**Taking screenshots whenever the test fails:**

We use again listeners to take screenshots. We implement the method onTestFailure(), to do this.

Before doing this, we need to save the screenshots in a folder. Let’s put this folder path in properties file.

I have added the following path in properties file.

FAILED\_SCREENSHOTS\_PATH = C:\\Users\\padal\\eclipse-workspacenew\\HybridKeywordDriven\\FailedScreenshots\\

The following is the full code of properties file.

WB\_PATH\_TESTS = C:\\Users\\padal\\eclipse-workspacenew\\HybridKeywordDriven\\tests\\tests\_main.xlsx

WB\_PATH\_TESTS\_SHEET = TestCases

WB\_PATH\_TEST\_CASES = C:\\Users\\padal\\eclipse-workspacenew\\HybridKeywordDriven\\tests\\

MASTERKW\_PATH = C:\\Users\\padal\\eclipse-workspacenew\\HybridKeywordDriven\\data\\masterkeywords.xlsx

MASTERKW\_SHEET = globalkeywords

DATA\_PATH = C:\\Users\\padal\\eclipse-workspacenew\\HybridKeywordDriven\\data\\data.xlsx

**FAILED\_SCREENSHOTS\_PATH = C:\\Users\\padal\\eclipse-workspacenew\\HybridKeywordDriven\\FailedScreenshots\\**

SPECIAL\_KEYWORDS = adminpostvehicle,adminlogin,adminpostbrand,registeruser,userlogin,subscribeuser

CHECK\_KEYWORDS = checkusersdifference,checkvehiclesdifference,checkbookingsdifference,checkbrandsdifference,checksubscribersdifference,checkqueriesdifference,checktestimonialsdifference

BOOKING\_VEHICLES = userclickvehicledetails

ADMIN\_UNAME = admin

ADMIN\_PWD = Test@12345

USER\_UNAME = subbu123

USER\_PWD = subbu123

Now let’s update the onTestFailure() method of TestNG listener, to capture screenshots.

@Override

**public** **void** onTestFailure(ITestResult result){

String testcase = (String)(result.getParameters())[0];

WebDriver driver = driverHolding.*mapdriver*.get(testcase+"driver");

Properties gldata = **new** Properties();

InputStream input = **null**;

**try** {

input = **new** FileInputStream("src/executionEngine/config.properties");

} **catch** (FileNotFoundException e1) {

e1.printStackTrace();

}

**try** {

gldata.load(input);

} **catch** (IOException e1) {

e1.printStackTrace();

}

String screenshotspath = gldata.getProperty("FAILED\_SCREENSHOTS\_PATH");

File src = ((TakesScreenshot)driver).getScreenshotAs(OutputType.***FILE***);

**try** {

FileUtils.*copyFile*(src, **new** File(screenshotspath+testcase+".png"));

} **catch** (IOException e) {

e.printStackTrace();

}

driver.quit();

driverHolding.*mapdriver*.remove(testcase+"driver");

System.***out***.println(testcase + " failed.");

}

Here we loaded the properties file and read the property of failed testcases path into screenshotpath. Then we used TakesScreenshot method to take screenshots and save them.

But here the problem is TakesScreenshot doesn’t take the screenshot of alerts present on the screen. So you have to use another method of taking screenshot from AWT class.

We developed another class for this and call this class method in onTestFailure.

**package** executionEngine;

**import** java.awt.Rectangle;

**import** java.awt.Robot;

**import** java.awt.Toolkit;

**import** java.awt.image.BufferedImage;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.util.Properties;

**import** javax.imageio.ImageIO;

**public** **class** captureScreenshot {

**public** String CaptureScreenShotWithTestStepNameUsingRobotClass(String testCaseName) **throws** IOException

{

Properties gldata = **new** Properties();

InputStream input = **new** FileInputStream("src/executionEngine/config.properties");

gldata.load(input);

String screenshotspath = gldata.getProperty("FAILED\_SCREENSHOTS\_PATH");

**try**{

// Creating Robot class object

Robot robotClassObject = **new** Robot();

// Get screen size

Rectangle screenSize = **new** Rectangle(Toolkit.*getDefaultToolkit*().getScreenSize());

// Capturing screenshot by providing size

BufferedImage tmp = robotClassObject.createScreenCapture(screenSize);

// Defining destination file path

String path=screenshotspath+testCaseName+System.*currentTimeMillis*()+".png";

// To copy temp image in to permanent file

ImageIO.*write*(tmp, "png",**new** File(path));

**return** path;

}**catch**(Exception e)

{

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

**return** "";

}

}

}

We call the method CaptureScreenShotWithTestStepNameUsingRobotClass(), from onTestFailure().

@Override

**public** **void** onTestFailure(ITestResult result){

// **TODO** Auto-generated method stub

String testcase = (String)(result.getParameters())[0];

WebDriver driver = driverHolding.*mapdriver*.get(testcase+"driver");

captureScreenshot cs = **new** captureScreenshot();

**try** {

cs.CaptureScreenShotWithTestStepNameUsingRobotClass(testcase);

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

driver.quit();

driverHolding.*mapdriver*.remove(testcase+"driver");

System.***out***.println(testcase + " failed.");

}

But the problem with this is the browser should be in visible state otherwise it will take the screenshot of whatever thing is visible on the screen.