**Set up TestNG Framework\_Mukesh**

1. Maven – build tool

2. TestNG – unit testing framework

3. Jenkins – CI tool

4. GIT – local repository

5. GITHUB – web repository

6. Page Object Model (POM) – design pattern

7. Reading External File

8. Selenium

9. Report – Extent report for information and testNG Reporter for tests…

Step 1 : create a Maven Project.

Step 2 : create packages ( right click on src/test/java – new – package - name like com.something.pages, com.something.testcases, com.something.utility, config\_package.

Step 3 : create few folders. ( right click on the project – new – folder – name folder )

reports, screenshots, xmlFiles, test\_data (XL file), config\_data folders etc.

: **create Excel file under folder**- C: > folder > right click on empty space > new > Microsoft Excel.

Step 4 : add all dependency to POM. ( Selenium, testNG, WebDriverManager, Extent report, Apache POI+Apache ooxml etc and maven surfire pluggin) go to maven dependency then type selenium, testNG, extent report,. then press Ctrl+S it will make the POM ready.

[www.crm.com](http://www.crm.com) – [jawadkarim121@yahoo.com](mailto:jawadkarim121@yahoo.com)

password – Poland2021

Step 5 : start design Page object design pattern.

o. create a class for every single web Page in page package.

i. in Utility package create a class Browser factory.

ii. create test cases into com.testcases package. Then extends to Base class.

iii. create a base\_class in page\_package and add @BeforeClass & @AfterClass annoted methods. Then test classes extends base\_class.

iv. convert the main test case into xml file, then put into xmlFiles folder.

v. create a file config.properties into config\_data folder. ( right click on the config folder – new – file – name config.properties. put the key value pairs into the file.. like Browsers = chrome,

link = [www.abcd.com](http://www.abcd.com) etc.

vi. into test\_data folder create an Excel sheet for data. ( right click on folder > properties > click location button > click test\_data folder > right click on emty space > new > michrosoft XL worksheet > do name > enter > click to open > store data.

vii. create a class ExcelDataProvider into utility package. Create a constructor & write the Excel sheet path.

viii. create a class TakesScreenShot into utility package.

ix. create a class Sending Email into utility package.

x. create a class Read\_object into config package. Create a constructor & write the configure file path for reading object repository.

xi. create a custom library class in page\_package...

Write methods in that class…

**package com.frame.utility**;

**public class ConfigDataProvider** {

**public** Properties pro;

**public ConfigDataProvider()** {

**try** {

File file = **new** File("./config/config.proerties");

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

pro = **new** Properties();

pro.load(fis);

} **catch** (Exception e) {

System.***out***.println("file exception : "+ e.getMessage());

}

}

**public String getDataFromConfig**(String keyToSearch) {

**return** pro.getProperty(keyToSearch);

}

**public String getBrowser**() {

**return** pro.getProperty("Browser");

}

**public String getLinkUrl**() {

**return** pro.getProperty("PageUrl");

}

}

package com.frame.utility; **//dependency apache poi 3.17 version**

public class XLdataProvider {

XSSFWorkbook wb;

public XLdataProvider() {

File file = **new** File("C:\\Users\\Elias Karim\\Documents\\FreeCRM.xlsx");

//Ctrl+Shift+right click on file + copy as path

**try** {

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

wb = **new** XSSFWorkbook(fis);

}

**catch** (Exception e) {

System.***out***.println("unable to read xl file" + "exception : "+ e.getMessage());

}

}

**public String getStringData**(String sheetName, **int** row, **int** col) {

**return** wb.getSheet(sheetName).getRow(row).getCell(col).getStringCellValue();

}

**public String getStringData**(**int** sheetNum, **int** row, **int** col) {

**return** wb.getSheetAt(sheetNum).getRow(row).getCell(col).getStringCellValue();

}

**public double getNumericData**(**int** sheetNum, **int** row, **int** col) {

**return** wb.getSheetAt(sheetNum).getRow(row).getCell(col).getNumericCellValue();

}

**public double getNumericData**(**String** sheetName, **int** row, **int** col) {

**return** wb.getSheetAt(sheetName).getRow(row).getCell(col).getNumericCellValue();

}

}

--------------

**package** utility;

**public** **class** ReadExcel {

**public** **void** xlData() **throws** Exception {

File file = **new** File("filePath");

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

XSSFWorkbook wb = **new** XSSFWorkbook(fis);

XSSFSheet s1 = wb.getSheetAt(0);

**int** rowCount = s1.getLastRowNum();

**int** cellCount = s1.getRow(0).getLastCellNum();

**for**(**int** i=0; i<=rowCount; i++) {

**for**(**int** j=0; j<=cellCount; j++) {

String data1[i][j] = s1.getRow(i).getCell(j).getStringCellValue();

//**int** data2 = (**int**) s1.getRow(i).getCell(j).getNumericCellValue();

System.***out***.println("data : "+data1);

}

}

}

}

**package com.frame.utility**;

**public** **class** BrowserFactory {

**public** **static** WebDriver startBrowser(WebDriver driver, String browser, String url) {

**if**(browser.equalsIgnoreCase("firefox")) {

WebDriverManager.*firefoxdriver*().setup();

driver = **new** ChromeDriver();

}

**else** **if**(browser.equalsIgnoreCase("chrome")) {

WebDriverManager.*chromedriver*().setup();

driver = **new** ChromeDriver();

}

**else** **if**(browser.equalsIgnoreCase("edge")) {

WebDriverManager.*edgedriver*().setup();

driver = **new** EdgeDriver();

}

driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.***SECONDS***);

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

driver.get(url);

driver.manage().timeouts().implicitlyWait(30, TimeUnit.***SECONDS***);

**return** driver;

}

**public** **static** **void** quitBrowser(WebDriver driver) {

driver.quit();

}

}

// in Helper class we will create methods for TakesScreenshot, alert, frames, windows, Sync issues, JavaScriptExecutor…

**package com.frame.utility**;

**public class Helper** {

**public static String captureScreenshot(**WebDriver driver) {

TakesScreenshot ts = (TakesScreenshot)driver;

String screenshotPath = System.*getProperty*("user.dir")

+"./screen\_shots/FreeCRM\_" +*getDate\_Time*()+".png";

**try** {

File sourceFile = ts.getScreenshotAs(OutputType.***FILE***);

FileHandler.*copy*(source, **new** File(screenshotPath));

} **catch** (Exception e) {

System.***out***.println("unable to capture screenshot : "+ e.getMessage());

}

**return** screenshotPath;

}

**public** **static** String getDate\_Time() {

Date date = **new** Date();

DateFormat dateFormat = **new** SimpleDateFormat("MM\_dd\_yyyy\_\_HH\_mm\_ss");

String formatedTime = dateFormat.format(date);

**return** formatedTime;

}

}

**public class LoginPage** {

WebDriver driver;

**public** LoginPage(WebDriver driver) {

**this**.driver = driver;

}

@FindBy(name = "email") WebElement userName;

@FindBy(how = How.xpath, using = "//input[@name = 'password']")

WebElement password;

@FindBy(how = How.xpath, using = "//div[text()= 'Login']")

WebElement logButton;

**public void logintoCRM**(String uid, String pass) {

userName.sendKeys(uid);

password.sendKeys(pass);

logButton.click();

}

}

**package com.frame.pages**;

**public class Base\_class** {

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

**public** **static** XLdataProvider excel;

**public** **static** ConfigDataProvider configData;

**public** **static** ExtentReports report;

**public** **static** ExtentTest logger;

**@BeforeSuite**

**public void setupSuite**() {

Reporter.*log*("setting up reports and test getting ready", **true**); //true will print in console.

excel = **new** XLdataProvider();

configData = **new** ConfigDataProvider();

File file = **new** File("./reports/freeCRM"+Helper.*getDate\_Time*()+".html");

ExtentHtmlReporter htmlReporter = **new** ExtentHtmlReporter(file);

report = **new** ExtentReports();

report.attachReporter(htmlReporter);

logger = report.createTest("MyFirstTest");

|  |
| --- |
| **Example: Extent report 5.**  ExtentReports extent = new ExtentReports();  ExtentSparkReporter spark = new ExtentSparkReporter("target/Spark/Spark.html");  extent.attachReporter(spark);  ExtentTest logger = extent.createTest("MyFirstTest");  Logger = extent.createTest("myTest");  Logger.info("info");  Logger.pass("pass");  Logger.warning("warn");  Logger.skip("skip");  Logger.fail("fail");  Logger.info("info").addScreenCaptureFromPath(imagePath);  ExtentTest test = extent.createTest("MyFirstTest");  // reference image saved to disk  test.fail(MediaEntityBuilder.createScreenCaptureFromPath("img.png").build());  // base64  test.fail(MediaEntityBuilder.createScreenCaptureFromBase64String("base64").build());  extent.flush(); |

Reporter.*log*("setting done and test can be started", **true**);

}

**@BeforeClass**

**public void startup**() {

Reporter.*log*("trying to start Browser", **true**);

driver = BrowserFactory.*startBrowser*(driver, configData.getBrowser(), configData.getLinkUrl());

Reporter.*log*("Browser started and application running", **true**);

}

**@AfterClass**

**public void teardown**() {

BrowserFactory.*quitBrowser*(driver);

Reporter.*log*("Browser closed", **true**);

}

**@AfterMethod**

**public void teardown**(ITestResult result) **throws** Exception {

Reporter.*log*("test is about to end", **true**);

**if**(result.getStatus() == ITestResult.***FAILURE***) {

logger.fail("test failed", MediaEntityBuilder.*createScreenCaptureFromPath*(Helper.*captureScreenshot*(driver)).build());

}

**else** **if**(result.getStatus() == ITestResult.***SUCCESS***) {

logger.pass("test passed", MediaEntityBuilder.*createScreenCaptureFromPath*(Helper.*captureScreenshot*(driver)).build());

}

report.flush();

Reporter.*log*("test completed and reports generated", **true**);

//open Extent report after test is done

String pathOfExtentReport = System.*getProperty*("user.dir")+"\\pathOfReport";

File ExtentReportFile = **new** File(pathOfExtentReport);

**try** {

Desktop.*getDesktop*().browse(ExtentReportFile.toURI());

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

**package test\_package**;

@Test

**public** **class** Execution **extends** Base\_class {

**public** **void** login\_app() {

logger = report.createTest("MyFirstTest");

Reporter.*log*("test will be started", **true**);

Login\_page loginPage = PageFactory.initsElements(driver, Login\_page.**class**);

logger.info("starting application");

loginPage.loginToCRM(excel.getStringData(“Sheet1”, 0, 0), excel.getStringData(“Sheet1”, 0, 1));

logger.info("login success");

//Helper. captureScreenshot(driver); it is in Base\_class.

**package** utility;

**public** **class** Helper {

**static** WebDriver *driver*;

**static** JavascriptExecutor *je*;

**static** Alert *alert*;

**public** Helper(WebDriver driver) {

**this**.*driver* = driver;

*je* = (JavascriptExecutor)driver;

*alert* = driver.switchTo().alert();

}

@Test

**public** **static** **void** setImplicitWait(**long** seconds) {

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

Reporter.*log*("implicit wait is set", **true**);

}

**public** **static** **void** setWaitForElementVisible(**int** seconds, WebElement element) {

WebDriverWait wait = **new** WebDriverWait(*driver*, seconds);

wait.until(ExpectedConditions.*visibilityOf*(element));

Reporter.*log*("explicit wait for visibility of element is set");

}

**public** **static** **void** setWaitForElementClickable(**int** seconds, WebElement element) {

WebDriverWait wait = **new** WebDriverWait(*driver*, seconds);

wait.until(ExpectedConditions.*elementToBeClickable*(element));

Reporter.*log*("explicit wait for visibility of element is set");

}

**public** **static** **void** switchToFrame(**int** frameIndex) {

*driver*.switchTo().frame(frameIndex);

Reporter.*log*("switched to "+frameIndex+" frame");

}

**public** **static** **void** switchToFrame(String frameName) {

*driver*.switchTo().frame(frameName);

Reporter.*log*("switched to "+frameName+" frame");

}

**public** **static** **void** switchToFrame(WebElement frameElement) {

*driver*.switchTo().frame(frameElement);

Reporter.*log*("switched to "+frameElement.toString()+" frame");

}

**public** **static** **void** switchBackToMaintWindow() {

*driver*.switchTo().defaultContent();

Reporter.*log*("switched to parent window");

}

**public** **static** **void** switchToNextWindow() {

Set <String> windows = *driver*.getWindowHandles();

Iterator itr = windows.iterator();

**while**(itr.hasNext()) {

String childWindow = (String) itr.next();

*driver*.switchTo().window(childWindow);

Reporter.*log*("switched to child window", **true**);

}

}

// by naveen.

**public** **static** **boolean** switchToRightWindow(String titleWord) {

Set <String> windows = *driver*.getWindowHandles();

List <String> windowList = **new** ArrayList <String>(windows);

**for**(String window : windowList) {

String windowTitle = *driver*.switchTo().window(window).getTitle();

**if**(windowTitle.contains(titleWord)) {

Reporter.*log*("found right window" , **true**);

Reporter.*log*("window title : "+ *driver*.getTitle(), **true**);

**return** **true**;

}

}

**return** **false**;

}

**public** **static** **void** closeAllChildTabs() {

String mainWindow = *driver*.getWindowHandle();

Set <String> windows = *driver*.getWindowHandles();

List <String> windowList = **new** ArrayList <String>(windows);

**for**(String windo : windowList) {

**if**(!windo.equals(mainWindow)) {

*driver*.switchTo().window(windo).close();

Reporter.*log*("closed all tabs except main tab");

}

}

}

**public** **static** **void** main(String[] args) {

driver.switchTo().window(switchToRightWindow(“Open Cart”));

}

//how to know entire page height and width ?

//right click > inspect > console > type document.body.scrollHeight;

//right click > inspect > console > type document.body.scrollWidth;

**public** **static** **void** scrollPage(**int** pageHeight) {

*je*.executeScript("window.scrollBy(0, pageHeight)");

//je.executeScript("window.scrollTo(0, pageHeight)");

Reporter.*log*("scrolled the page", **true**);

}

**public** **static** **void** scrollPageToElement(WebElement element) {

*je*.executeScript("arguments[0].scrollIntoView(true);", element);

Reporter.*log*("scrolled to the element", **true**);

}

**public** **static** **void** scrollPageToLocation(WebElement element) {

*je*.executeScript("window.scollTo(arguments[0], arguments[1]", element.getLocation().x, element.getLocation().y);

Reporter.*log*("scrolled to the element location", **true**);

}

**public** **static** **void** alertAccept() {

*alert*.accept();

Reporter.*log*("alert accepted", **true**);

}

**public** **static** **void** alertDismiss() {

*alert*.dismiss();

Reporter.*log*("alert dismissed", **true**);

}

**public** **static** **String** getAlertText() {

String alert\_text = *alert*.getText();

Reporter.*log*("alert message : "+alert\_text, **true**);

**Return** alert\_text;

}

**public** **static** **void** alertSendkeys(String message) {

*alert*.sendKeys(message);

Reporter.*log*(message+" is sent to alert", **true**);

}

**public** **static** **void** selectUsingindex(WebElement element, **int** indexValue, String optName) {

Select oslect = **new** Select(element);

oslect.selectByIndex(indexValue);

Reporter.*log*(optName+" is selected", **true**);

}

**public** **static** **void** selectUsingValue(WebElement element, String value, String optName) {

Select oslect = **new** Select(element);

oslect.selectByValue(value);

Reporter.*log*(optName+" is selected", **true**);

}

**public** **static** **void** selectUsingVisivleText(WebElement element, String value, String optName) {

Select oslect = **new** Select(element);

oslect.selectByVisibleText(value);

Reporter.*log*(optName+" is selected", **true**);

}

**public** **static** **void** selectPrintOptions(WebElement element) {

Select oselect = **new** Select(element);

List<WebElement> allOptions = oselect.getOptions();

**for**(WebElement option : allOptions) {

String optionNames = option.getText();

Reporter.*log*("Option : "+ optionNames, **true**);

}

}

}

**================== Extent Reports Integration ======================**

How To Use Extent Report with Selenium WebDriver and Add Screenshot

**public** **class** CaptureScreenshot {

**static** String *screenAbsolutePath*;

**public** **static** String screenshot(WebDriver driver) {

**try** {

TakesScreenshot ts = (TakesScreenshot)driver;

File sourceFile = ts.getScreenshotAs(OutputType.***FILE***);

File destFile = **new** File("./screenshots/"+System.*currentTimeMillis*()+".png");

FileUtils.*copyFile*(sourceFile, destFile);

*screenAbsolutePath* = destFile.getAbsolutePath();

} **catch** (Exception e) {

System.***out***.println("exception : "+e.getMessage());

}

**return** *screenAbsolutePath*;

}

}

**public** **class** ExtentReport {

@Test

**public** **void** facebook() **throws** Exception {

ExtentHtmlReporter reporter = **new** ExtentHtmlReporter(**new** File("./reports/extent.html"));

ExtentReports reports = **new** ExtentReports();

reports.attachReporter(reporter);

ExtentTest logger = reports.createTest("check extent report");

WebDriverManager.*edgedriver*().setup();

WebDriver driver = **new** EdgeDriver();

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

logger.info("browser started");

driver.get("https://facebook.com");

logger.info("facebook page opened");

String imagePath = CaptureScreenshot.*screenshot*(driver);

logger.pass("<b><color=red>"+"testpassed"+"</b></color>")

.addScreenCaptureFromPath(imagePath);

reports.flush();

//open Extent report after test is done

String pathOfExtentReport = System.*getProperty*("user.dir")+"\\pathOfReport";

File ExtentReportFile = **new** File(pathOfExtentReport);

**try** {

Desktop.*getDesktop*().browse(ExtentReportFile.toURI());

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

==🡺 to run pom.xml add plugins to pom.xml

🡺 <build>

<pluginManagement>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M5</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testng.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</pluginManagement>

</build>

**How to run pom.xml from cmd ?**

go to pom.xml path open cmd

mvn test > enter.

**How to publish report with Jenkins:**

1. In Jenkins add HTML publisher plugin..

2. go to project > configure > post build Action > add post build Action > click publish HTML repodrts > will open publish HTML repodrts window > add > enter the project directory.html