**Use the below code for creating driver instance with file downloading facility**

Driver instance code would be the following:

**package** base;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.chrome.ChromeOptions;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.ie.InternetExplorerDriver;

**import** org.openqa.selenium.remote.RemoteWebDriver;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**import** cucumber.api.testng.AbstractTestNGCucumberTests;

**public** **class** DriverInstance **extends** AbstractTestNGCucumberTests {

**private** **static** **final** ThreadLocal<RemoteWebDriver> ***remoteWebdriver*** = **new** ThreadLocal<>();

**private** **static** **final** ThreadLocal<WebDriverWait> ***wait*** = **new** ThreadLocal<>();

**public** **void** setWait() {

***wait***.set(**new** WebDriverWait(getDriver(), 30));

}

**public** WebDriverWait getWait() {

**return** ***wait***.get();

}

**public** **void** setDriver(String browser, **boolean** headless) {

**switch** (browser) {

**case** "chrome":

ChromeOptions options = **new** ChromeOptions();

Map<String, Object> chromePrefs = **new** HashMap<String, Object>();

chromePrefs.put("profile.default\_content\_settings.popups", 0);

chromePrefs.put("download.default\_directory","C:\\Users\\DELL\\eclipse-workspace\\fleet4\\ExportFiles" );

options.setExperimentalOption("prefs",chromePrefs);

Here, above 4 lines are deliberately included to make the downloaded automation files (xls, pdf , graphical pdf files) to reach the export files folder which was created in the maven project called fleet4. Before adding the above 4 lines of code, please create the folder in the maven project and give the name for it .Then go to properties and find the complete path of that folder .Subsequently, paste the folder path in the above 4 lines code.

options.addArguments("--start-maximized");

options.addArguments("--disable-notifications");

options.addArguments("--incognito");

***remoteWebdriver***.set(**new** ChromeDriver(options));

**break**;

**case** "firefox":

***remoteWebdriver***.set(**new** FirefoxDriver());

**break**;

**case** "ie":

***remoteWebdriver***.set(**new** InternetExplorerDriver());

**default**:

**break**;

}

}

**public** RemoteWebDriver getDriver() {

**return** ***remoteWebdriver***.get();

}

}

**Note:** Create a feature file called F10.feature to write the automation flow of download files using gherkin language. Also, you create the class file corresponds to the feature file is so called Export.java under pages package. Then include F10 feature file in the below code (TC001\_validate\_amber\_connect.java under test cases package).

TC001\_validate\_amber\_connect.java code will be the following:

**package** testcases;

**import** org.testng.annotations.BeforeTest;

**import** cucumber.api.CucumberOptions;

**import** testng.api.base.ProjectSpecificMethods;

@CucumberOptions(

plugin= {"rerun:C:\\Users\\DELL\\eclipse-workspace\\fleet4\\target\\failedrerun.txt"},

features={"src/main/java/features/F10\_Export\_File.feature"},

glue= {"pages"}, monochrome=**true**)

**public** **class** TC001\_Validate\_amber\_connect **extends** ProjectSpecificMethods{

@BeforeTest

**public** **void** setValues() {

testcaseName = "Validate Login, SADemo1,Dashboard,Vehicle status,Vehicle type and Logout, Vehicle information";

testDescription ="Validate Login(TC001 to TC006), SADemo1(TC007), Dashboard(TC008 to TC013), Vehicle status(TC014 to TC023), Vehicle type(TC024 to TC033) , Vehicle information(TC034 to TC045),Logout(TC046)";

authors="Anu";

category ="Smoke";

}

}

F10.Feature file will be the following:

#Author: your.email@your.domain.com

Feature: Validation for the Export files

Scenario: Validation for the Export files

#Export xls pdf and graphical pdf files after clicking filter and SADEMO5

When Click Fleet User for files

Then Enter Fleet ID for files as '6174242'

Then Enter the Username for files as 'demouser@amberconnect.co.za'

Then Enter the password for files as 'Amber@2327'

Then Select User Type as Admin for files

Then Select the Privacy Policy checkbox for files

Then click Login button for files

Then Click on the trips for files

Then click on the filter and SADEMO5 and apply for files

Then click on the Download and click on Export in xls

Then click on the Download and click on Export in pdf

Then click on the Download and click on Export in Graphical pdf

#Export xls pdf and graphical pdf files after clicking the map

Then click on the personal and small map for files

Then click on the Export trip and Export in XLS

Then click on the Export trip and Export in pdf

Then click on the Export trip and Click on the Trips Detail log

Export file code under the pages package would be the following:

**package** pages;

**import** design.Locators;

**import** testng.api.base.ProjectSpecificMethods;

**import** **static** org.testng.Assert.*assertEquals*;

**import** java.io.File;

**import** java.util.HashMap;

**import** java.util.Map;

**import** java.util.UUID;

**import** org.openqa.selenium.Alert;

**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.chrome.ChromeOptions;

**import** org.openqa.selenium.interactions.Actions;

**import** org.openqa.selenium.remote.DesiredCapabilities;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**import** org.testng.Assert;

**import** cucumber.api.java.en.And;

**import** cucumber.api.java.en.Given;

**import** cucumber.api.java.en.Then;

**import** cucumber.api.java.en.When;

**public** **class** Export\_file **extends** ProjectSpecificMethods {

WebDriver driver;

File folder;

//Export xls pdf and graphical pdf files after clicking filter and SADEMO5

@Given("Click Fleet User for files")

**public** Export\_file clickFleetuser() {

click(locateElement(Locators.***XPATH***, "//\*[@src='images/fleet-user.svg']"));

reportStep("Fleet user is clicked successfully", "pass");

**return** **this**;

}

@When("Enter Fleet ID for files as {string}")

**public** Export\_file enterfleetID(String data) {

clearAndType(locateElement(Locators.***ID***, "fleetId"), data);

reportStep(data + " entered successfully", "pass");

**return** **this**;

}

@And("Enter the Username for files as {string}")

**public** Export\_file enterUsername(String data) {

clearAndType(locateElement(Locators.***ID***, "username"), data);

reportStep(data + " entered successfully", "pass");

**return** **this**;

}

@And("Enter the password for files as {string}")

**public** Export\_file enterPassword(String data) {

clearAndType(locateElement(Locators.***ID***, "password"), data);

reportStep(data + " entered successfully", "pass");

**return** **this**;

}

@And("Select User Type as Admin for files")

**public** Export\_file selectAdmin() {

clickWithNoSnap(locateElement(Locators.***ID***, "radio-00"));

reportStep("Admin userType is Selected successfully", "pass");

**return** **this**;

}

@Then("Select the Privacy Policy checkbox for files")

**public** Export\_file selectagreeCheck(){

clickUsingJs(locateElement(Locators.***ID***, "agreeCheck"));

reportStep(" Privacy Policy checkbox is selected successfully", "pass");

**return** **this**;

}

@Then("click Login button for files")

**public** HomePage PickclickLogin() {

click(locateElement(Locators.***CLASS\_NAME***, "loginBtn"));

reportStep("click Login button for files is successfull", "pass");

**return** **new** HomePage();

}

@Then("Click on the trips for files")

**public** Export\_file ClickTrips() **throws** Exception

{

click(locateElement(Locators.***XPATH***, "(//\*[contains(@data-placement,'bottom')])[1]"));

click(locateElement(Locators.***XPATH***, "//\*[contains(@alt,'trips')]"));

reportStep("Click on the trips for files is successful", "pass");

**return** **this**;

}

@Then("click on the filter and SADEMO5 and apply for files")

**public** Export\_file ClickSADemo5() **throws** Exception

{

Thread.*sleep*(10000);

click(locateElement(Locators.***XPATH***, "//\*[contains(@class,'headerFilterIcon')]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "(//\*[contains(@name,'vehiclelist')])[7]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//a[@class='yes-btn show-btn']"));

reportStep("click on the filter and SADEMO5 and apply for files is successfull", "pass");

**return** **this**;

}

@Then("click on the Download and click on Export in xls")

**public** Export\_file ClickExport() **throws** Exception

{

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "(//\*[contains(@alt,'downloads')])[1]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//a[@onclick=\"exportTrip('xls')\"]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//\*[contains(@class,'PopupClassifyTitle')]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//p[text()='PERSONAL']"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//\*[contains(@class,'yes-btn muliselectbtn')]"));

reportStep("click on the Download and click on Export in xls is successfull", "pass");

**return** **this**;

}

@Then("click on the Download and click on Export in pdf")

**public** Export\_file ClickExportPdf() **throws** Exception

{

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "(//\*[contains(@alt,'downloads')])[1]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//a[normalize-space()='Export in PDF']"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//\*[contains(@class,'PopupClassifyTitle')]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//p[text()='PERSONAL']"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//\*[contains(@class,'yes-btn muliselectbtn')]"));

reportStep("click on the Download and click on Export in pdf is successfull", "pass");

**return** **this**;

}

@Then("click on the Download and click on Export in Graphical pdf")

**public** Export\_file ClickExportGraphicalPdf() **throws** Exception

{

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "(//\*[contains(@alt,'downloads')])[1]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//a[normalize-space()='Export in Graphical PDF']"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//\*[contains(@class,'PopupClassifyTitle')]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//p[text()='PERSONAL']"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//\*[contains(@class,'yes-btn muliselectbtn')]"));

reportStep("click on the Download and click on Export in Graphical pdf is successfull", "pass");

**return** **this**;

}

//Export xls pdf and graphical pdf files after clicking the map

@Then("click on the personal and small map for files")

**public** Export\_file ClickExportSmallMap() **throws** Exception

{

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "(//\*[contains(@data-id,'Personal')])[1]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "(//\*[contains(@class,'col-lg-4 left-map-image noPadColBoot')])[1]"));

reportStep("click on the small map for files is successfull", "pass");

**return** **this**;

}

@Then("click on the Export trip and Export in XLS")

**public** Export\_file ClickExportTripXLS() **throws** Exception

{

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "(//\*[contains(@data-toggle,'dropdown')])[6]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "(//\*[contains(@data-translate,'\_page\_genenaral\_export\_xls')])[1]"));

reportStep("click on the Export trip and Export in XLS is successfull", "pass");

**return** **this**;

}

@Then("click on the Export trip and Export in pdf")

**public** Export\_file ClickExportTripPdf() **throws** Exception

{

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "(//\*[contains(@data-toggle,'dropdown')])[6]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "//\*[contains(@data-translate,'\_page\_genenaral\_export\_pdf')]"));

reportStep("click on the Export trip and Export in pdf is successfull", "pass");

**return** **this**;

}

@Then("click on the Export trip and Click on the Trips Detail log")

**public** Export\_file ClickTripsDetail() **throws** Exception

{

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "(//\*[contains(@data-toggle,'dropdown')])[6]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "(//\*[contains(@class,'normalExport')])[3]"));

Thread.*sleep*(8000);

click(locateElement(Locators.***XPATH***, "(//\*[contains(@src,'images/close.png')])[10]"));

reportStep("click on the Export trip and Click on the Trips Detail log is successfull", "pass");

**return** **this**;

}

}