**Page Object Model**

**This is not a framework. This is an approach, design pattern. We will integrate with data driven framework. We will use TestNG to write the test cases.**

**->For each and every page, we create separate java class. -**

**-> OR – Collection of Objects.**

**Base class**

* **Maximize window**
* **Pageload TimeOut**
* **Implicit wait**
* **Delete All cookies**
* **Get(url)**
* **Close**
* **Quit**

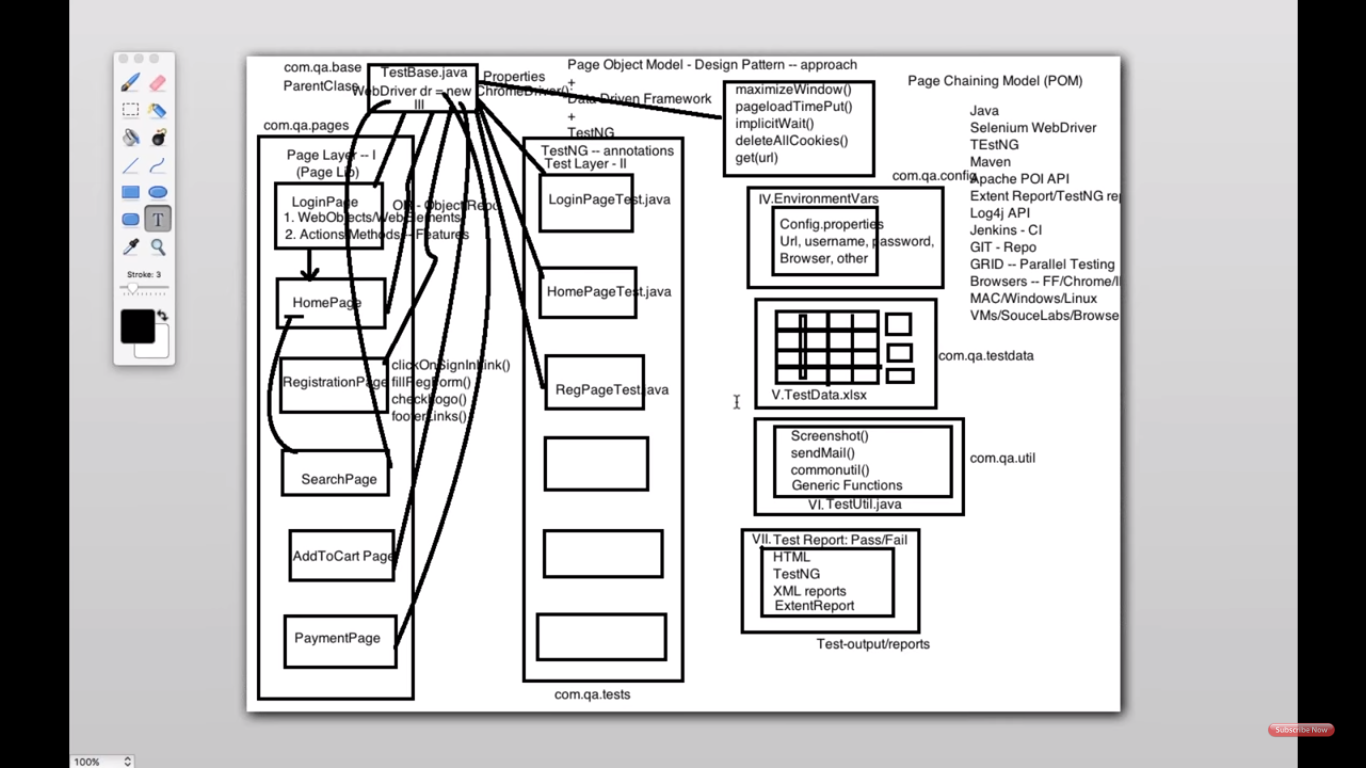
**Excel file – Apache Poi API**

**Utility file**

* **Screenshot**
* **Send mail**

**Login page redirects to home page. Home page is landing page of login page.**

**Page Object model is also called page chaining model.**



**@CacheLookup**

@FindBy(xpath="//td[contains(text(),'Contacts')]")

@CacheLookup

**public** WebElement contactsLabel;

it says It will save the contactsLabel in a memory . Everytime you are interacting with this element , it will get from cache instead of page . Performance of script is improved. Everytime it will try tointeract the page //td containing . So to avoid this we use cachelookup.

Disadvantage : if the page is refreshed , the cache memory will be corrupted.

You should not use for all the element. Script will be fast

WebDriver Fire event

1 . WebDriver Fire event : To generate selenium action logs .

You need to add WebEvent listener in util package

2. Then in TestUtils class , add the function takeScreenshotAtEndOfTest()

**public** **static** **void** takeScreenshotAtEndOfTest() **throws** IOException {

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

String currentDir = System.*getProperty*("user.dir");

FileUtils.*copyFile*(scrFile, **new** File(currentDir + "\\screenshots\\" + System.*currentTimeMillis*() + ".png"));

}

**Throwable means it will handle both exception and errors.**

**No need to write if else condition , if error take take screenshot.It will automatically take screenshot**

3 . Goto Base class , you need to create an object for EventfiringWebDriverClass

4 . In the base class , add the below line

*e\_driver* = **new** EventFiringWebDriver(*driver*);

// Now create object of EventListerHandler to register it with EventFiringWebDriver

*eventListener* = **new** WebEventListener();

*e\_driver*.register(*eventListener*);

*driver* = *e\_driver*;

**public** **static** EventFiringWebDriver *e\_driver*;

**public** **static** WebEventListener *eventListener*;

Extent Report

1 . Add extent report dependency in pom.xml

<!-- https://mvnrepository.com/artifact/com.relevantcodes/extentreports -->

<dependency>

<groupId>com.relevantcodes</groupId>

<artifactId>extentreports</artifactId>

<version>2.41.2</version>

</dependency>

2 Create a package com.cr.qa.ExtentReportListener in src/main/java

3. In the package , paste the class ExtentReporterNG

Extent Report depends on testNG.

4 , If you want to change the extent report name – change Extent.html to some other name.

extent = **new** ExtentReports(outputDirectory + File.***separator***

+ "Extent.html", **true**);

5 .Listener will listen each and every activity of your execution.

Add listener class in testng.xml

<listeners>

<listener class-name=*"com.crm.qa.ExtentReportListener.ExtentReporterNG"* />

</listeners>

Add screenshot in Extent Report

1 . Add the below code in TestBase.class

**public** **static** ExtentReports *extent*;

**public** **static** ExtentTest *extentTest*;

@BeforeTest

**public** **void** setExtent(){

extent = **new** ExtentReports(System.*getProperty*("user.dir")+"\\test-output\\ExtentReport.html", **true**);

extent.addSystemInfo("Host Name", "Ananth Windows");

extent.addSystemInfo("User Name", "Ananth Kumar");

extent.addSystemInfo("Environment", "QA");

@AfterTest

**public** **void** endReport(){

extent.flush();

extent.close();

}

**public** **static** String getScreenshot(WebDriver driver, String screenshotName) **throws** IOException{

String dateName = **new** SimpleDateFormat("yyyyMMddhhmmss").format(**new** Date());

TakesScreenshot ts = (TakesScreenshot) driver;

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

// after execution, you could see a folder "FailedTestsScreenshots"

// under src folder

String destination = System.*getProperty*("user.dir") + "/FailedTestsScreenshots/" + screenshotName + dateName

+ ".png";

File finalDestination = **new** File(destination);

FileUtils.*copyFile*(source, finalDestination);

**return** destination;

}

ITestResult is coming from TestNG.

The pass / failed test cases will be stored in result object.

I have to attach screenshot for only failed test cases.

For failure ,failure test cases are returned as 2 (get status)

pass -1

skip – 3

Result.getName wll give u which the method name which got failed.

2 . In the test case class , add the below line in @After method

@AfterMethod

**public** **void** tearDown(ITestResult result) **throws** IOException {

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

extentTest.log(LogStatus.***FAIL***, "TEST CASE FAILED IS "+result.getName()); //to add name in extent report

extentTest.log(LogStatus.***FAIL***, "TEST CASE FAILED IS "+result.getThrowable()); //to add error/exception in extent report

String screenshotPath = *getScreenshot*(*driver*, result.getName());

extentTest.log(LogStatus.***FAIL***, extentTest.addScreenCapture(screenshotPath)); //to add screenshot in extent report

//extentTest.log(LogStatus.FAIL, extentTest.addScreencast(screenshotPath)); //to add screencast/video in extent report

}

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

extentTest.log(LogStatus.***SKIP***, "Test Case SKIPPED IS " + result.getName());

}

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

extentTest.log(LogStatus.***PASS***, "Test Case PASSED IS " + result.getName());

}

extent.endTest(extentTest); //ending test and ends the current test and prepare to create html report

*driver*.quit();

}

3 . In each test case , add the below line

extentTest = extent.startTest("validateCreateNewContact");

How to Send Mail

<http://learn-automation.com/send-report-through-email-in-selenium-webdriver/>

step 1 : Add 3 files in the package com.qtpselenium.facebook.pom.mail

step 2 : Add the dependency in the pom.xml file

<dependency>

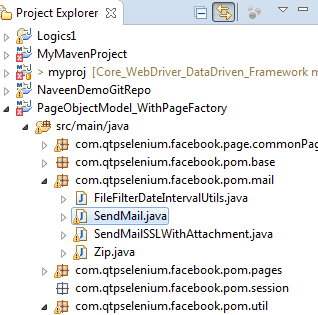
<groupId>javax.mail</groupId>

<artifactId>mail</artifactId>

<version>1.4</version>

</dependency>

Step3:



Step 4 : Make necessary changes in SendMailSSLWithAttachment - this file will help you send the mail

This below function will get the latest generated report in D:\report

**private** **static** File getLatestFilefromDir(String dirPath){

File dir = **new** File(dirPath);

File[] files = dir.listFiles();

**if** (files == **null** || files.length == 0) {

**return** **null**;

}

File lastModifiedFile = files[0];

**for** (**int** i = 1; i < files.length; i++) {

**if** (lastModifiedFile.lastModified() < files[i].lastModified()) {

lastModifiedFile = files[i];

}

}

**return** lastModifiedFile;

}

Step 5 :Create @AfterSuite and call the class to send the mail .

I have created this in login test.

@AfterSuite

**public** **void** sendemail(){

SendMailSSLWithAttachment.*sendEMail*();

test.log(LogStatus.***INFO***, "Emil sent sucessfully to the recipients");

}

If the mail is not sent , open the link and turn on the settings

<https://myaccount.google.com/lesssecureapps?pli=1>