Handling Excel Sheets using Apache POI in Selenium

Introduction

Handling Excel sheets in Selenium can be crucial for data-driven testing. Apache POI is a popular library used to read and write Excel files in Java. Let's explore how to handle Excel sheets using Apache POI in Selenium in five different ways, focusing on the WorkbookFactory and XSSFWorkbook classes to fetch and read data. We'll use the Heroku website as an example and explain each step in detail.

Prerequisites:

1. Add the required dependencies to your pom.xml file for Apache POI and WebDriverManager.

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>5.2.3</version>
</dependency>
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>5.2.3</version>
</dependency>
<dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
```

```
    <version>5.3.2</version>
</dependency>
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.1.2</version>
</dependency>
```

Example 1: Using WorkbookFactory to Read Excel File

1. Using WorkbookFactory to Read Excel File

Steps:

1. Setup WebDriverManager:
2. Open Excel File using WorkbookFactory:
3. Read Data from the Excel File:
4. Perform Selenium Actions using the Data:

```
import org.apache.poi.ss.usermodel.*;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import io.github.bonigarcia.wdm.WebDriverManager;

import java.io.File;
```

Handling Excel Sheets using Apache POI in Selenium

```
import java.io.FileInputStream;
import java.io.IOException;


public class ExcelReaderExample1 {
    public static void main(String[] args) throws IOException {
        // Setup WebDriverManager and initialize ChromeDriver
        WebDriverManager.chromedriver().setup();
        WebDriver driver = new ChromeDriver();

        // Path to the Excel file
        FileInputStream file = new FileInputStream(new File("path/to/excel/file.xlsx"));

        // Load the workbook using WorkbookFactory
        Workbook workbook = WorkbookFactory.create(file);
        Sheet sheet = workbook.getSheetAt(0); // Assuming we are reading the first sheet

        // Iterate through rows and cells
        for (Row row : sheet) {
            for (Cell cell : row) {
                switch (cell.getCellType()) {
                    case STRING:
                        System.out.print(cell.getStringCellValue() + "\t");
                        break;
                    case NUMERIC:
                        System.out.print(cell.getNumericCellValue() + "\t");
```

```
                    break;
                default:
                    break;
            }
        }
        System.out.println();
    }

    // Close the workbook and file stream
    workbook.close();
    file.close();

    // Perform some actions using Selenium WebDriver
    driver.get("https://the-internet.herokuapp.com/");
    // More selenium actions...

    driver.quit();
  }
}
```

Example 2: Using XSSFWorkbook Directly

2. Using XSSFWorkbook Directly

Handling Excel Sheets using Apache POI in Selenium

Steps:

1. Setup WebDriverManager:

2. Open Excel File using XSSFWorkbook:

3. Read Data from the Excel File:

4. Perform Selenium Actions using the Data:

```
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import io.github.bonigarcia.wdm.WebDriverManager;

import java.io.FileInputStream;
import java.io.IOException;

public class ExcelReaderExample2 {
    public static void main(String[] args) throws IOException {
        // Setup WebDriverManager and initialize ChromeDriver
        WebDriverManager.chromedriver().setup();
        WebDriver driver = new ChromeDriver();

        // Path to the Excel file
        FileInputStream file = new FileInputStream("path/to/excel/file.xlsx");

        // Load the workbook using XSSFWorkbook
```

```
        XSSFWorkbook workbook = new XSSFWorkbook(file);
        XSSFSheet sheet = workbook.getSheetAt(0); // Assuming we are reading the first sheet

        // Iterate through rows and cells
        for (int i = 0; i <= sheet.getLastRowNum(); i++) {
            for (int j = 0; j < sheet.getRow(i).getLastCellNum(); j++) {
                System.out.print(sheet.getRow(i).getCell(j).getStringCellValue() + "\t");
            }
            System.out.println();
        }

        // Close the workbook and file stream
        workbook.close();
        file.close();

        // Perform some actions using Selenium WebDriver
        driver.get("https://the-internet.herokuapp.com/");
        // More selenium actions...

        driver.quit();
    }
}
```

Example 3: Reading Specific Data

Handling Excel Sheets using Apache POI in Selenium

3. Reading Specific Data (Example: Login Credentials)

Steps:

1. Setup WebDriverManager:

2. Open Excel File using WorkbookFactory:

3. Read Specific Data from the Excel File:

4. Use Data for Selenium Actions:

```
import org.apache.poi.ss.usermodel.*;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import io.github.bonigarcia.wdm.WebDriverManager;

import java.io.FileInputStream;
import java.io.IOException;

public class ExcelReaderExample3 {
    public static void main(String[] args) throws IOException {
        // Setup WebDriverManager and initialize ChromeDriver
        WebDriverManager.chromedriver().setup();
        WebDriver driver = new ChromeDriver();

        // Path to the Excel file
```

```
        FileInputStream file = new FileInputStream("path/to/excel/file.xlsx");

        // Load the workbook using WorkbookFactory
        Workbook workbook = WorkbookFactory.create(file);
        Sheet sheet = workbook.getSheet("LoginData");

        // Read specific data (e.g., login credentials)
        String username = sheet.getRow(1).getCell(0).getStringCellValue();
        String password = sheet.getRow(1).getCell(1).getStringCellValue();

        // Close the workbook and file stream
        workbook.close();
        file.close();

        // Use the data for Selenium actions
        driver.get("https://the-internet.herokuapp.com/login");
        driver.findElement(By.id("username")).sendKeys(username);
        driver.findElement(By.id("password")).sendKeys(password);
        driver.findElement(By.cssSelector(".fa-sign-in")).click();

        // More selenium actions...

        driver.quit();
    }
}
```

Handling Excel Sheets using Apache POI in Selenium

Example 4: Handling Excel Data in a Data-Driven Framework

4. Handling Excel Data in a Data-Driven Framework

Steps:

1. Setup WebDriverManager:

2. Create a Utility Class to Read Excel Data:

3. Use the Utility Class in Test Scripts:

Utility Class:

```
import org.apache.poi.ss.usermodel.*;
import java.io.FileInputStream;
import java.io.IOException;

public class ExcelUtils {
    private Workbook workbook;

    public ExcelUtils(String excelPath) throws IOException {
        FileInputStream file = new FileInputStream(excelPath);
        this.workbook = WorkbookFactory.create(file);
    }

    public String getCellData(String sheetName, int rowNum, int colNum) {
```

```
        Sheet sheet = workbook.getSheet(sheetName);
        Row row = sheet.getRow(rowNum);
        Cell cell = row.getCell(colNum);
        return cell.getStringCellValue();
    }

    public void closeWorkbook() throws IOException {
        this.workbook.close();
    }
}
```

Test Script:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import io.github.bonigarcia.wdm.WebDriverManager;

import java.io.IOException;

public class ExcelReaderExample4 {
    public static void main(String[] args) throws IOException {
        // Setup WebDriverManager and initialize ChromeDriver
        WebDriverManager.chromedriver().setup();
        WebDriver driver = new ChromeDriver();
```

```
        // Path to the Excel file
        ExcelUtils excel = new ExcelUtils("path/to/excel/file.xlsx");

        // Read specific data (e.g., login credentials)
        String username = excel.getCellData("LoginData", 1, 0);
        String password = excel.getCellData("LoginData", 1, 1);

        // Use the data for Selenium actions
        driver.get("https://the-internet.herokuapp.com/login");
        driver.findElement(By.id("username")).sendKeys(username);
        driver.findElement(By.id("password")).sendKeys(password);
        driver.findElement(By.cssSelector(".fa-sign-in")).click();

        // More selenium actions...

        driver.quit();
        excel.closeWorkbook();
    }
}
```

Example 5: Reading Data from Excel and Writing Results

5. Reading Data from Excel and Writing Results

Handling Excel Sheets using Apache POI in Selenium

Steps:

1. Setup WebDriverManager:

2. Open Excel File using XSSFWorkbook:

3. Read Data and Write Results:

```
import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import io.github.bonigarcia.wdm.WebDriverManager;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class ExcelReaderExample5 {
    public static void main(String[] args) throws IOException {
        // Setup WebDriverManager and initialize ChromeDriver
        WebDriverManager.chromedriver().setup();
        WebDriver driver = new ChromeDriver();

        // Path to the Excel file
```

Handling Excel Sheets using Apache POI in Selenium

```
FileInputStream file = new FileInputStream("path/to/excel/file.xlsx");
XSSFWorkbook workbook = new XSSFWorkbook(file);
XSSFSheet sheet = workbook.getSheet("LoginData");

// Read specific data (e.g., login credentials)
String username = sheet.getRow(1).getCell(0).getStringCellValue();
String password = sheet.getRow(1).getCell(1).getStringCellValue();

// Use the data for Selenium actions
driver.get("https://the-internet.herokuapp.com/login");
driver.findElement(By.id("username")).sendKeys(username);
driver.findElement(By.id("password")).sendKeys(password);
driver.findElement(By.cssSelector(".fa-sign-in")).click();

// Verify login and write result back to Excel
boolean loginSuccessful = driver.findElement(By.cssSelector(".flash.success")).isDisplayed();
Row resultRow = sheet.getRow(1);
Cell resultCell = resultRow.createCell(2); // Assuming column 2 is for results
resultCell.setCellValue(loginSuccessful ? "Pass" : "Fail");

// Write the results back to the Excel file
FileOutputStream fileOut = new FileOutputStream("path/to/excel/file.xlsx");
workbook.write(fileOut);
fileOut.close();
```

```
        // Close the workbook and file stream
        workbook.close();
        file.close();


        driver.quit();
    }
}
```