60. Introduction to excelJS node module and setting up JS Project

how to read the Excel data and also how to write back into Excel data or modify the Excel data using JavaScript.

So once we know how to do it in the next phase, we will see how to integrate this into our existing JavaScript testing frameworks like Cypress and Playwright or WebDriver.

So for that there is one node module which you need to download which is Excel JS.

Exceljs use for Read, manipulate and write spreadsheet data and styles to XLSX and JSON.

Google->Exceljs->

Open the terminal -> create one new directory ( mkdir ExcelJSUtil ) -> So inside this directory I will initiate node project by simply giving ( npm init )

it simply create the hierarchy where you can enter all these dependencies.

This is a utility will walk you through creating this package dot Json file.

So we need this file because in this file itself we will give this dependency so that your project recognizes it.

So without package dot Json you cannot add external dependencies to your project.

So to this, file will be automatically created for you if you use npm init as a command.

we have to create this(package.json) inside this folder(ExcelJSUtil) So that way this would be our dedicated project name. I created this directory( mkdir ExcelJSUtil ) but I did not go inside this. So just let's go inside this ( cd ExcelJSUtil )

Once you go to that folder again run npm init.

now it will create that package dot Json inside this project.

So this project now we will import into our VS code where you write your JavaScript and you can work on it there.

In Vs code open and add this ExcelJSUtil file project.

Then run run this ( npm install exceljs )command in terminal. So this will help you to install this plugin into this project.

 if you also want entry of that ExcelJS into your package dot Json.

So instead of same command you can put ( npm install exceljs hyphen hyphen savedev )

if you write like this it will install Excel JS into your project. And also it will create one entry here in the package dot json. So that way going forward, anyone can import this project and then directly use it.

Now let me remove this node module again. It again installed right again node module is created. But you see that an entry is added into your package dot Json file.

So if you have entry like this you can zip this project and you can send it to anyone whenever they download, they will automatically have this dependency showing. So they can actually simply call npm install in their system. They don't need to give this Excel js.

So this is how you create a simple node JavaScript project with the dependencies now.

Create one new file (exceldemo.js)

61. Traversing rows and columns of excel worksheet with excelJS library

First we need to import that dependency into your file

Const Exceljs = require(‘exceljs’); So this will help us to import all those dependency file into your test.So make sure you import that into one variable Exceljs.

So now this holds actual piece of information with which you can read and write.

So usually we call Excel as a workbooks. So here we are more interested in workbooks here not Json or anything.

We created object for the class.

Using then for this code without using await

Based on the bewlow code this workbook will hold the collection of sheets

Worksheet will holds only one sheet. From that collection we are asking sheet1( sheet1 is the name)

Eachrow method – iterate over all rows that have vaues in a worksheet

const ExcelJs = require('exceljs');

const workbook = new ExcelJs.Workbook();   // created the object(Exceljs)

workbook.xlsx.readFile("C:/Users/Hari.Sankar/Downloads/exceldownloadTest.xlsx").then(function()

{

    const worksheet = workbook.getWorksheet('Sheet1');

    worksheet.eachRow((row,rowNumber)  =>

    {

        row.eachCell((cell,colNumber)  =>

        {

            console.log(cell.value);

        })

    })

})

Run : PS C:\Users\Hari.Sankar\ExcelJSUtil> node excelDemo.js

Now with await same code

const ExcelJs = require('exceljs');

async function excelTest()

{

const workbook = new ExcelJs.Workbook();   // created the object(Exceljs)

await workbook.xlsx.readFile("C:/Users/Hari.Sankar/Downloads/exceldownloadTest.xlsx")

    const worksheet = workbook.getWorksheet('Sheet1');

    worksheet.eachRow((row,rowNumber)  =>

    {

        row.eachCell((cell,colNumber)  =>

        {

            console.log(cell.value);

        })

    })

}

excelTest();

62. Build Util functions to read and update excel file strategically

Req :  verify if Apple is present in the Excel. So if it is present give me the coordinates.

So that means what is the row number and what is the column number

{

const workbook = new ExcelJs.Workbook();   // created the object(Exceljs)

await workbook.xlsx.readFile("C:/Users/Hari.Sankar/Downloads/exceldownloadTest.xlsx")

    const worksheet = workbook.getWorksheet('Sheet1');

    worksheet.eachRow((row,rowNumber)  =>

    {

        row.eachCell((cell,colNumber)  =>

        {

            if(cell.value == "Apple")

            {

            console.log(rowNumber);

            console.log(colNumber);

        }

        })

    })

} // output : 3,2

New req : Now my goal is replace Apple with iPhone ( hardcode the cell numbers)

const ExcelJs = require('exceljs');

async function excelTest()

{

const workbook = new ExcelJs.Workbook();   // created the object(Exceljs)

await workbook.xlsx.readFile("C:/Users/Hari.Sankar/Downloads/exceldownloadTest.xlsx")

    const worksheet = workbook.getWorksheet('Sheet1');

    worksheet.eachRow((row,rowNumber)  =>

    {

        row.eachCell((cell,colNumber)  =>

        {

            if(cell.value == "Apple")

            {

            console.log(rowNumber);

            console.log(colNumber);

        }

        })

    })

    const cell = worksheet.getCell(3,2);

    cell.value = "Iphone";

    await workbook.xlsx.writeFile("C:/Users/Hari.Sankar/Downloads/exceldownloadTest.xlsx")

}

excelTest();

another way - without HardCoding

const ExcelJs = require('exceljs');

async function excelTest()

{

let output = {row:-1,column:-1}; // created global object with two properties

const workbook = new ExcelJs.Workbook();   // created the object(Exceljs)

await workbook.xlsx.readFile("C:/Users/Hari.Sankar/Downloads/exceldownloadTest.xlsx")

    const worksheet = workbook.getWorksheet('Sheet1');

    worksheet.eachRow((row,rowNumber)  =>

    {

        row.eachCell((cell,colNumber)  =>

        {

            if(cell.value == "Republic")

            {

            output.row = rowNumber;

            output.column = colNumber; /republic rowno,colon sitting inside the output

        }

        })

    })

    const cell = worksheet.getCell(output.row,output.column);

    cell.value = "test";

    await workbook.xlsx.writeFile("C:/Users/Hari.Sankar/Downloads/exceldownloadTest.xlsx")

}

excelTest();

63. How to get and update the data from excel based on filter search criteria

 instead of writing the entire code in one function, I am, uh, segregating them into two different

pieces.

See, now we have built two reusable generic methods in JavaScript which is read Excel, Write Excel where you can call the Excel method with the parameters. And it will take care of doing all the operations without even touching the code. So all the desired text, replace text. Which file everything you can send as a parameters and code will take care.

const ExcelJs = require('exceljs');

async function writeExcelTest(searchText,replaceText,filePath)

{

const workbook = new ExcelJs.Workbook();   // created the object(Exceljs)

await workbook.xlsx.readFile(filePath);

    const worksheet = workbook.getWorksheet('Sheet1');

    const output = await readExcel(worksheet,searchText);

    const cell = worksheet.getCell(output.row,output.column);

    cell.value = replaceText;

    await workbook.xlsx.writeFile(filePath);

}

async function readExcel(worksheet,searchText)

{

    let output = {row:-1,column:-1};

    worksheet.eachRow((row,rowNumber)  =>

    {

        row.eachCell((cell,colNumber)  =>

        {

            if(cell.value == searchText)

            {

            output.row = rowNumber;

            output.column = colNumber;

        }

        })

    })

return output;

}

writeExcelTest("Mango","Rabbit","C:/Users/Hari.Sankar/Downloads/exceldata.xlsx");

change is an object which holds this information

const ExcelJs = require('exceljs');

async function writeExcelTest(searchText,replaceText,change,filePath)

{

const workbook = new ExcelJs.Workbook();   // created the object(Exceljs)

await workbook.xlsx.readFile(filePath);

    const worksheet = workbook.getWorksheet('Sheet1');

    const output = await readExcel(worksheet,searchText);

    const cell = worksheet.getCell(output.row,output.column+change.colChange);

    cell.value = replaceText;

    await workbook.xlsx.writeFile(filePath);

}

async function readExcel(worksheet,searchText)

{

    let output = {row:-1,column:-1};

    worksheet.eachRow((row,rowNumber)  =>

    {

        row.eachCell((cell,colNumber)  =>

        {

            if(cell.value == searchText)

            {

            output.row = rowNumber;

            output.column = colNumber;

        }

        })

    })

return output;

}

// update mango price to 350

writeExcelTest("Mango",350,{rowChange:0,colChange:2},"C:/Users/Hari.Sankar/Downloads/exceld.xlsx");

64. Strategy to handle download & uploading files using Playwright

65. End to end example for excel edits and upload with assertions using playwright

As of now whatever the pure node js code we implemented same code we created one new file(upload-download.spec.js) and paste the code here.

So when you click on download here, it downloads the Excel sheet.

So that Excel sheet will have the content exactly on how it is looking here.

And then when you modify that Excel and upload it again, then the modified data will also reflect in the table.

So now using playwright you need to validate that.

Add this in package dot json "dependencies": {

   "exceljs": "~4.3.0"

}

Add this const {test,expect} = require("@playwright/test");

const exceljs = require('exceljs');

const {test,expect} = require("@playwright/test");

async function writeExcelTest(searchText,replaceText,change,filePath)

{

const workbook = new exceljs.Workbook();   // created the object(Exceljs)

await workbook.xlsx.readFile(filePath);

    const worksheet = workbook.getWorksheet('Sheet1');

    const output = await readExcel(worksheet,searchText);

    const cell = worksheet.getCell(output.row,output.column+change.colChange);

    cell.value = replaceText;

    await workbook.xlsx.writeFile(filePath);

}

async function readExcel(worksheet,searchText)

{

    let output = {row:-1,column:-1};

    worksheet.eachRow((row,rowNumber)  =>

    {

        row.eachCell((cell,colNumber)  =>

        {

            if(cell.value == searchText)

            {

            output.row = rowNumber;

            output.column = colNumber;

        }

        })

    })

return output;

}

// update mango price to 350

//writeExcelTest("Mango",350,{rowChange:0,colChange:2},"C:/Users/Hari.Sankar/Downloads/exceld.xlsx");

test('Upload download excel validation',async ({page})=>

{

  const textSearch = 'Mango';

  const updateValue = '350';

  await page.goto("https://rahulshettyacademy.com/upload-download-test/index.html");

  const downloadPromise = page.waitForEvent('download');

  await page.getByRole('button',{name:'Download'}).click();

  await downloadPromise;

  writeExcelTest(textSearch,updateValue,{rowChange:0,colChange:2},"C:/Users/Hari.Sankar/Downloads/download.xlsx");

  await page.locator("#fileinput").click();

  await page.locator("#fileinput").setInputFiles("C:/Users/Hari.Sankar/Downloads/download.xlsx");

// for assertions

  const textlocator = page.getByText(textSearch);

  const desiredRow = await page.getByRole('row').filter({ has: textlocator });

  await expect(desiredRow.locator("#cell-4-undefined")).toContainText(updateValue);

})