22-03-2017

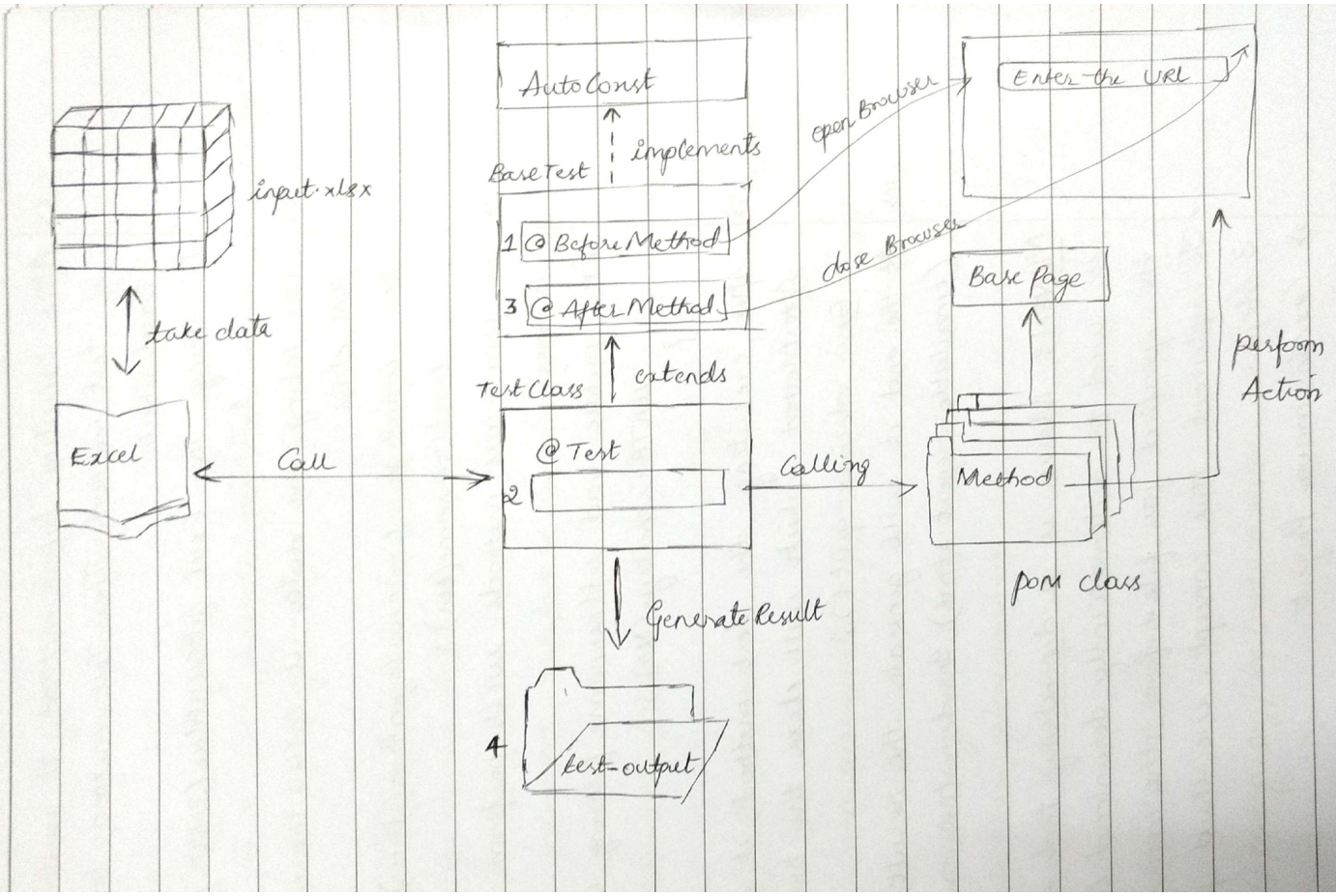
AUTOMATION FRAMEWORK ARCHITECTURE

1) In our framework, we have used methods to avoid repetition of the code, hence it is called as method driven Automation framework
2) We are taking multiple inputs to test the application thoroughly using Excel. Hence it is also called as data driven automation framework
3) Combination of any two framework is called as Hybrid framework
4) In the framework for every test case we develop test class using testNG. and all the test classes extends BaseTest class which implements AutoConst interface.
5) BaseTest class has BeforeMethod & AfterMethod, hence during runtime, it will execute Before Method, which opens the Browser and enters the URL.

Ex:

```
System.setProperty(GECKO-KEY, GECKO-VALUE);
WebDriver driver = new ChromeDriver();
driver.get("http://localhost/login.do");
```

Scanned by CamScanner



Scanned by CamScanner

6) After executing before method, it will start the execution of test method. The Test method takes data from Excel sheet.

Ex. String un = Excel.getCellValue(INPUT_PATH, "ValidLoginLogout", 1, 0);

7) After taking the data, it calls the method present in pom class.

Ex: LoginPage l = new LoginPage(driver);
l.setUsername(un);

8) The pom class methods will perform action on the browser

Ex. public void setUsername(String un){
unTB.sendkeys(un);
}

9) After executing the test method, it will execute, AfterMethod which will close the browser.

Ex. driver.quit();

10) At the end it will generate the result in HTML format (emailable-report.html) Inside test-output folder.

MAVEN: Maven is a dependency tool. In selenium it is used to automatically download latest version of the specified jar file before the execution of the framework.

→ To implement this concept, we shud perform following 3 steps:

1) Create maven Project
2) Specify dependency
3) Run Maven project.

Scanned by CamScanner

1. Creating Maven Project:

→ Right Click on the java project
→ Configure
→ Convert to Maven Project
→ Finish ⇒ It will generate pom.xml inside java project.
(POM stands for project object model.)

2. Specify the Dependency: In pom.xml.

→ Go to dependencies tab
→ Click on add
→ Specify groupId, artifactId and version.
→ Click OK.
→ Save pom.xml file.

dependency #1: Selenium.

http://www.seleniumhq.org/download/maven.jsp

<groupId> org.seleniumhq.selenium
<artifactId> selenium-java
<version> 3.3.1

dependency #2: testNG

http://testng.org/doc/maven.html

<groupId> org.testng
<artifactId> testng
<version> 6.10

Scanned by CamScanner

dependency #3: POI

http://mvnrepository.com/artifact/org.apache.poi/poi

<groupId> org.apache.poi
<artifactId> poi
<version> 3.15

<groupId> org.apache.poi
<artifactId> poi-ooxml
<version> 3.15

3. Run Maven Project:

To run the maven project, we right click on pom.xml. Go to Run as > and select MavenTest.

During runtime, it performs following steps.

a) Compare the version of the jar files ~~with~~ present in maven local repository with ~~the~~ version present in maven centralized repository. If there is any latest version available, it will downloads into repository(local)

b) Then it will compile the java code (all the java files present under src) using maven compiles plugin

c) After the compilation it should execute testNG.xml

* for this we should use surefire plugin available in following url.

=> http://maven.apache.org/surefire/maven-surefire-plugin/

Scanned by CamScanner

__/__/____

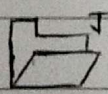
copy the plugin content → Go to pom.xml → paste it below maven-compiles-plugin (bellow </plugin> ; line#17)
d> Save pom.xml file.

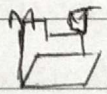
⇒ After specifying the surefire plugin → Right click on pom.xml → go to Run as → select → Maven Test / Maven Install

⇒ Result will be present under "./target/surefire-reports/emailable-report.html"

P.Q-X: What are the differences b/w Java Project & Maven Project.

JAVA PROJECT	MAVEN PROJECT
1) We should download jar file manually.	1) It will be downloaded automatically
2) location of downloaded files will be "jars"	2) ~~Java~~ .m2/repository.
3) Jars files will be listed under "Referenced libraries"	3) Jar files will be listed under Maven Dependencies.
4) We execute using testng.xml	4) We execute using pom.xml.
5) Result location test-output folder.	5) Result location target folder.





Scanned by CamScanner

SELENIUM GRID:

1) Testing the application in different environment, is called as compatibility testing
2) Selenium supports only web application and we test the web application on different browsers and operating systems.
3) The framework will be stored and configured in the centralized system called HUB & It will be executed on multiple remote systems called NODE

Configuring the NODE:

→ Required Softwares

a) JDK
b) Browser
c) Brauser specific driver executable
d) Selenium Server standalone jar file.

Step1: Create a folder in the required location and copy paste .exe & .jar files

a) chromedriver.exe (cd.exe)
b) geckodriver.exe (gc.exe)
c) Selenium Server Standalone jar file (sss.jar)

Step2: In the same location create a text file and type the following command.

```
java -Dwebdriver.chrome.driver=cd.exe -Dwebdriver.
gecko.driver=gc.exe -jar sss.jar
```

Scanned by CamScanner

Save & close the file. Rename the file as RunMe.bat

Step3: Double click on the batch file. It should display following message -

Selenium Server is up and Running.

Configuring the HUB: ~~in the from~~

~~Step~~ In the framework, update BeforeMethod present in BaseClass as shown below and execute it

```
@BeforeMethod
public void openApplication() throws MalformedURLException {
URL rsURL = new URL("http://192.168.1.24/4444/wd/hub");
DesiredCapabilities d= new DesiredCapabilities();
d.setBrowserName("chrome");
driver = new RemoteWebDriver(rsURL, d);
driver.get("http://localhost/login.do");
driver.manage().timeouts().implicitlyWait(10, TimeUnits.SECONDS);
}
```

EXECUTING THE SCRIPTS ON MULTIPLE SYSTEM PARALLELY:

To implement this concept, we use selenium grid along with testNG parallel option.

* Sending data from testNG.xml into testNG class.

1> Create a testNG class as shown below -

Scanned by CamScanner

```
package qsp;

public class Demo {
   @Parameters({"subject"})
   @Test
   public void testA(String s) throw InterruptedException{
     for(int i=1; i<=3; i++) {
      Reporter.log(s, true);
      Thread.sleep(500); }
   }
}
```

NOTE: If any test class contains parameters annotations, then execute it from testng.xml or else we get TestNGException.

Content of testNG.xml

```
<suite name="Suite" parallel="tests">
    <test name="Test1">
      <parameter name="subject" value="java" />
      <classes>
        <class name="qsp.Demo" />
      </classes>
    </test>
    <test name="Test2">
      <parameter name="subject" value="selenium" />
      <classes>
       <class name="qsp.Demo" />
```

Scanned by CamScanner

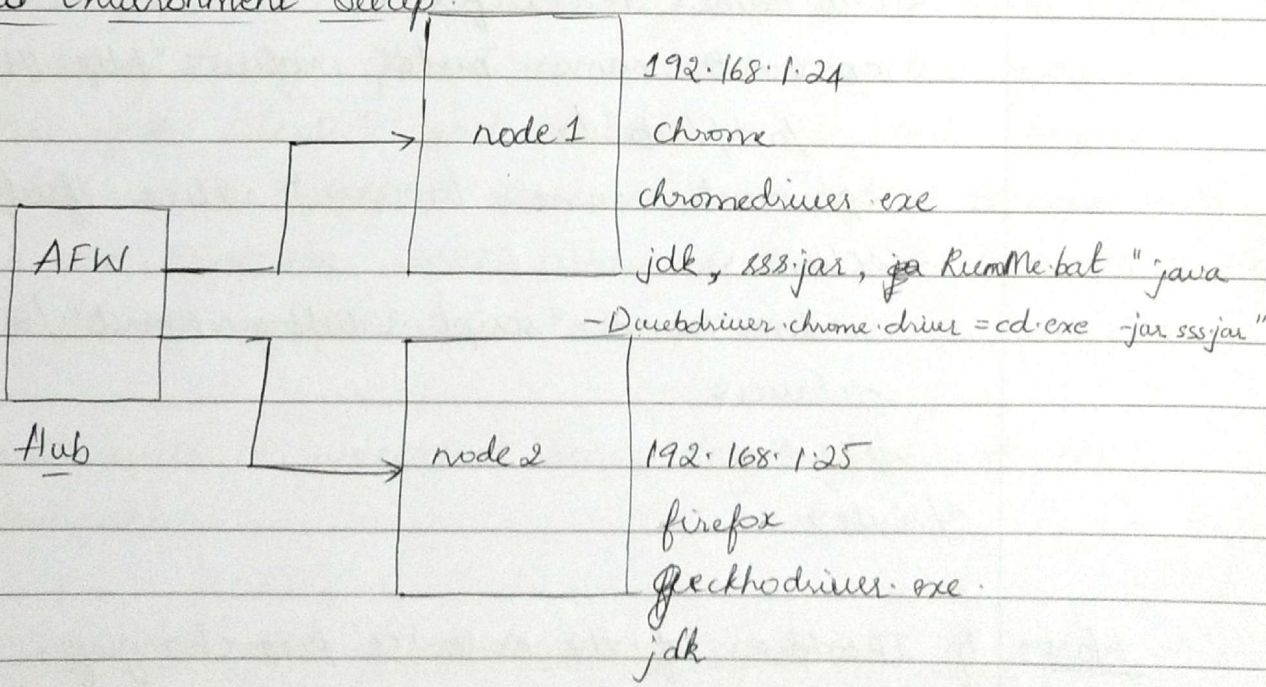
</classes>
</test>
</suite>

o/p : java
selenium
java
selenium
java
selenium

24-03-2017

EXECUTING FRAMEWORK ON MULTIPLE SYSTEM, MULTIPLE BROWSERS PARALLELY :

Grid Environment Setup :



RunMe.bat → java -Dwebdriver.gecko.driver = ge.exe
-jar sss.jar.

Scanned by CamScanner

⇒ Manually run the batch file in both node1 & node2. and ensure that it has displayed the message "Selenium Server is Up & Running".

⇒ Update testng.xml as shown below.

```
<suite name = "Suite" parallel = "tests">
  <test name = "TestChrome">
    <parameter name = "node" value = "http://192.168.1.24:4444/
      wd/hub" />
    <parameter name = "browser" value = "chrome" />
    <classes>
      <class name = "script.ValidLoginLogout" />
    </classes>
  </test>
  <test name = "TestFirefox">
    <parameter name = "node" value = "http://192.168.1.25:4444
      /wd/hub" />
    <parameter name = "browser" value = "firefox" />
    <classes>
      <class name = "script.ValidLoginLogout" />
    </classes>
  </test>
</suite>
```

NOTE: If IP address of the node is keep changing, then we can use name of the computer in place of ip address

Ex: http://(compname):4444/wd/hub

Scanned by CamScanner

⇒ Update BeforeMethod present in BaseTest:

```
@Parameters ({"node", "browser"})
@BeforeMethod
public void openApplication (String node, String browser) throws
    MalformedURLException
{   URL rsURL = new URL(node);
    DesiredCapabilities dc = new DesiredCapabilities();
    dc.setBrowserName(browser);
    driver = new RemoteWebDriver(rsURL, dc);
    driver.get("http://localhost/login.do");
  driver.manage().timeouts().implicitlyWait(10, TimeUnits.Seconds);
}
```

⇒ Execute pom.xml which internally executes testng.xml after upgrading the jar files, since we have mention parallel option for "tests" and these are 2 test blocks, testNG will create 2 threads, the first thread will run the script on node1, where as the second thread will execute the script on node2 at the same time.

⇒ Result will be generated inside target folder of the framework.

Capturing Screenshot when the script fails :-

Step1: Create a folder with the name 'screenshot' inside the java project.

Scanned by CamScanner

Step2: Specify its path in 'AutoConst' Interface

Ex: String IMG_PATH = "./screenshots/";

Step3: Create a class inside generic package as shown below

Ex:

```
package generic;
import java.util.Date;
public class Photo {
public static void getScreenShot(WebDriver driver,
        String folderPath) {
SimpleDateFormat s = new SimpleDateFormat("MM-dd-yyyy
        hh-mm-ss");
String datetime = s.format(new Date());
TakesScreenshot t = (TakesScreenshot) driver;
File SrcFile = t.getScreenshotAs(OutputType.FILE);
File destFile = new File(folderPath+datetime+".png");
try{
    FileUtils.copyFile(srcFile, destFile);
  }
catch(Exception e) { }
  }
}
```

png ⇒ portable network graphics.

Step4: Update AfterMethod present in BaseTest as shown below

Scanned by CamScanner

```
@AfterMethod
public void closeApplication (ITestResult r){
    if (r.getStatus() == 2) {
        photo.getScreenShot(driver, IMG_PATH);
    }
    driver.quit();
}
```

getStatus == 1 ⇒ PASS
getStatus == 2 ⇒ FAIL

Step5: Run pom.xml and it will take screenshot only when script is failed.

25-03-2017

GITHUB:

It is an online repository used by developers (application developers and automation developers) to store the source code. With respect to Selenium, we use github to store and share the automation framework. Initially automation lead will design the framework and they will upload it to github server.

All the automation engineers will download the framework from github server in the local system and everyday, they will write a script and upload it to github server (push) and they will download the script written by others for review and execution purpose (pull).

Uploading the Framework:

This step is done by lead after designing the framework (only one time)

Scanned by CamScanner

Step1: Login to GitHub.com

Step2: Click on Start a project

Step3: Specify the repository name 'BCSM13'

Step4: Select private.

Step5: Click on create repository.

Step6: It generates URI (Uniform Resource ~~Information~~ Identifier)

Ex: https://github.com/javagalbhanu/BCSM13.git

Step7: In eclipse, right click on java project, Go to Team, Select share project, select Git & click next — click on create; specify a folder name at the end. → Finish → Finish

NOTE The above step will create local repository of github.

Step8: Right click on the java project, Go to Team & select commit. Specify the comment; Select all the checkbox; click on ~~Commit~~ Commit & push

Step9: Specify the URI. It will automatically display host & repository path. Specify username & password. Click next. Select HEAD → Click on Add Spec → Finish → Click OK

Scanned by CamScanner

If update fails, select 'add all branch specs' and click on force update.

Downloading Framework from GitHub to Local System

This step will be done by all the automation engineers only one time.

Step 1: In eclipse go to file → Import → Git → Projects from Git → Next → clone URI → Next → Specify URI → username/pwd → Next → Next → Next → Ensure that first radio button is selected (Import existing projects) → Next → Finish.

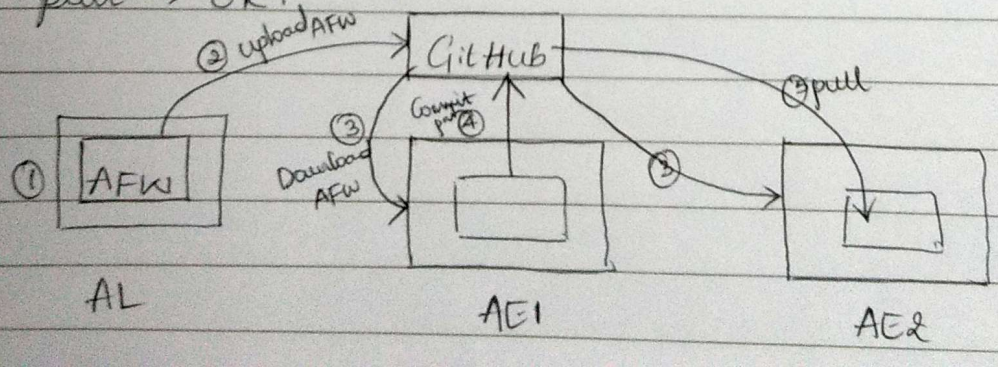
Uploading file to GitHub:

Step 1: Right click on file/folder → go to Team → Select Commit → specify the comment → select checkbox → Commit & push → OK

If the file is modified, then also we should follow the same procedure.

Downloading files from Github:

Step 1: Right click on the java project → Go to Team → select pull → OK.



Scanned by CamScanner