Src main Folder:

PageObject package:

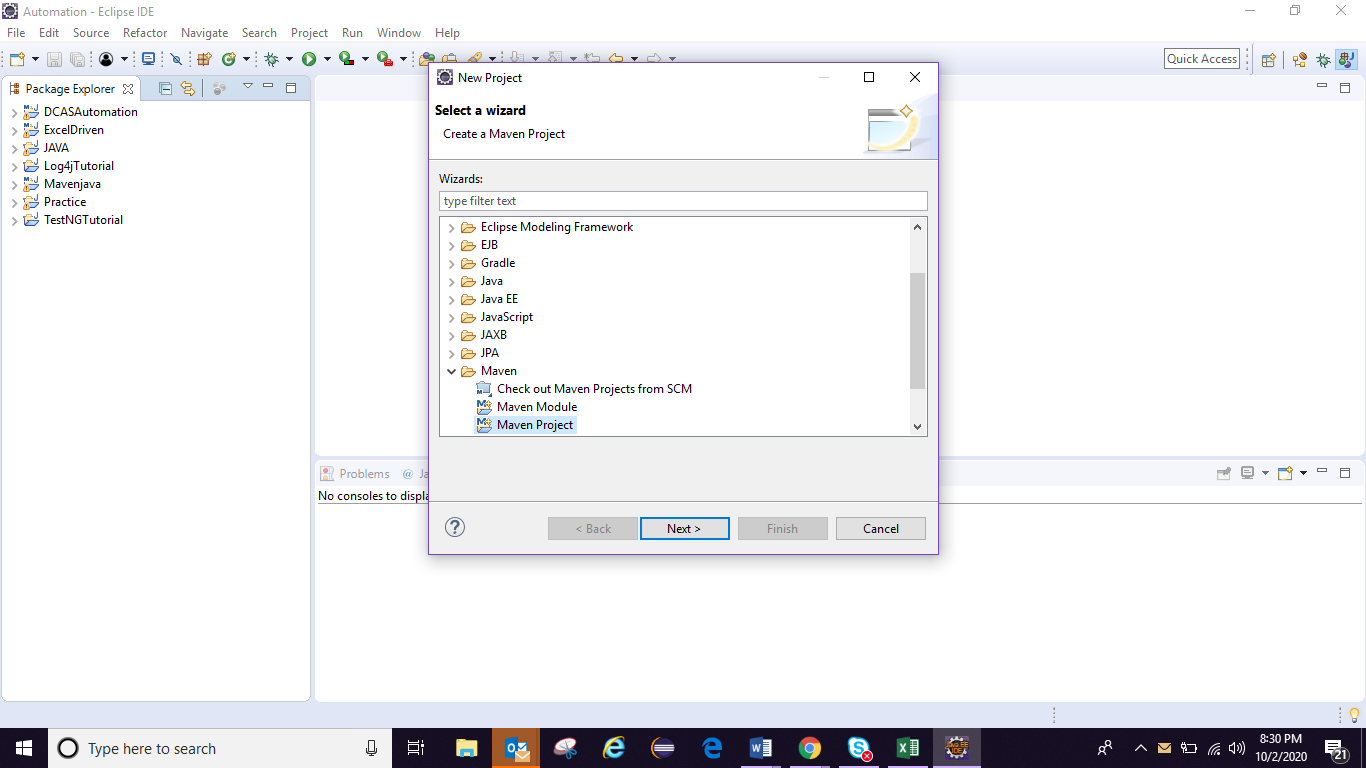
1. Inside PageObject pages we should write all object of webElement and write locators of WebElement:
2. We have to create PageObject page for each each page.

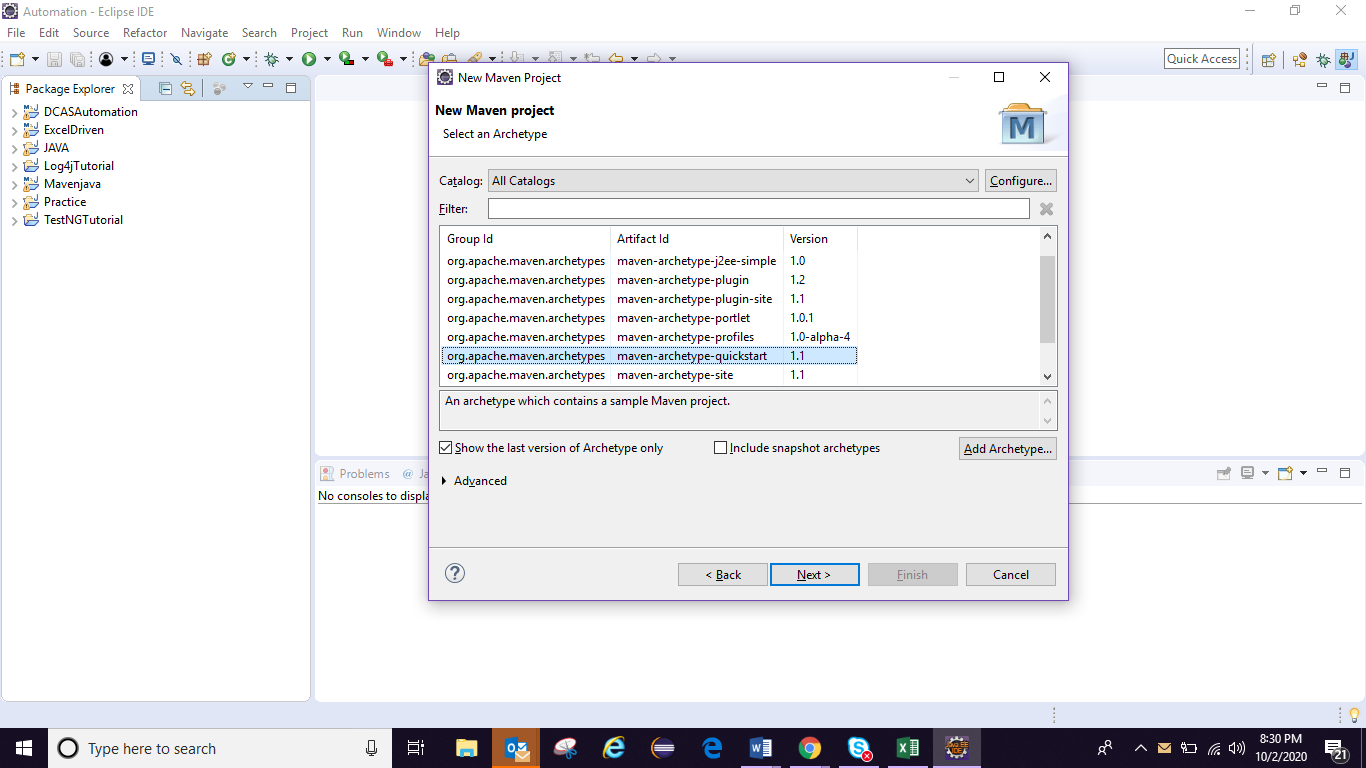
Resource package:

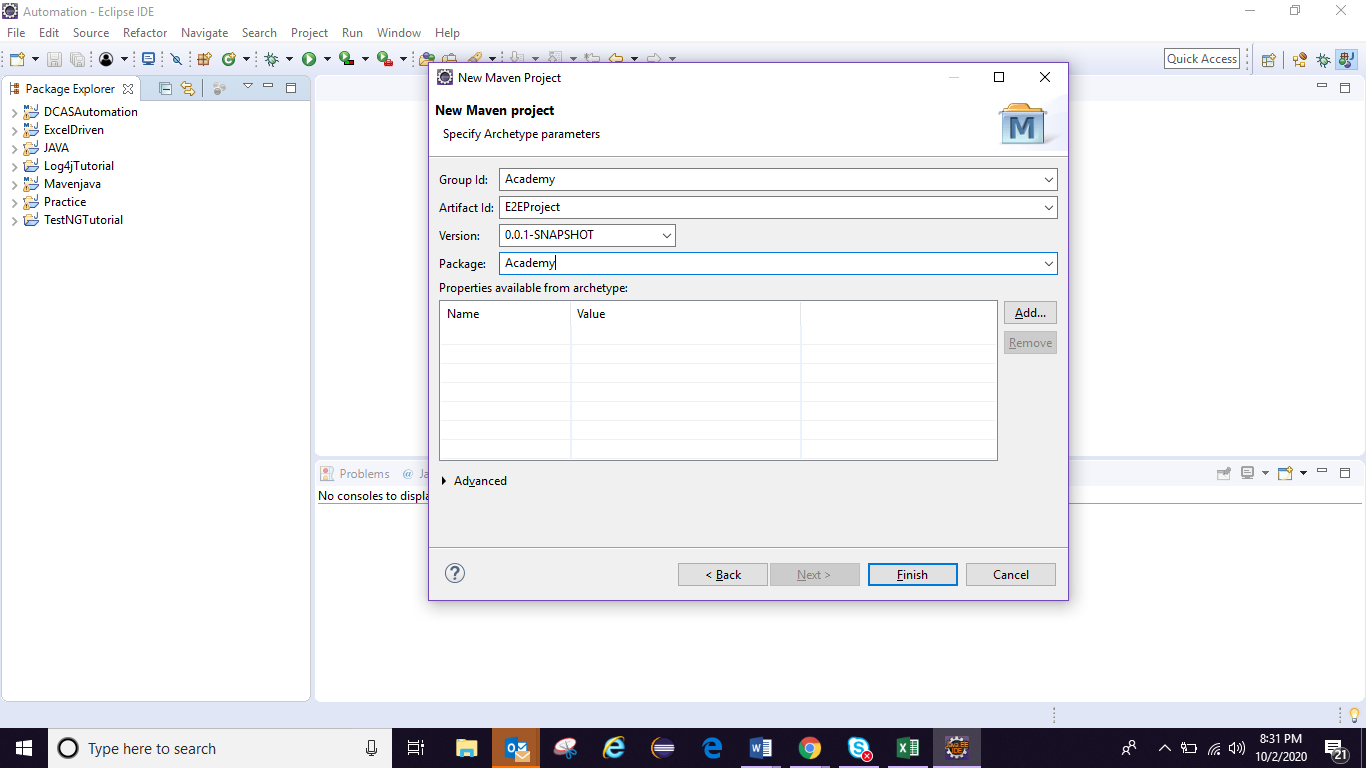
1. Here we will write all unitity classes like

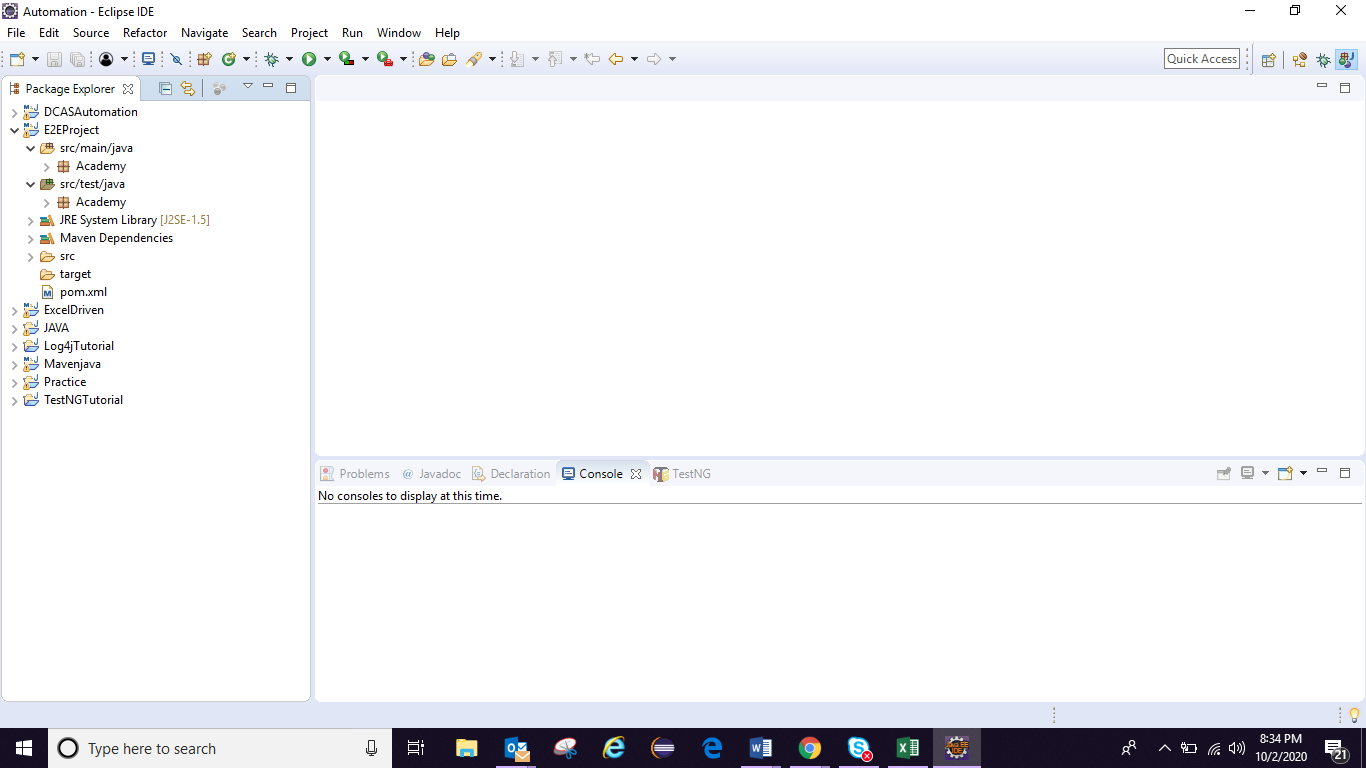
* Base class -> where have to write open browser and url, screenshot method
* ExtentReport class to fetch extent report
* Create data.properties file where needs to be mention about browser and url, credentials
* Write log4j.xml file here if including logs framework here

Src test Folder:

1. Have to write all test cases.
2. Create listeners class also here and call createTest () of ExtentReports class to create test instance which unique for every test method using that can use in every listener’s override methods
3. 







Main Folder:

Mention each page and their xpath and methods of each webElemet

Test Folder: all the test cases has to write

# Now integrate testing.xml into Maven means into pom.xml

Type in google “ testing maven integration ” :

<Build>

1. <plugins>
2. <plugin>
3. <groupId>org.apache.maven.plugins</groupId>
4. <artifactId>maven-surefire-plugin</artifactId>
5. <version>3.0.0-M5</version>
6. <configuration>
7. <suiteXmlFiles>
8. <suiteXmlFile>testng.xml</suiteXmlFile>
9. </suiteXmlFiles>
10. </configuration>
11. </plugin>
12. </plugins>

</Build>

## To add Log4j with Maven-> have add dependences in pom.xml

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-api</artifactId>

<version>2.13.3</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-core</artifactId>

<version>2.13.3</version>

</dependency>

## Now put log4j.xml in Resource folder of project:

## And put below plugins into pom.xml in side <Build></Build> tag

<resources>

<resource>

<directory>src/main/resources</directory>

</resource>

</resources>

## add common io dependency in pom.xml to take screenshots:

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.8.0</version>

</dependency>

### to add screenshots with listener have to crate

Listeners class and have add below properties in testing.xml:

<listeners>

<listener class-name=*"Academy.Listeners"*/>

</listeners>

## Extent Reports: have to get Extent Report jar:

1. For Extent Report will write an unility class and always to get the report we have to use extent.createTest() method for each test case but if we write the same extent.createTest() in side Listeners method’s of onTestStart() method then for every test method no need to write.
2. Because if we write in onTestStart() method it will run each and every time before staring test method .
3. Then we need to flush or release report after running all test so have to write extent.flush() in side onFinish() method of Listeners.

## To run tset method parallelly and to make framework as thread safe call have to create an object of ThreadLocal and put all the test method into ThreadLocal , so that override of the test instance problem can be resolved.

ExtentReport:

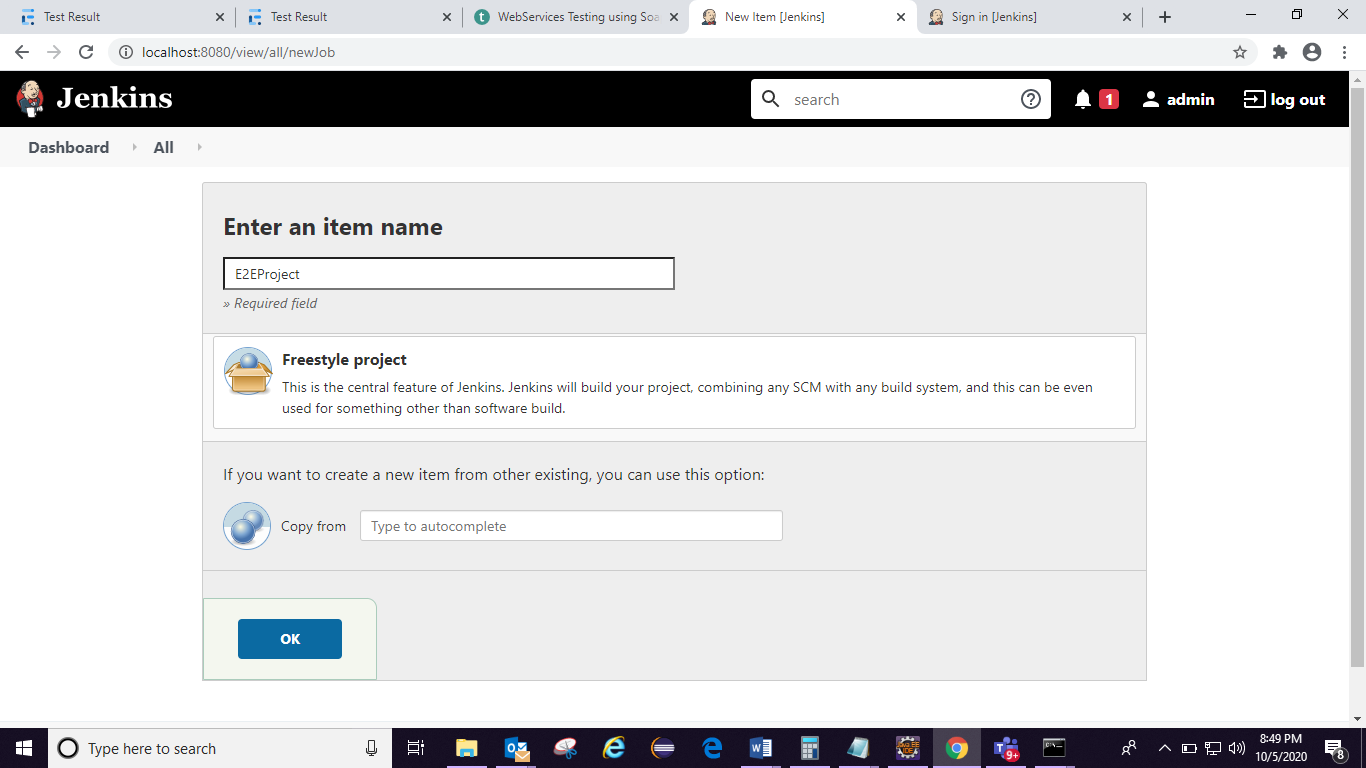
1. We can get all the success logs if test is passes
2. Get the failed log
3. We already are getting failed test case screenshot and placing in report file path now want to pull that screenshot in Extent Report

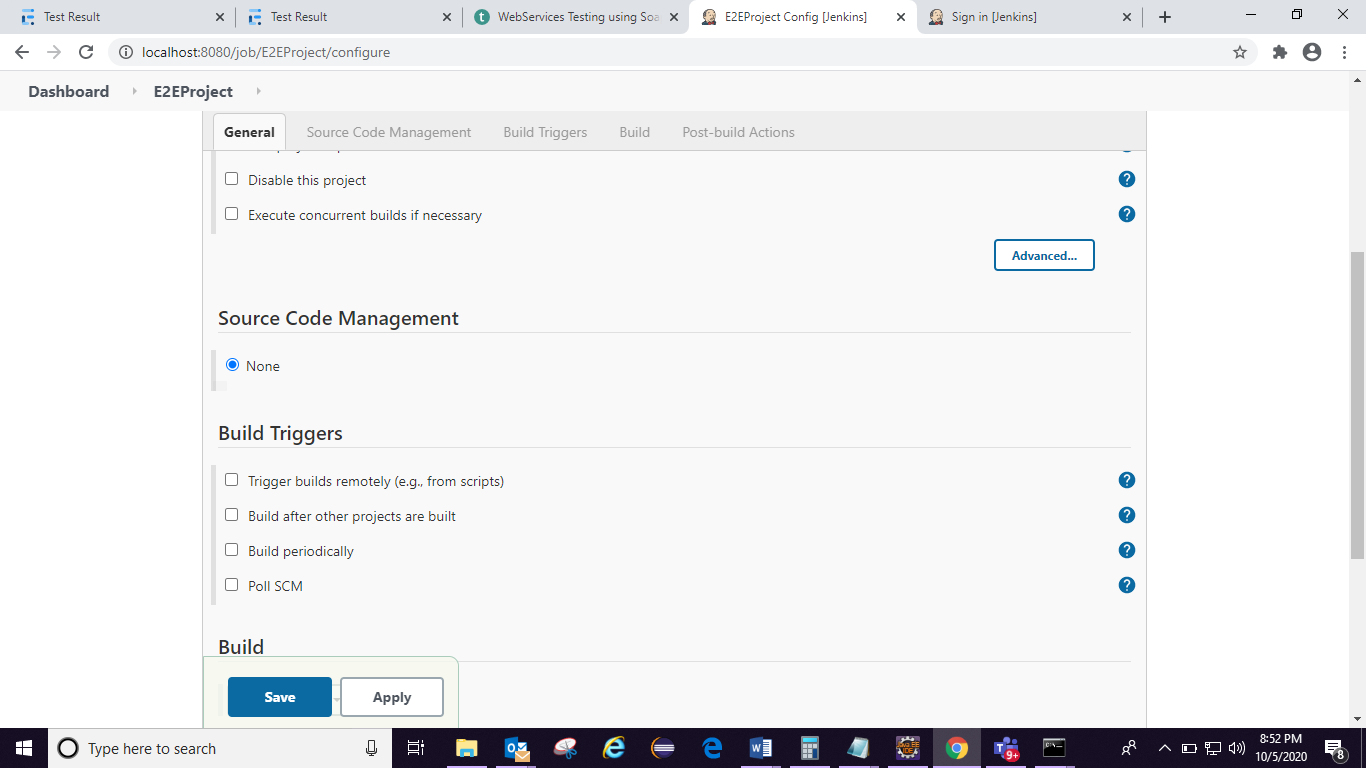
Screenshot:

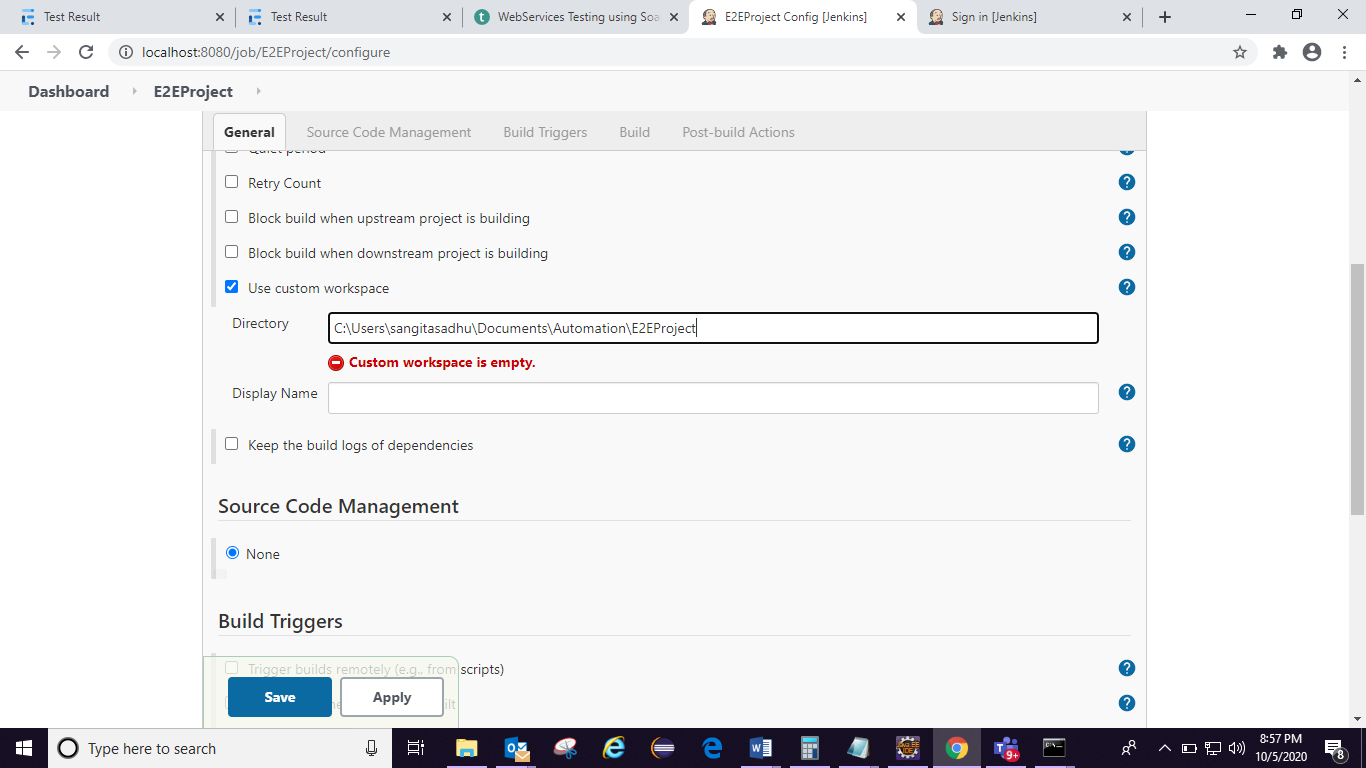
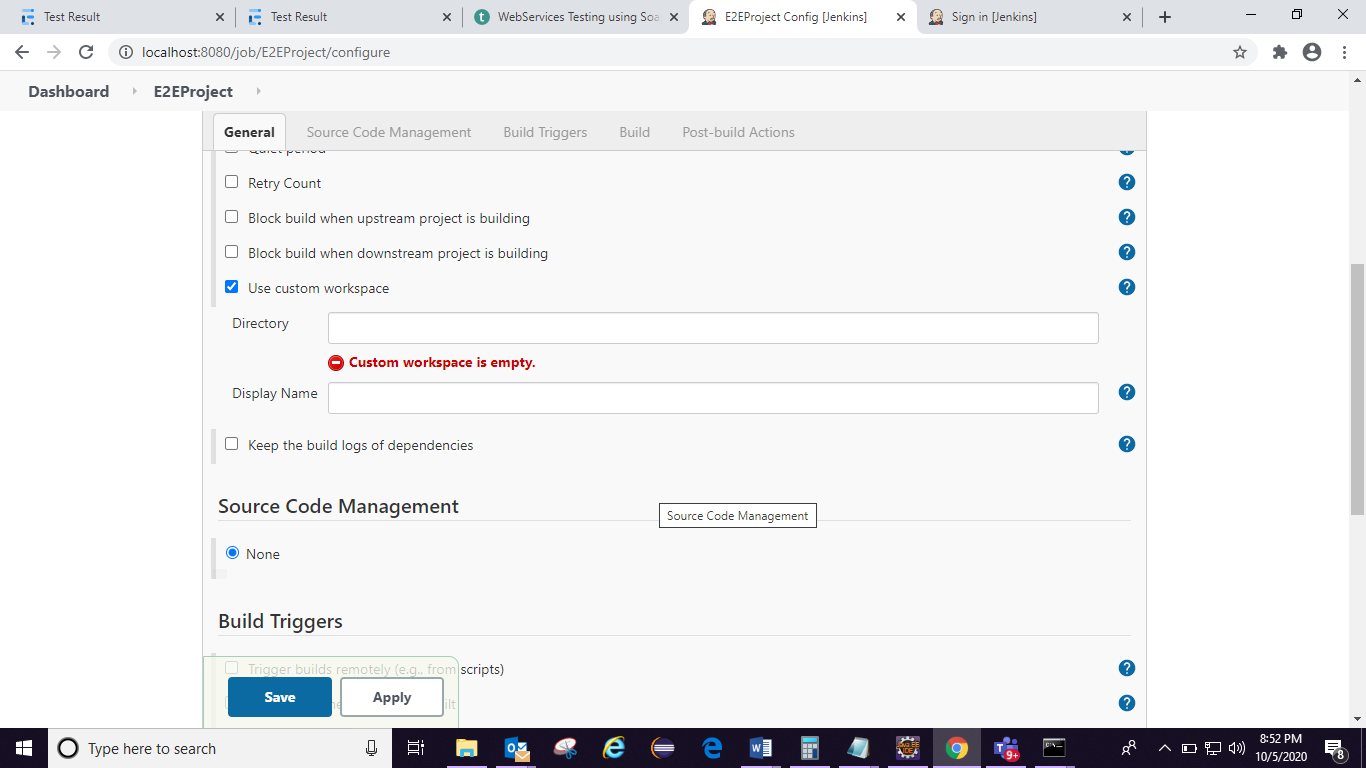
extentTest.get().addScreenCaptureFromPath(getScreenShot(testMethodName,driver),result.getMethod().getMethodName());

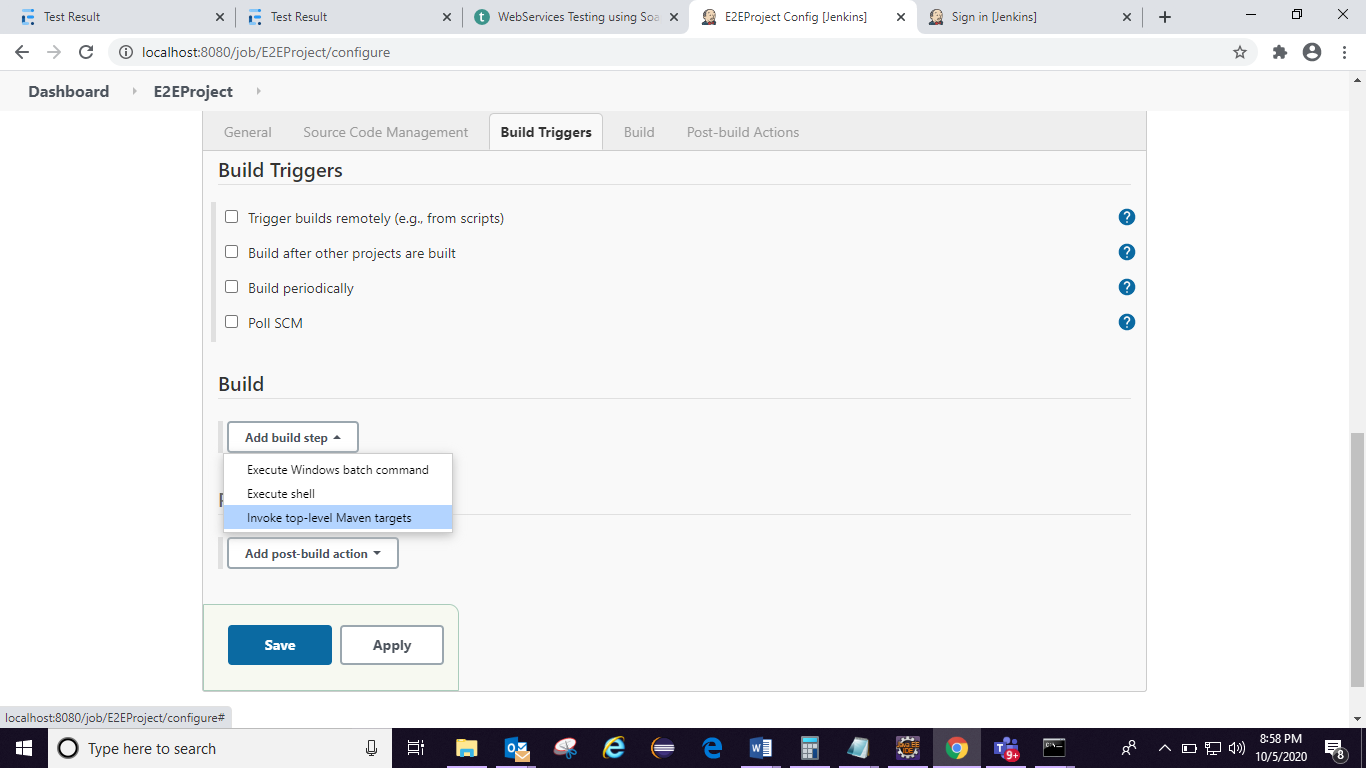
1. First arg: screenshots path
2. Test method or test case name

Jenkins:

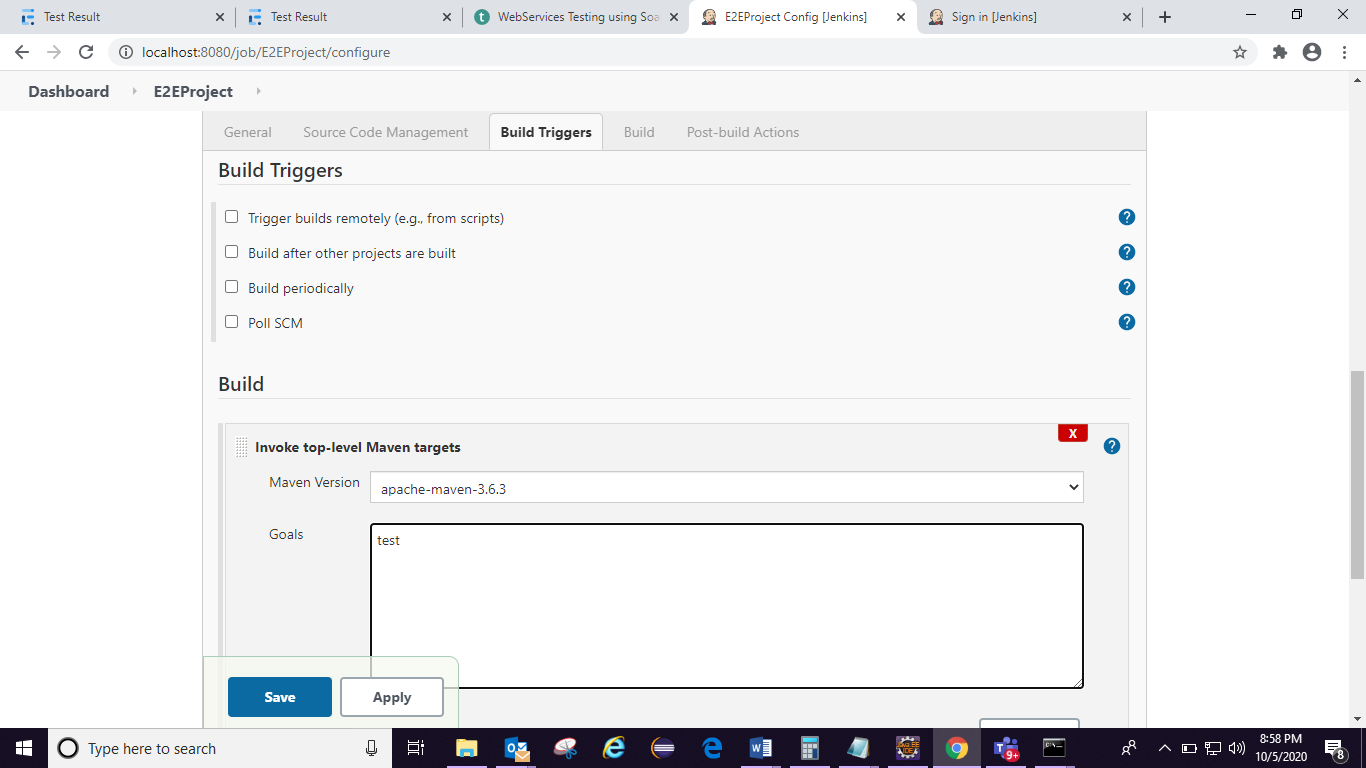




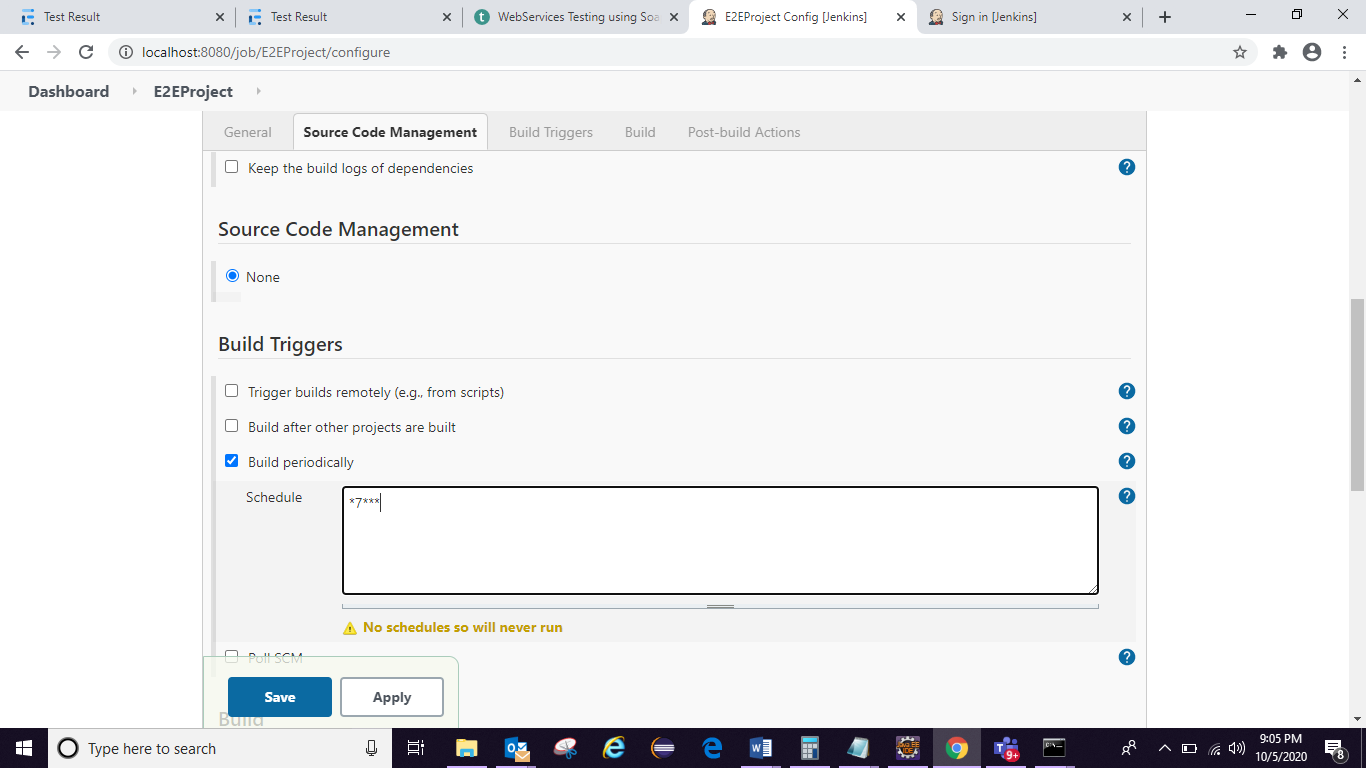




Goal section is similar to write maven command run time automatically will call mvn so write only test:



## to schedule job everyday:



1. if we want to send parameter through maven command instead of writing in the test. Should able to send what browser want to send from maven command

Mvn test -Dparametername=value

Ex: mvn test -Dbrowser=chrome

Prop.getProperty(“browser”) -> it will give the value of browser

But in maven it will give as

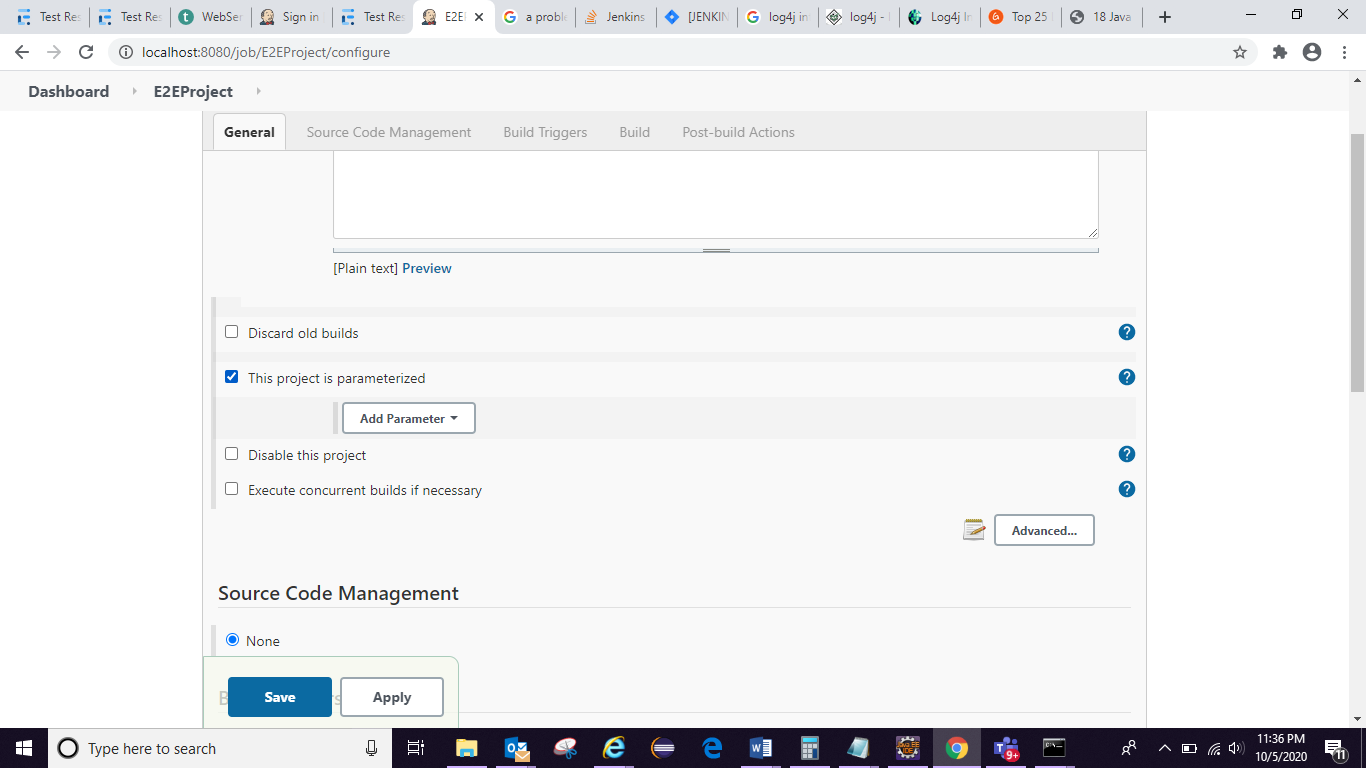
String browserName=System.getProperty(“browser”) -> it will check in maven command

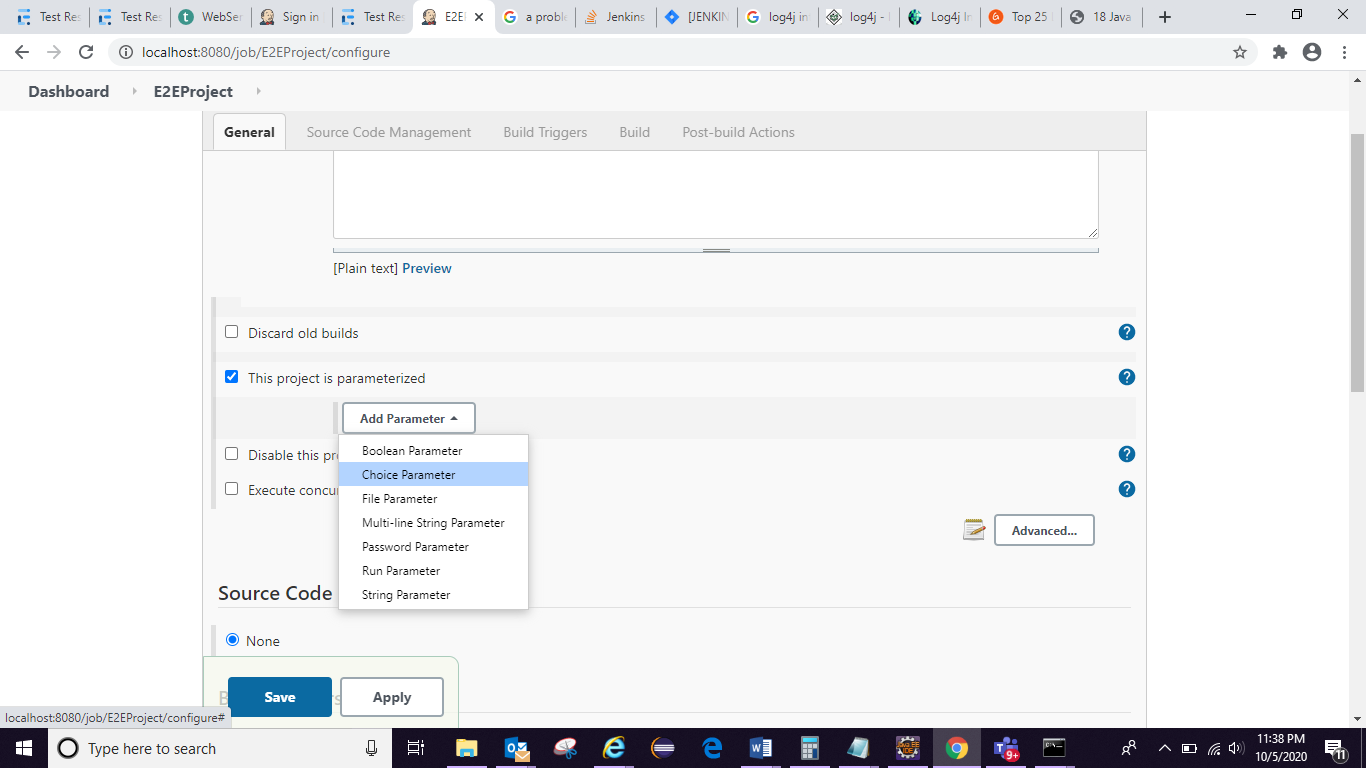
Then through cmd: go to project path-> mvn test -Dbrowser=chrome

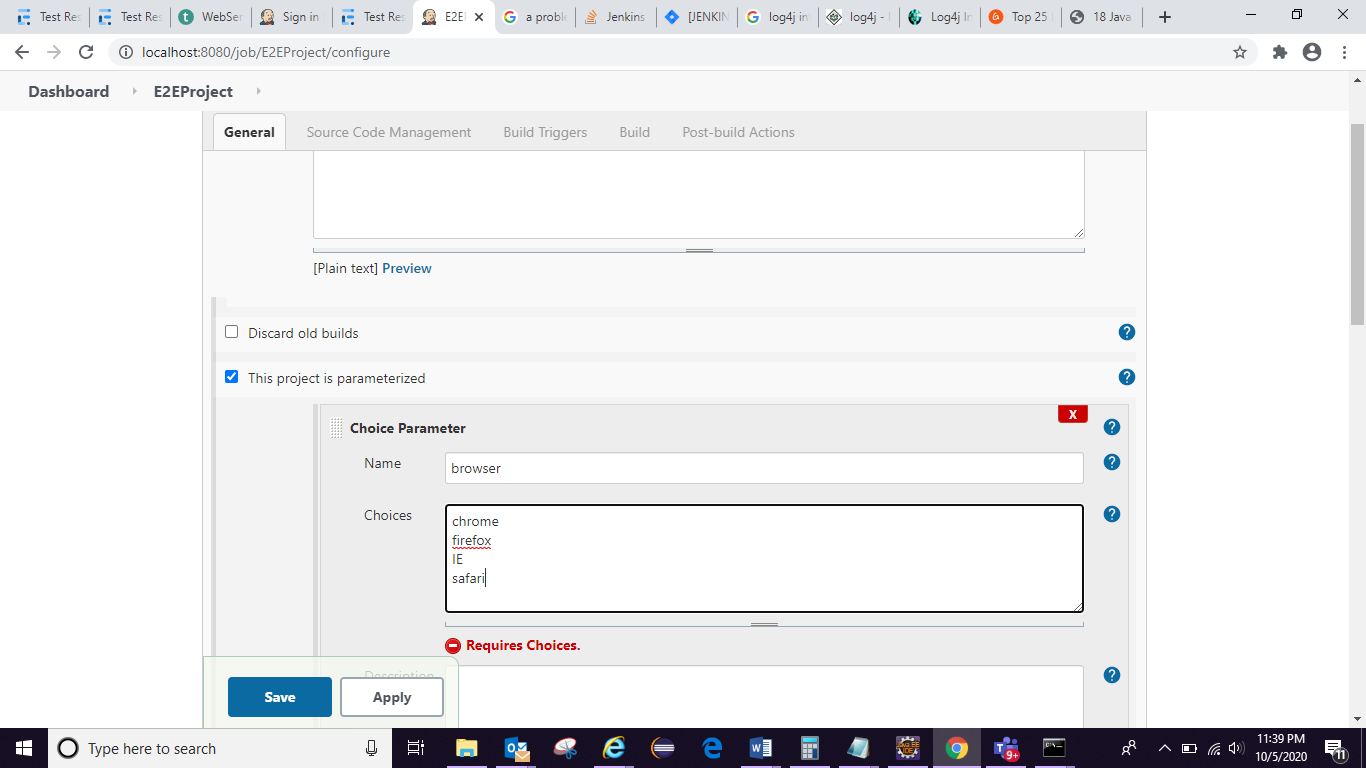
Note: if we want to send parameter through maven command instead of writing in the test.

Using this we are controlling in which browser you want to execute from command not from code so business people can easily execute tests

### To build with parameter: so that client can run test case in any browser from Jenkins just to change parameter or browser name from build of Jenkins:

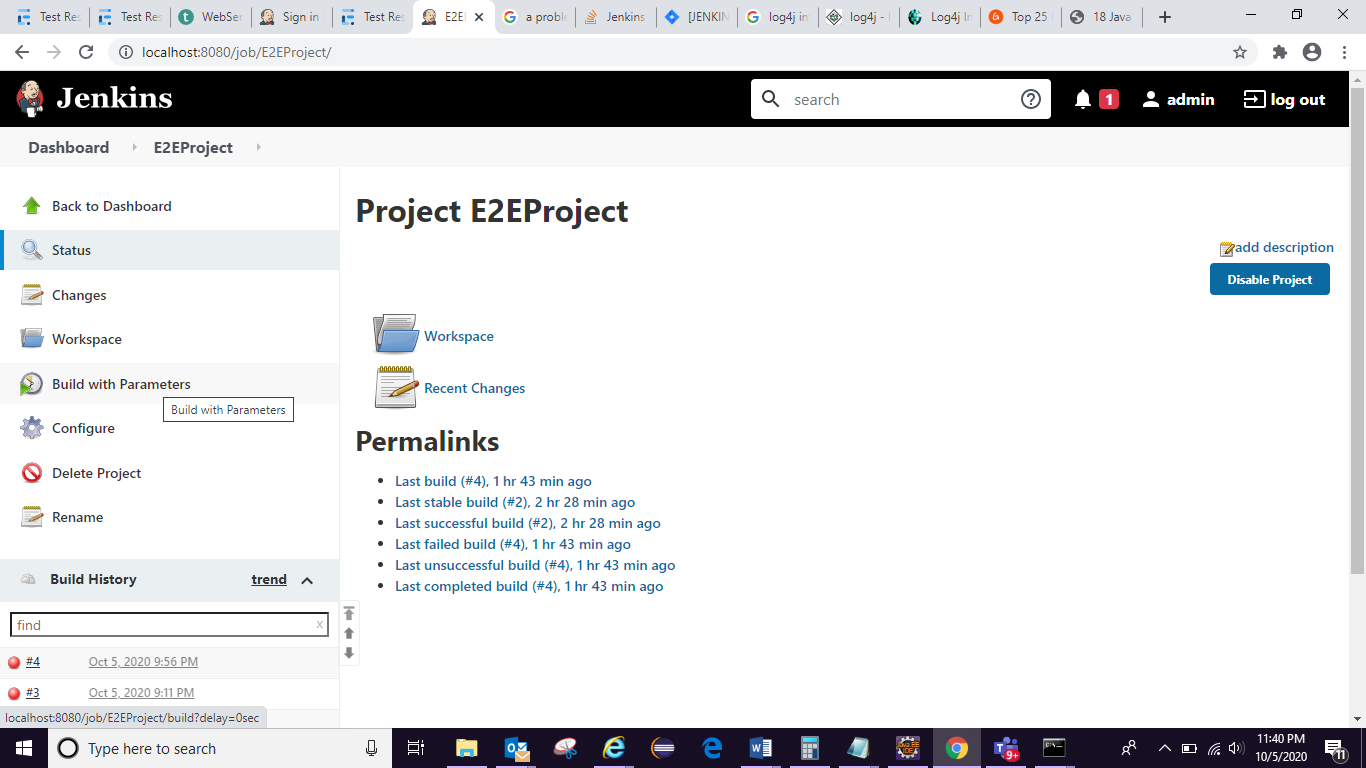


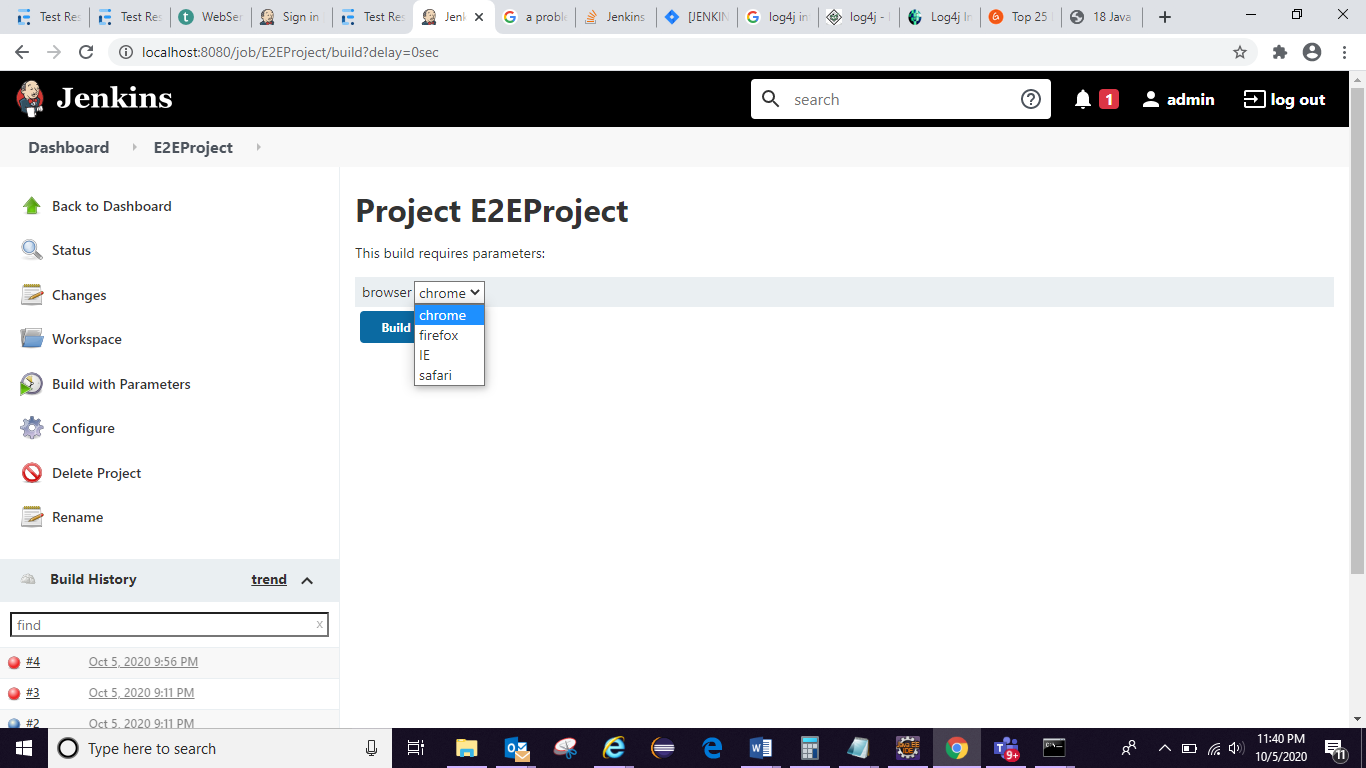




## choice parameter basically belongs to jenkins variable:

Any jenkins variable is acceptable when give “$browser”





Now want to integrate mvn test -Dbrowser=”$browser” to Jenkins: if we give this command in maven then Jenkins variable from choice parameter will come here(=”$browser”) whatever we choose as any browser from build with parameter:

