrieve certain supported mime type files: configure excel and csv are supported currently

I have created customised ExcelReader class for it.

**package** main;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**import** jxl.Sheet;

**import** jxl.Workbook;

**import** jxl.read.biff.BiffException;

**public** **class** ExcelReader {

**public** String path;

**public** **int** rowCount=0;

**public** **void** readExcel(String filePath, String sheetName) **throws** BiffException, IOException {

// String FilePath = "D:\\Tutorials\\IDE2E\\SLVehRegistration\\Data\\VR.xls";

path = filePath;

FileInputStream fs = **new** FileInputStream(path);

Workbook wb = Workbook.*getWorkbook*(fs);

// TO get the access to the sheet

Sheet sh = wb.getSheet(sheetName);

// To get the number of rows present in sheet

**int** totalNoOfRows = sh.getRows();

// To get the number of columns present in sheet

**int** totalNoOfCols = sh.getColumns();

**for** (**int** row = 1; row < totalNoOfRows; row++) {

**for** (**int** col = 0; col < totalNoOfCols; col++) {

System.***out***.print(sh.getCell(col, row).getContents() + "\n");

}

System.***out***.println();

}

}

**public** **int** returnRowCount(String filePath, String sheetName) **throws** BiffException, IOException {

path = filePath;

FileInputStream fs = **new** FileInputStream(path);

Workbook wb = Workbook.*getWorkbook*(fs);

// TO get the access to the sheet

Sheet sh = wb.getSheet(sheetName);

// To get the number of rows present in sheet

**this**.rowCount = sh.getRows();

**return** rowCount;

}

/\*

\* Function: returnCellData

\* Input Parameters: filePath, sheetName, rowNo (Row number), columnNo (Column Number)

\* Return Type: Cell Data

\*

\*

\*/

**public** String returnCellData(String filePath, String sheetName, **int** rowNo, **int** columnNo) **throws** BiffException, IOException {

path = filePath;

FileInputStream fs = **new** FileInputStream(path);

Workbook wb = Workbook.*getWorkbook*(fs);

// TO get the access to the sheet

Sheet sh = wb.getSheet(sheetName);

// To get the number of rows present in sheet

rowCount = sh.getRows();

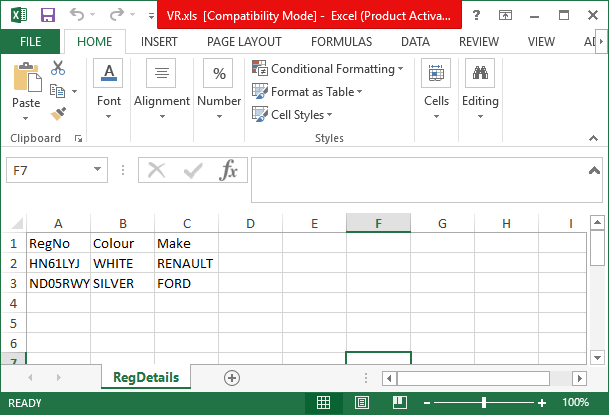
String cellData = sh.getCell(columnNo, rowNo).getContents();

**return** cellData;

}

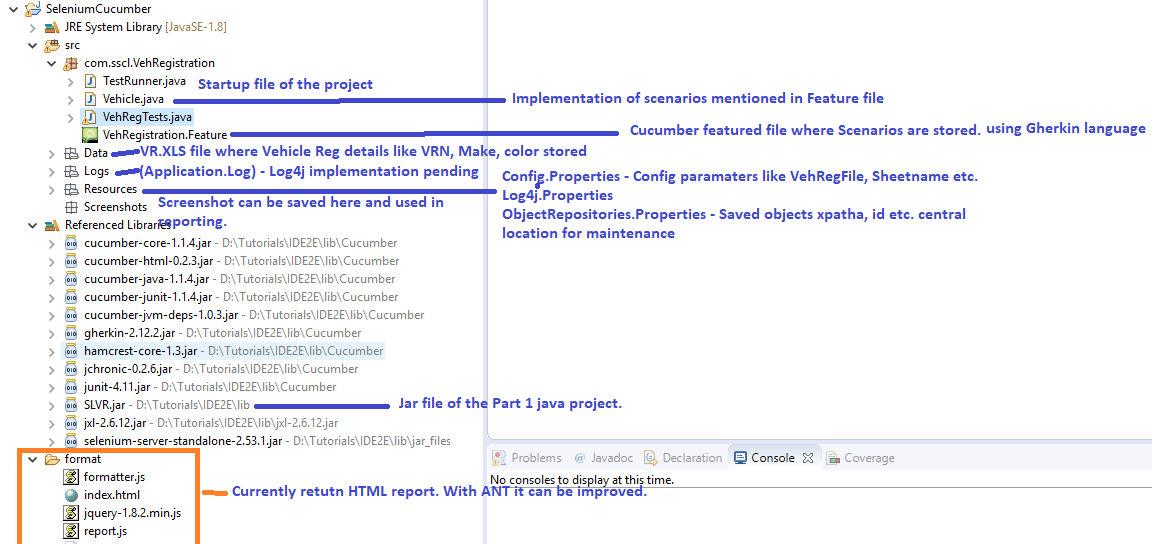
}

Limitation: The ExcelReader has limited functionality and read data from .XLS file only.



Part 2: Write a selenium/cucumber framework to do the following:

I have design Selenium Cucumber framework with Junit for Part 2.



1. Use the above service layer bean to get supported files (excel or csv are supported, from input directory)

I have exported Part 1 project as .jar file and imported into Part 2 as an external Jar file.

**TestRunner. Java:**

package com.sscl.VehRegistration;

import org.junit.runner.RunWith;  
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)  
@Cucumber.Options(format = {"html:format"},

features = "D:\\Tutorials\\IDE2E\\SeleniumCucumber\\src\\com\\sscl\\VehRegistration\\VehRegistration.feature")

public class TestRunner {

}

1. Go through the file and read vehicle registration details in the file.

// In order to pass data between implementation scenario, I have created Vehicle class.

ArrayList<Vehicle> veh = **new** ArrayList<Vehicle>();

**public** **static** WebDriver *driver*;

**public** **static** Logger *log* = Logger.*getLogger*("devpinoyLogger"); // Implementation for customised logs

@Given("I retrieve list of VRNs from excel sheet")

**public** **void** I\_retrieve\_list\_of\_VRNs\_from\_excel\_sheet() **throws** BiffException, IOException {

*log*.info("I retrieve list of VRNs from excel sheet");

String path = System.*getProperty*("user.dir") + "\\src\\Resources\\Config.Properties";

Properties config = **new** Properties();

FileInputStream fis = **new** FileInputStream(path);

config.load(fis);

String vrFile = config.getProperty("VehRegFile");

// System.out.println(vrFile);

String sheetName = config.getProperty("sheetName");

// System.out.println(sheetName);

ExcelReader DT = **new** ExcelReader();

// int rows = DT.returnRowCount(vrFile, sheetName);

String vrn = "";

String make = "";

String color = "";

**for** (**int** j = 1; j < 3; j++) {

vrn = DT.returnCellData(vrFile, sheetName, j, 0);

color = DT.returnCellData(vrFile, sheetName, j, 1);

make = DT.returnCellData(vrFile, sheetName, j, 2);

veh.add(**new** Vehicle(vrn, make, color));

}

}

1. Open webpage : <https://www.gov.uk/get-vehicle-information-from-dvla> and go through all vehicles from excel/csv file.

@And("I open DVLA site from internet")

**public** **void** I\_open\_DVLA\_site\_from\_internet() **throws** IOException {

*log*.info("I open DVLA site from internet");

// Log4j logger implementation pending

String path = System.*getProperty*("user.dir") + "\\src\\Resources\\ObjectRepository.Properties";

Properties config = **new** Properties();

FileInputStream fis = **new** FileInputStream(path);

config.load(fis);

**if** (config.getProperty("browser").equals("firefox")){

*driver* = **new** FirefoxDriver();

} **else** **if** (config.getProperty("browser").equals("chrome")){

System.*setProperty*("webdriver.chrome.driver", "D:\\Tutorials\\IDE2E\\lib\\chromedriver.exe");

*driver* = **new** ChromeDriver();

}

*driver*.get(config.getProperty("url"));

*driver*.manage().window().maximize();

*driver*.manage().timeouts().implicitlyWait(10, TimeUnit.***SECONDS***);

}

 4.                  On the Vehicle details page assert the details (Make/Color) match with expected output in excel/csv file.

@When("I search VRN on DVLA")

**public** **void** I\_search\_VRN\_on\_DVLA() **throws** IOException {

String path = System.*getProperty*("user.dir") + "\\src\\Resources\\ObjectRepository.Properties";

Properties config = **new** Properties();

FileInputStream fis = **new** FileInputStream(path);

config.load(fis);

**if** (veh.size() > 0) {

System.***out***.println("size:" + veh.size());

**for** (**int** i = 0; i < veh.size(); i++) {

//System.out.println("\n-----");

//System.out.print(veh.get(i).VRN + "\t");

//System.out.print(veh.get(i).Make + "\t");

//System.out.print(veh.get(i).Colour);

*driver*.findElement(By.*xpath*(config.getProperty("btnStartNow"))).click();

*driver*.findElement(By.*xpath*(config.getProperty("txtVRN"))).sendKeys(veh.get(i).VRN);

*driver*.findElement(By.*xpath*(config.getProperty("btnContinue"))).click();

//verify the make and colour

WebElement makeFound = *driver*.findElement(By.*xpath*(config.getProperty("spanMake")));

System.***out***.println(makeFound.getText());

WebElement colourFound = *driver*.findElement(By.*xpath*(config.getProperty("spanColour")));

System.***out***.println(colourFound.getText());

Assert.*assertEquals*(veh.get(i).Make, makeFound.getText());

Assert.*assertEquals*(veh.get(i).Colour, colourFound.getText());

**if** (veh.get(i).Make == makeFound.getText() && veh.get(i).Colour ==colourFound.getText()) {

System.***out***.println("Vehicle details found correctly.");

} **else** {

// to do: Capture Screenshot

}

// Go back the start page where search Vehicle details for next set of VRN will start

*driver*.navigate().back();

*driver*.navigate().back();

}

}

}

@Then("I get Make and Colour of entered VRN")

**public** **void** I\_get\_Make\_and\_Colour\_of\_entered\_VRN() {

// Always release the memory

**if** (*driver* != **null**) {

*driver*.quit();

}

}

**How the currently designed framework can be improved:**

* I am use Maven where the dependency can be managed using POM.
* Currently the framework generate simple HTML report. With the help of ANT more robust report can be generated.
* If I used TestNG instead of Junit I can have more control of the test run (I have categorised etc.)
* The project can kept in Jeankins for CI.
* The log in very basic. I can use Log4j to keep the log in log file.
* Provide screenshot if case of failure which still under development
* I have used Selenium WebDriver 2.53.1 which was last stable release of WebDriver. Currently 3.9.1 is the latest WebDriver release in market but it still has lots of issue particularly with Firefox.

Due to time constraint and current work load, I’m not able to provide the above functionality. But whatever I have designed is fully working.

**The final output when I run the Selenium Cucumber project with Junit is as shown below:**

