42. Working with Excel File

If we have to execute the same test case with multiple sets of data multiple times called **Data Driven Testing.** Test data will be in Excel files, Database.

Usually, we use Excel files to create test data, and use the same data in our test cases.

openpyxl and **pandas** Libraries are used for interacting with Excel files in Python.

Structure of Excel file - File → Workbook → Sheets → Rows → Cells

Operations on Excel File

- Count number of rows and columns
- Read Data from all rows and columns and specific column data
- writing same data into multiple cells
- writing multiple data into multiple cells

Read Data from Cell	Write Data into Cell
openpyxl	**openpyxl**
data=sheet.cell(r,c).value	sheet.cell(r,c).value = data
pandas	**pandas**
data = df.iloc[row_index, column_index] → **indices**	df.to_excel(file, index=False) → **All data**
data = df.loc[row_label, column_label] → **names**	df.at[row_index, column_label] = data → single cell

Reading Excel.py

openpyxl

```
import openpyxl
file = r"C:\Selenium with Python\data.xlsx"
workbook=openpyxl.load_workbook(file)
sheet=workbook["Sheet1"]
```

rows and columns

```
print("Number of Rows: ",sheet.max_row)
print("Number of Columns: ",sheet.max_column)
print()
```

Read all rows and columns data

```
for r in range(1,rows+1):
  for c in range(1,cols+1):
     print(sheet.cell(r,c).value,end='      ')
  print()
```

pandas

```
import pandas as pd
file = r"C:\Selenium with Python\data.xlsx"
df = pd.read_excel(file, sheet_name="Sheet1",
header=None)
```

rows and columns

```
print("Number of Rows: “,{len(df)})
print("Number of Columns: “,{len(df.columns)})
print()
```

Read all rows and columns data

```
for index, row in df.iterrows():
  for value in row:
     print(value, end='      ')
  print()
```

Writing Data Into Excel.py

writing same data into multiple cells

openpyxl

without column names

```
import openpyxl
file=r"C:\\Selenium with Python\\testdata.xlsx"
```

pandas

without column names

```
import pandas as pd
file = r"C:\\Selenium with Python\\testdata.xlsx"
```

```
workbook=openpyxl.load_workbook(file)
  sheet=workbook.active  #get only active sheet
                or
          sheet=workbook["Data"]
for r in range(1,6):
   for c in range(1,4):
      sheet.cell(r,c).value="welcome"
workbook.save(file)
```

with column names

```
import openpyxl
file = r"C:\\Selenium with Python\\testdata.xlsx"
workbook = openpyxl.load_workbook(file)
  sheet=workbook.active  #get only active sheet
                or
          sheet=workbook["Data"]
columns = ["Column1", "Column2", "Column3"]
for c, columns in enumerate(columns, start=1):
   sheet.cell(1, c).value = columns
for r in range(2, 7):  # Rows 2 to 6
   for c in range(1, 4):  # Columns 1 to 3
      sheet.cell(r, c).value = "welcome"
workbook.save(file)
```

```
df = pd.DataFrame(index=range(1, 6),
columns=range(1, 4))
df[:] = "welcome"
df.to_excel(file, index=False, header=False)
```

with column names

```
import pandas as pd
file = "C:\\Selenium with Python\\testdata.xlsx"
columns = ["Column1", "Column2", "Column3"]
df = pd.DataFrame(index=range(1, 6),
columns=columns)
df[:] = "welcome"
df.to_excel(file, index=False)
```

writing multiple data into multiple cells

openpyxl

without column names

```
import openpyxl
file=r"C:\\Selenium with Python\\testdata2.xlsx"
workbook=openpyxl.load_workbook(file)
  sheet=workbook.active  #get only active sheet
                or
          sheet=workbook["Data"]
sheet.cell(1,1).value=123
sheet.cell(1,2).value="smith"
sheet.cell(1,3).value="engineer"
sheet.cell(2,1).value=567
sheet.cell(2,2).value="john"
sheet.cell(2,3).value="manager"
sheet.cell(3,1).value=567
sheet.cell(3,2).value="david"
```

pandas

without column names

```
import pandas as pd
data = [
   [123, 'smith', 'engineer'],
   [567, 'john', 'manager'],
   [567, 'david', 'developer']
]
df = pd.DataFrame(data)
file = r"C:\\Selenium with Python\\testdata2.xlsx"
df.to_excel(file, index=False, header=False,
sheet_name='Data')
```

with column names

```
import pandas as pd
data = {
   'Column1': [123, 567, 567],
```

```
sheet.cell(3,3).value="developer"
workbook.save(file)
```

with column names

```
import openpyxl
file = r"C:\\Selenium with Python\\testdata2.xlsx"
workbook = openpyxl.load_workbook(file)
 sheet=workbook.active  #get only active sheet
```

or

```
sheet=workbook["Data"]
columns = ["Column1", "Column2", "Column3"]
# Write column names to the first row
for col_num, column_name in enumerate(columns, start=1):
  sheet.cell(1, col_num).value = column_name
data = [
  [123, "smith", "engineer"],
  [567, "john", "manager"],
  [567, "david", "developer"]
]
for row_num, row_data in enumerate(data, start=2):
  for col_num, value in enumerate(row_data, start=1):
      sheet.cell(row_num, col_num).value = value
workbook.save(file)
```

```
  'Column2': ['smith', 'john', 'david'],
  'Column3': ['engineer', 'manager', 'developer']
}
df = pd.DataFrame(data)
file = r"C:\\Selenium with Python\\testdata2.xlsx"
df.to_excel(file, index=False, sheet_name='Data')
```

Read specific column data using Pandas

```
import pandas as pd
file_path = r"C:\\Selenium with Python\\data.xlsx"
sheet_name = 'Sheet1'
column_name = 'BookName'
df = pd.read_excel(file_path, sheet_name=sheet_name)
column_data = df[column_name]
for value in column_data.values:
  print(value)
```

Read specific column data using Openpyxl other than Date

```
import openpyxl
file_path = r"C:\Users\DELL\PycharmProjects\Selenium with Python B2\Day28\data.xlsx"
workbook = openpyxl.load_workbook(file_path)
sheet = workbook['Sheet1']
header_row_index = 1
column_name = 'BookName'
column_index = None
for cell in sheet[header_row_index]:
  if cell.value == column_name:
    column_index = cell.column
    break
if column_index is None:
  raise ValueError(f"Column '{column_name}' not found in the header row.")
```

```
for row in sheet.iter_rows(min_row=header_row_index + 1, min_col=column_index,
max_col=column_index, values_only=True):
  print(row[0])
```

Read specific column data using Openpyxl with Date

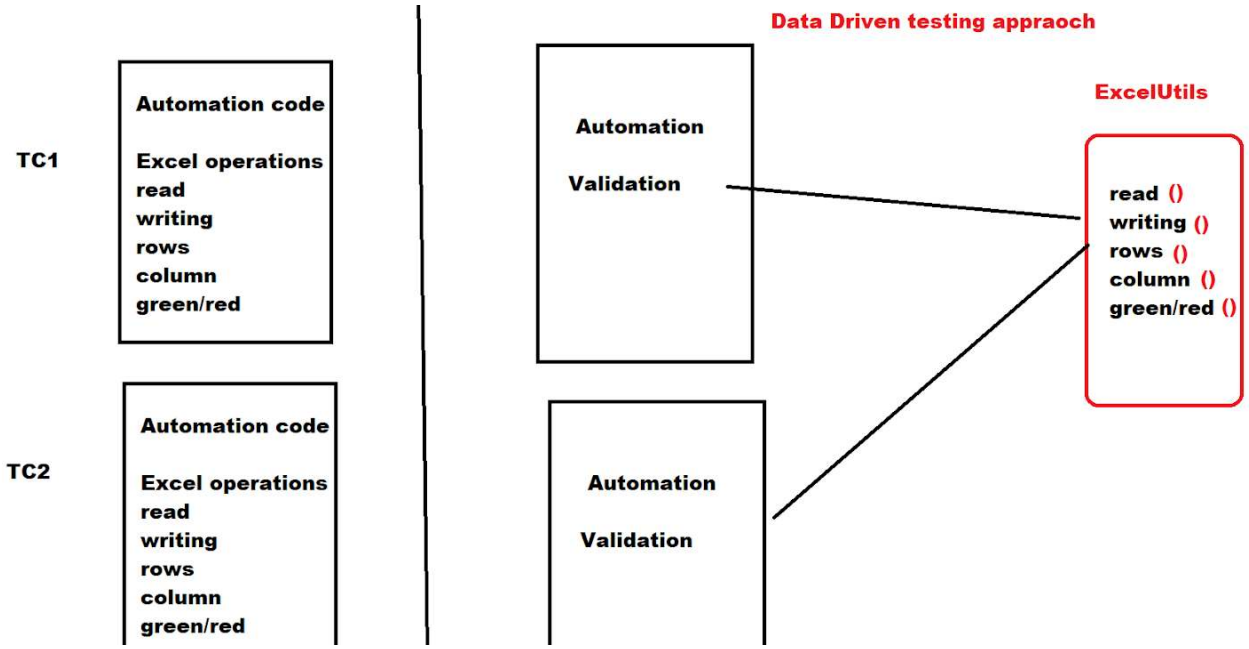
```
import openpyxl
from datetime import datetime
file_path = r"C:\Users\DELL\PycharmProjects\Selenium with Python B2\Day28\data.xlsx"
workbook = openpyxl.load_workbook(file_path)
sheet = workbook['Sheet1']
header_row_index = 1
column_name = 'PurchasedDate'
column_index = None
for cell in sheet[header_row_index]:
  if cell.value == column_name:
    column_index = cell.column
    break
if column_index is None:
  raise ValueError(f"Column '{column_name}' not found in the header row.")
column_data = []
for row in sheet.iter_rows(min_row=header_row_index + 1, min_col=column_index,
max_col=column_index, values_only=True):
  cell_value = row[0]
  if isinstance(cell_value, datetime):
    formatted_date = cell_value.strftime('%d-%b-%Y')
    column_data.append(formatted_date)
  else:
    column_data.append(cell_value)
print(column_data)
```

Another type of date formats to be specified for strftime()

- Day-Month-Year (e.g., 29-Jul-2019): **%d-%b-%Y**
- Month/Day/Year (e.g., 07/29/2019): **%m/%d/%Y**
- Year-Month-Day (e.g., 2019-07-29): **%Y-%m-%d**
- Full Month Name Day, Year (e.g., July 29, 2019): **%B %d, %Y**
- Day Full Month Name Year (e.g., 29 July 2019): **%d %B %Y**

43. Data Driven Testing using Excel File

If there are 10 TC, apart from **Automation code(complex)** if we again include Excel operations, **Complexity** of the test case will be increased.Also **Duplication** because whatever the operations we do in one test case we do the same in another test case.To avoid this we use a **utility file** which contains all the reusable functions(no of rows, no of columns, read the data, write the data, apply colors, etc) and these functions are invoked in every automation test.Data Driven Test contains Web Element Actions and Validation Points.



XLUtils.py

```
import openpyxl
from openpyxl.styles import PatternFill
def getRowCount(file, sheetName):
  workbook = openpyxl.load_workbook(file)
  sheet = workbook[sheetName]
  return (sheet.max_row)
def getColumnCount(file, sheetName):
  workbook = openpyxl.load_workbook(file)
  sheet = workbook[sheetName]
  return (sheet.max_column)
def readData(file, sheetName, rownum, columnno):
  workbook = openpyxl.load_workbook(file)
  sheet = workbook[sheetName]
  return sheet.cell(rownum, columnno).value
def writeData(file, sheetName, rownum, columnno, data):
  workbook = openpyxl.load_workbook(file)
  sheet = workbook[sheetName]
  sheet.cell(rownum, columnno).value = data
  workbook.save(file)
def fillGreenColor(file, sheetName, rownum, columnno):
  workbook = openpyxl.load_workbook(file)
```

```
  sheet = workbook[sheetName]
  greenFill = PatternFill(start_color='60b212',end_color='60b212', fill_type='solid')
  sheet.cell(rownum, columnno).fill = greenFill
  workbook.save(file)
def fillRedColor(file, sheetName, rownum, columnno):
  workbook = openpyxl.load_workbook(file)
  sheet = workbook[sheetName]
  redFill = PatternFill(start_color='ff0000', end_color='ff0000', fill_type='solid')
  sheet.cell(rownum, columnno).fill = redFill
  workbook.save(file)
```

Data Driven Test Cases using utility file

CertificatesofDepositsCaluclator.py

```
import time
import XLUtils
from selenium import webdriver
from selenium.webdriver.common.by import By
options = webdriver.ChromeOptions()
options.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=options)
driver.get("https://www.cit.com/cit-bank/resources/calculators/certificate-of-deposit-calculator/")
driver.maximize_window()
driver.implicitly_wait(10)
```

find_element

```
inideposit = driver.find_element(By.XPATH,"//input[@id='mat-input-0']")
length = driver.find_element(By.XPATH,"//input[@id='mat-input-1']")
apr = driver.find_element(By.XPATH,"//input[@id='mat-input-2']")
calbutton = driver.find_element(By.XPATH,"//button[@id='CIT-chart-submit']")
```

file_path

```
path = r"C:\Selenium with Python\caldata2.xlsx"
```

count_rows

```
rows = XLUtils.getRowCount(path, "Sheet1")
print("row count is : " , rows)
```

read_data

```
for r in range(2,rows+1):
        inidepo = XLUtils.readData(path, "Sheet1", r, 1)
        interestrate = XLUtils.readData(path, "Sheet1", r, 2)
        monthlength = XLUtils.readData(path, "Sheet1", r, 3)
        compoundingmonths = XLUtils.readData(path, "Sheet1", r, 4)
        exptotal = XLUtils.readData(path, "Sheet1", r, 5)
```

clear data in the application

```
        inideposit.clear()
```

```
        length.clear()
        apr.clear()
        time.sleep(3)
                                        pass data to the application
        inideposit.send_keys(inidepo)
        length.send_keys(monthlength)
        apr.send_keys(interestrate)
                                        BootStrap Dropdown
        compoundrp = driver.find_element(By.XPATH,"//mat-select[@id='mat-select-0']")
        compoundrp.click()
        options = driver.find_elements(By.XPATH,"//div[@id='mat-select-0-panel']//mat-option")
        for option in options:
            if(option.text==compoundingmonths):
                option.click()
        calbutton.click()
        acttotal = driver.find_element(By.XPATH,"//span[@id='displayTotalValue']").text
        print("exp total is from excel: ", exptotal)
        print("act total is from app: ", acttotal)
                                        validation
        if(exptotal==acttotal):
            print("test passed")
            XLUtils.writeData(path, "Sheet1" ,r, 7 ,"Passed")
            XLUtils.fillGreenColor(path, "Sheet1" ,r, 7)
        else:
            print("test failed")
            XLUtils.writeData(path, "Sheet1" ,r, 7 ,"Failed")
            XLUtils.fillRedColor(path, "Sheet1" ,r, 7)
print("calculation has been completed")
driver.close()
```

FixedDepositCalculator.py

```
import time
import XLUtils
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select
options = webdriver.ChromeOptions()
options.add_argument("--disable-notifications")
options.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=options)
driver.get("https://www.moneycontrol.com/fixed-income/calculator/state-bank-of-india-sbi/fixed-depo
sit-calculator-SBI-BSB001.html")
```

```
driver.maximize_window()
driver.implicitly_wait(10)
                                                    file_path
file = "C:\Selenium with Python\caldata.xlsx"
                                                    count_rows
rows = XLUtils.getRowCount(file, "Sheet1")
print("row count is : " , rows)
                                                    read_data
for r in range(2,rows+1):
        pric = XLUtils.readData(file,"Sheet1",r,1)
        rateofinterest = XLUtils.readData(file,"Sheet1",r,2)
        per1 = XLUtils.readData(file, "Sheet1", r, 3)
        per2 = XLUtils.readData(file, "Sheet1", r, 4)
        fre = XLUtils.readData(file, "Sheet1", r, 5)
        exp_mvalue = XLUtils.readData(file, "Sheet1", r, 6)
                                        find element and pass data to element
        driver.find_element("xpath","//input[@id='principal']").send_keys(pric)
        driver.find_element("xpath","//input[@id='interest']").send_keys(rateofinterest)
        driver.find_element("xpath","//input[@id='tenure']").send_keys(per1)
                                                    Dropdown
        perioddrp = Select(driver.find_element("xpath","//select[@id='tenurePeriod']"))
        perioddrp.select_by_visible_text(per2)
        frequencydrp = Select(driver.find_element("xpath","//select[@id='frequency']"))
        frequencydrp.select_by_visible_text(fre)
        driver.find_element(By.XPATH,"//*[@id='fdMatVal']/div[2]/a[1]/img").click()  # calculate button
        act_mvalue = driver.find_element("xpath","//span[@id='resp_matval']/strong").text
        print("exp total is from excel: ", exp_mvalue)
        print("act total is from app: ", act_mvalue)
                                                    Validation
        if float(exp_mvalue)==float(act_mvalue):
           print("test passed")
           XLUtils.writeData(file,"Sheet1",r,8,"Passed")
           XLUtils.fillGreenColor(file,"Sheet1",r,8)
        else:
           print("test failed")
           XLUtils.writeData(file,"Sheet1",r,8,"Failed")
           XLUtils.fillRedColor(file,"Sheet1",r,8)
        driver.find_element(By.XPATH,"//*[@id='fdMatVal']/div[2]/a[2]/img").click()
        time.sleep(2)
print("calculation has been completed")
driver.close()
```