Outlook

Fwd: Software Testing

From 727823TUIT204 SANJAI C R V <727823tuit204@skct.edu.in>
Date Mon 8/25/2025 11:01 AM
To 727823tuit217@skct.edu.in <727823tuit217@skct.edu.in>

---------- Forwarded message ---------
From: **727823TUIT204 SANJAI C R V** <727823tuit204@skct.edu.in>
Date: Sun, Aug 24, 2025 at 6:29 PM
Subject: Software Testing
To: Sanjai CRV <sanjaicrv05@gmail.com>

Excel read and write

```
package utils;

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

import org.apache.poi.ss.usermodel.DataFormatter;
import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ExcelReader {

        public static FileInputStream file;
        public static XSSFWorkbook workbook;
        public static XSSFSheet sheet;
        public static FileOutputStream fileout;
        public static XSSFRow row;
        public static XSSFCell col;
        public static int rowvalue;
        public static int colvalue;

        public static String readdata(String filepath, String sheetname, int rownumber, int
colnumber) throws IOException

        {

                try {
                        file = new FileInputStream(filepath);
                        workbook = new XSSFWorkbook(file);
                } catch (FileNotFoundException e) {
```

```
                        e.printStackTrace();
                } catch (IOException e) {

                        e.printStackTrace();
                }
                sheet = workbook.getSheet(sheetname);
                row = sheet.getRow(rownumber);
                col = row.getCell(colnumber);

                // String value = col.toString();
                String value;

                try {
                        DataFormatter d = new DataFormatter();
                        value = d.formatCellValue(col);
                        return value;

                } catch (Exception e) {
                        value = "";
                }
                workbook.close();
                file.close();
                return value;

        }

        public static void writedata(String filepath, String sheetname, int rownumber, int
colnumber, String value)
                        throws IOException {
                try {
                        file = new FileInputStream(filepath);
                        workbook = new XSSFWorkbook(file);
                } catch (FileNotFoundException e) {

                        e.printStackTrace();
                } catch (IOException e) {

                        e.printStackTrace();
                }

                sheet = workbook.getSheet(sheetname);

                row = sheet.getRow(rownumber);

                col = row.getCell(colnumber);

                col.setCellValue(value);

                try {
                        fileout = new FileOutputStream(filepath);
                        workbook.write(fileout);
                } catch (FileNotFoundException e) {

                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }

                workbook.close();

                file.close();

                fileout.close();

        }
```

```
}
```

Logger Handler

```
package utils;

import java.text.SimpleDateFormat;
import java.util.Date;

import org.apache.log4j.FileAppender;
import org.apache.log4j.Logger;
import org.apache.log4j.PatternLayout;

public class LoggerHandler {

    private static final Logger logger = Logger.getLogger(LoggerHandler.class);



    static {
        try {
            // Set up a file appender with a timestamp in the filename
            String timestamp = new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss").format(new Date());
            String logFileName = "logs/logfile_" + timestamp + ".log";

            FileAppender fileAppender = new FileAppender(new PatternLayout("%d{ISO8601} %-5p %c -
%m%n"), logFileName, true);
            logger.addAppender(fileAppender);
        } catch (Exception e) {
            logger.error("Failed to initialize logger file appender", e);
        }
    }
    public static void trace(String message) {
        logger.trace(message);
    }

    public static void debug(String message) {
        logger.debug(message);
    }

    public static void info(String message) {
        logger.info(message);
    }

    public static void warn(String message) {
        logger.warn(message);
    }

    public static void error(String message) {
        logger.error(message);
```

```
    }

    public static void fatal(String message) {
        logger.fatal(message);
    }

}
```

Screenshot

```
package utils;

import java.io.File;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;

import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;

import com.google.common.io.Files;

public class Screenshot extends Base {

    public static TakesScreenshot ts;

    public static void captureScreenShot(String filename) {
        String timestamp = new SimpleDateFormat("yyyy.MM.dd.HH.mm.ss").format(new Date());
        String name = filename + timestamp + ".png";
        ts = (TakesScreenshot) driver;
        File file = ts.getScreenshotAs(OutputType.FILE);

        // Create the screenshots directory if it doesn't exist
        File screenshotsDir = new File(System.getProperty("user.dir") + "/screenshots");
        if (!screenshotsDir.exists()) {
            screenshotsDir.mkdirs();
        }

        File target = new File(screenshotsDir, name);
        try {
            Files.copy(file, target);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

driver helper

```java
package utils;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;
import java.util.Set;

public class WebDriverHelper {
    private WebDriver driver;

    public WebDriverHelper(WebDriver driver) {
        this.driver = driver;
    }

    public void waitForElementToBeVisible(By locator, int timeoutInSeconds) {
        try {
            new WebDriverWait(driver, Duration.ofSeconds(timeoutInSeconds))
                .until(ExpectedConditions.visibilityOfElementLocated(locator));
        } catch (Exception e) {
            // Handle or rethrow the exception here
            e.printStackTrace();
        }
    }

    public void clickOnElement(By locator) {
        try {
            WebElement webElement = driver.findElement(locator);
            webElement.click();
        } catch (Exception e) {
            // Handle or rethrow the exception here
            e.printStackTrace();
        }
    }

    public void sendKeys(By locator, String data) {
        try {
            WebElement webElement = driver.findElement(locator);
            webElement.sendKeys(data);
        } catch (Exception e) {
```

```
            // Handle or rethrow the exception here
            e.printStackTrace();
        }
    }

    public String getText(By locator) {
        try {
            WebElement webElement = driver.findElement(locator);
            return webElement.getText();
        } catch (Exception e) {
            // Handle or rethrow the exception here
            e.printStackTrace();
            return " ";
        }
    }

    public void jsClick(By locator) {
        try {
            WebElement element = driver.findElement(locator);
            JavascriptExecutor js = (JavascriptExecutor) driver;
            js.executeScript("arguments[0].click();", element);
        } catch (Exception e) {
            // Handle or rethrow the exception here
            e.printStackTrace();
        }
    }

    public void javascriptScroll(By locator) {
        try {
            WebElement element = driver.findElement(locator);
            JavascriptExecutor js = (JavascriptExecutor) driver;
            js.executeScript("arguments[0].scrollIntoView();", element);
        } catch (Exception e) {
            // Handle or rethrow the exception here
            e.printStackTrace();
        }
    }

    public void switchToNewWindow() {
        try {
            Set<String> windowHandles = driver.getWindowHandles();
            for (String windowHandle : windowHandles) {
                if (!windowHandle.isEmpty()) {
                    driver.switchTo().window(windowHandle);
                } else {
                    throw new Exception("New window could not be retrieved");
                }
            }
        } catch (Exception e) {
            // Handle or rethrow the exception here
            e.printStackTrace();
        }
    }

    public void enterAction(By locator) {
        try {
            WebElement webElement = driver.findElement(locator);
```

```
            webElement.sendKeys(Keys.ENTER);
        } catch (Exception e) {
            // Handle or rethrow the exception here
            e.printStackTrace();
        }
    }

    public void hoverOverElement(By locator) {
        try {
            WebElement webElement = driver.findElement(locator);
            Actions actions = new Actions(driver);
            actions.moveToElement(webElement).perform();
        } catch (Exception e) {
            // Handle or rethrow the exception here
            e.printStackTrace();
        }
    }
}
```

Reporter

```
package utils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Base64;
import java.util.Date;
import java.util.Properties;
import java.util.TimeZone;

import com.google.common.io.Files;


import com.aventstack.extentreports.ExtentReports;
import com.aventstack.extentreports.ExtentTest;
import com.aventstack.extentreports.MediaEntityBuilder;
import com.aventstack.extentreports.Status;
import com.aventstack.extentreports.reporter.ExtentSparkReporter;
import com.aventstack.extentreports.reporter.configuration.Theme;

import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
```

```
public class Reporter extends Base {
    public static TakesScreenshot ts;

    private static Properties prop;
    private static ExtentReports extentReport;
    private static ExtentTest test;


    public static ExtentReports generateExtentReport() {
        return generateExtentReport(null);
    }

    public static ExtentReports generateExtentReport(String reportName) {
        if (extentReport == null) {
            extentReport = createExtentReport(reportName);
        }
        return extentReport;
    }

    private static ExtentReports createExtentReport(String reportName) {
        ExtentReports extentReport = new ExtentReports();

        // Load properties from browser.properties file
        String filepath = System.getProperty("user.dir") + "/config/browser.properties";
        try {
            FileInputStream file = new FileInputStream(filepath);
            prop = new Properties();
            prop.load(file);
        } catch (IOException e) {
            System.out.println(e.getLocalizedMessage());
        }

        // Get the current timestamp for the report name
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss");
        TimeZone istTimeZone = TimeZone.getTimeZone("Asia/Kolkata"); // IST timezone
        dateFormat.setTimeZone(istTimeZone);
        String timestamp = dateFormat.format(new Date());

        // Define the report file path with the timestamp and provided report name
        String reportFilePath = System.getProperty("user.dir") + "/reports/";
        if (reportName == null || reportName.isEmpty()) {
            reportName = "Test Report";
        }
        reportFilePath += reportName + "_" + timestamp + ".html";

        File extentReportFile = new File(reportFilePath);

        ExtentSparkReporter sparkReporter = new ExtentSparkReporter(extentReportFile);

        sparkReporter.config().setTheme(Theme.DARK);
        sparkReporter.config().setReportName("test Report");
        sparkReporter.config().setDocumentTitle("test Automation Report");
        sparkReporter.config().setTimeStampFormat("dd/MM/yyyy hh:mm:ss");

        extentReport.attachReporter(sparkReporter);

        extentReport.setSystemInfo("Application URL", prop.getProperty("url"));
```

```
        extentReport.setSystemInfo("Browser Name", prop.getProperty("browserName"));
        extentReport.setSystemInfo("Email", prop.getProperty("validEmail"));
        extentReport.setSystemInfo("Password", prop.getProperty("validPassword"));
        extentReport.setSystemInfo("Operating System", System.getProperty("os.name"));
        extentReport.setSystemInfo("Username", System.getProperty("user.name"));
        extentReport.setSystemInfo("Java Version", System.getProperty("java.version"));

        return extentReport;
    }

    public static String captureScreenshotAsBase64(WebDriver driver, String screenshotName) {
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss");
        TimeZone istTimeZone = TimeZone.getTimeZone("Asia/Kolkata"); // IST timezone
        dateFormat.setTimeZone(istTimeZone);
        String timestamp = dateFormat.format(new Date());

        TakesScreenshot screenshotDriver = (TakesScreenshot) driver;
        byte[] screenshotBytes = screenshotDriver.getScreenshotAs(OutputType.BYTES);

        String base64Screenshot = "";
        try {
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            baos.write(screenshotBytes);
            base64Screenshot = Base64.getEncoder().encodeToString(baos.toByteArray());

            // Save the screenshot to a file for reference
            saveScreenshotToFile(screenshotBytes, screenshotName + "_" + timestamp + ".png");
        } catch (IOException e) {
            e.printStackTrace();
        }

        return base64Screenshot;
    }

    private static String saveScreenshotToFile(byte[] screenshotBytes, String fileName) {
        String screenshotsDirPath = System.getProperty("user.dir") +
"/reports/errorScreenshots/";

        try {
            File screenshotsDir = new File(screenshotsDirPath);
            if (!screenshotsDir.exists())
             {
                screenshotsDir.mkdirs();
            }

            String destinationScreenshotPath = screenshotsDirPath + fileName;
            FileOutputStream outputStream = new FileOutputStream(destinationScreenshotPath);
            outputStream.write(screenshotBytes);
            outputStream.close();
        }

        catch (IOException e) {
            e.printStackTrace();
        }
        String destinationScreenshotPath = screenshotsDirPath + fileName;

        return destinationScreenshotPath;
```

```
    }


    public static String captureScreenShot(String filename) {
        String timestamp = new SimpleDateFormat("yyyy.MM.dd.HH.mm.ss").format(new Date());
        String name = filename + timestamp + ".png";

        String destPath =  "./"+name;

        ts = (TakesScreenshot) driver;
        File file = ts.getScreenshotAs(OutputType.FILE);

        // Create the screenshots directory if it doesn't exist
        File screenshotsDir = new File(System.getProperty("user.dir") + "/reports");

        if (!screenshotsDir.exists()) {
            screenshotsDir.mkdirs();
        }

        File target = new File(screenshotsDir, name);
        try {
            Files.copy(file, target);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return destPath;
    }

    public static void attachScreenshotToReport(String filename, ExtentTest test, String
description) {
        try {
            test.log(Status.INFO, description,
MediaEntityBuilder.createScreenCaptureFromPath(captureScreenShot(filename)).build());
        } catch (Exception e) {
            e.printStackTrace();
        }


    }

}
```

event handler

```
package utils;

import org.openqa.selenium.support.events.WebDriverListener;
```

```
public class EventHandler implements WebDriverListener {

}
```

Base.java

```
package utils;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.time.Duration;
import java.util.Properties;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.support.events.EventFiringDecorator;
import org.openqa.selenium.support.events.WebDriverListener;

public class Base {

    public static WebDriver driver;
    public static FileInputStream file;
    public static Properties prop;

    public void loadProperties() throws IOException {
        String propertiesPath = System.getProperty("user.dir") + "/config/browser.properties";
        try {
            file = new FileInputStream(propertiesPath);
            prop = new Properties();
            prop.load(file);

        } catch (FileNotFoundException e) {
            e.printStackTrace();

        }
    }

    public void openBrowser() {

        try {
```

```
            loadProperties();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        String executionType = prop.getProperty("executiontype");
        String browserName = prop.getProperty("browser");

        if ("remote".equalsIgnoreCase(executionType)) {
            URL gridUrl;
            try {
                gridUrl = new URL(prop.getProperty("gridurl"));
                driver = new RemoteWebDriver(gridUrl, new ChromeOptions());
            } catch (MalformedURLException e) {

                e.printStackTrace();
            }

        } else if ("local".equalsIgnoreCase(executionType)) {
            switch (browserName.toLowerCase()) {
                case "chrome":
                    driver = new ChromeDriver();
                    break;

                case "edge":
                    driver = new EdgeDriver();
                    break;

                case "firefox":
                    driver = new FirefoxDriver();
                    break;

                default:
                    System.err.println("Unsupported browser: " + browserName);
                    break;
            }
        } else {
            System.err.println("Invalid execution type: " + executionType);
        }

        if (driver != null)

        {
            driver.manage().window().maximize();
            driver.get(prop.getProperty("url"));
            driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(8));
            driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(8));

        }
        // Dont remove the listener Object

        WebDriverListener listener = new EventHandler();
        driver = new EventFiringDecorator<>(listener).decorate(driver);

    }

}
```