

## Katalon Studio:

Katalon Studio is an automation testing solution developed by Katalon LLC. The software is built on top of the open-source

automation frameworks Selenium, Appium with a specialized IDE interface for web, API, mobile and desktop application testing.

[https://www.youtube.com/watch?v=xBjNhauVDio&list=PLhW3qG5bs-L\\_D4ZePNNjvmIUluu6mBHbu&index=1](https://www.youtube.com/watch?v=xBjNhauVDio&list=PLhW3qG5bs-L_D4ZePNNjvmIUluu6mBHbu&index=1)

Automation Step by Step - Raghav Pal

### What is Katalon Studio ?

A complete web and mobile test automation tool  
(extends the capabilities of Selenium and Appium)

Available for windows and mac

Provides an intuitive GUI

Free

### What can you do with it ?

Integration with :

- JIRA, qTest
- Git
- Jenkins

## How can it help in your Test Automation

- Create **quick automation** Test Cases
- Has support for **Web, mobile and API**
- **Manual Testers** can start using it right away
- Learning curve is very short
- Useful features for **test creation, execution and reporting**
- In-built integration with **JIRA, Jenkins, Git** etc...



Effortless Test Automation. Web and Mobile.  
A complete automation framework supports manual testers to kickstart automation testing easily.



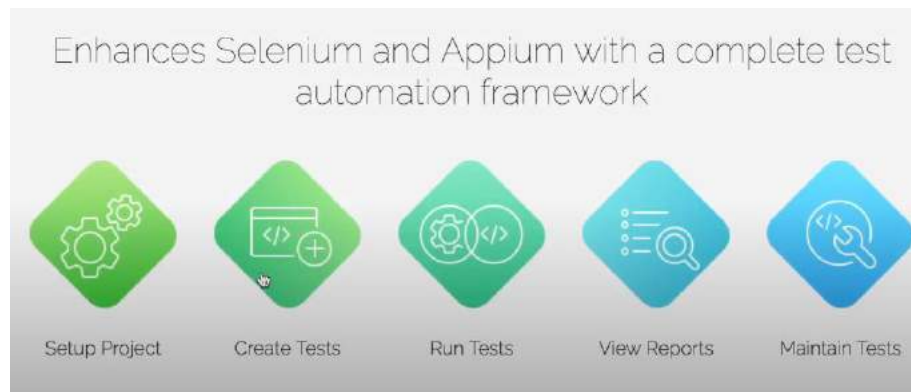
**Jenkins**



## What can you do with it ?

- Automate your web and mobile testing
- Can do web services (api) testing
- Create quick automation tests
- Can do record and playback
- Can run cross browser tests

<https://www.katalon.com/features/>



## Introduction to Runtime Engine

**Katalon Runtime Engine (KRE)** is the test execution add-on of Katalon Studio. KRE allows you to execute automation tests in CLI mode. It can be used in a variety of scenarios, such as scheduling your tests, integrating with CI/CD system, or bundling your tests to execute in virtual containers like Docker.

Install latest Katalon version 7.x.x.

### 1 GUI of Katalon Studio



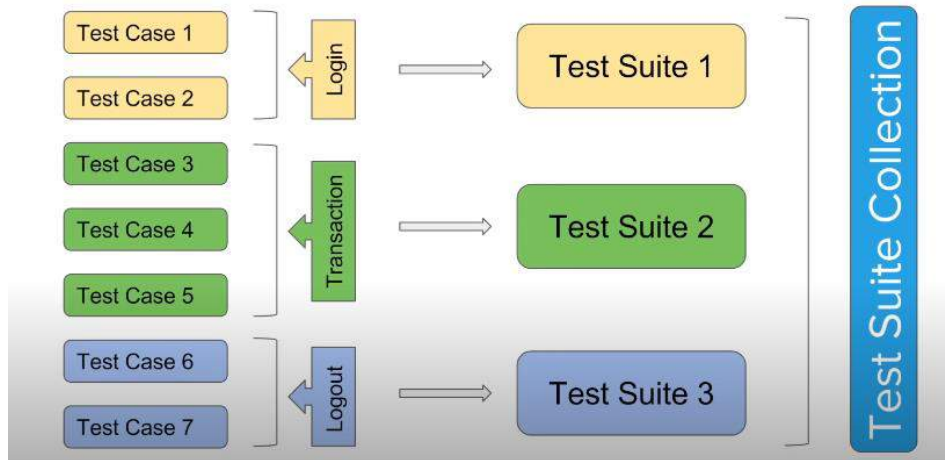
Creating testcase: 3 modes:

1. Record & play
2. Manual mode

3. Script mode: by using keywords(in-built, custom) or Java | groovy

Creating test suite from test cases/test case folders.

Test Suite collection:

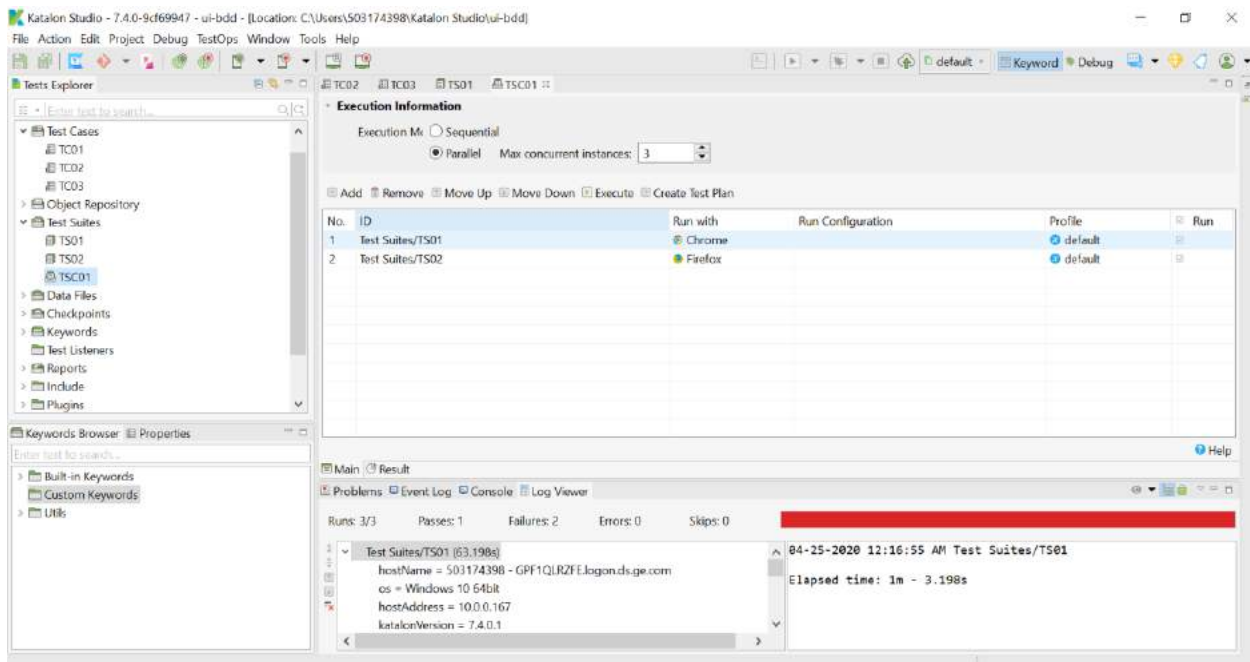


## What is it Required...

to group Test Suites logically

to run test suites in sequence or in parallel

to run test suites on multiple browsers / env



Today we will learn

- 1 Test Case Report
- 2 Test Suite Report
- 3 Test Suite Collection Report

## 2 Test Suite Report historical report

created after Test Suite execution  
 named with timestamp - YYYYMMDD\_HHmmss  
 provides summary + details  
 html report

Test Suite Report is preserved (historical report)

## 3 Test Suite Collection Report historical report

created after Test Suite Collection execution  
named with timestamp - YYYYMMDD\_HH:mm:ss  
provides summary + details  
html report

~~This Report is preserved (historical report)~~

We can configure email for sending the report: Projects-Settings-email.

Test suite has execution info->mail recipient. To put email ids

Executing TS and TSC from Command line.(console mode)

Generated command from "buildcmd" icon. Generated command can be used in ci-cd as well

## 3 Advantages

- Faster runs
- Auto-schedule
- Parallel executions
- Integration with other processes
- No need to open GUI
- Configure parameters : command | properties file



## Using properties file in console mode

Generate properties file (from command builder)

Use command:

katalon -propertiesFile="<absolute path to console.properties file>" -runMode=console

Integrating with Jenkins:

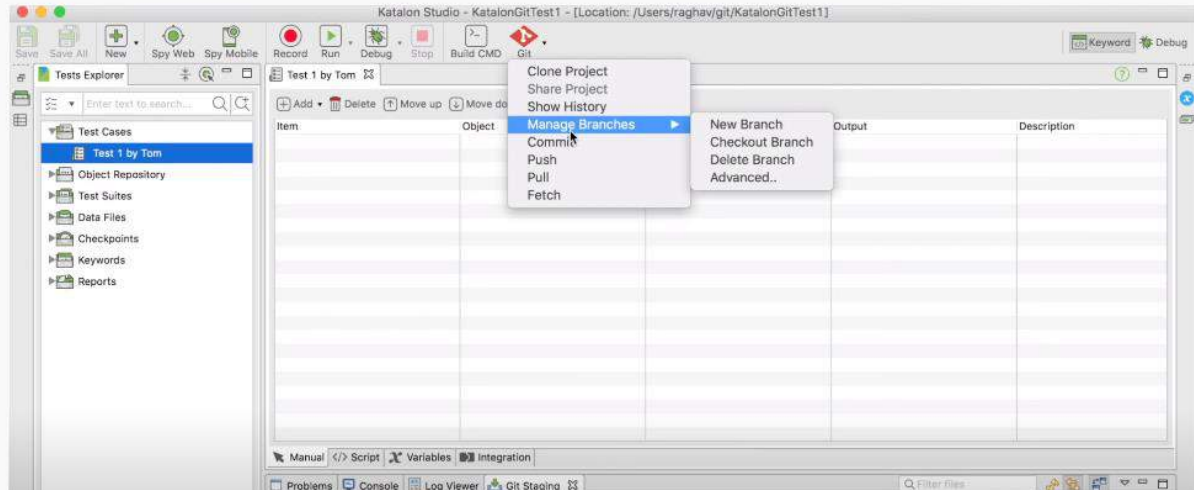
Create a project in katalon.

Go to window->preferences->Katalon->Git->check enable.

When Git icon is enable, to clone github/bitbucket repo give repo url and git credentials.

To commit any changes.

Git icon->commit->drag changes to staged area, give commit msg& press commit & push.



Katalon Recorder: chrome plugin:

Selenium IDE alternative to record and export Selenium scripts. With reports & screenshots. Fast & open-source.

## Katalon Automation Recorder

### How to create Automation Tests Faster

=====

#### Main Features:

1. Powerful IDE to record and run web ui tests
2. Can edit your recorded tests
2. Can export tests in Java, C#, Ruby, Python, Groovy
3. Plugin available on Chrome and Firefox (coming soon)
4. Uses Selenium 3 Core engine

#### Quick Guide

<https://forum.katalon.com/discussion/4056/quick-guide?new=1>

#### KT From Divya:

1. Katalon Project Folder structure
2. TC to call runner

```
//CucumberKW.runFeatureFile("Include/features/feature1/login.feature")
CucumberKW.runWithCucumberRunner("Include/scripts/runner/TestRunner.groovy")
```

Runner file will have path of all feature files.
3. Creating feature files.
4. Creating step def files(.groovy)(create **local variable accessible within class** in variables section present beside script mode option)  
Cucumber Runner file is created inside scripts folder.  
<https://docs.katalon.com/katalon-studio/docs/cucumber-kw-run-cucumber-runner.html>
5. Creating keywords for common actions(page class methods): use @Keyword for each method  
To use them:  
CustomKeywords.'ClickNumber.clickNumber'(firstOperand.toString())
6. Creating object repositories (OR)
7. Creating test data files. Eg internal data file for messages, users etc
8. Creating execution profiles(env files for different environments, variable declared are **global var.** here., we can choose any profile from dropdown from execution)  
To access anywhere, GlobalVariable.baseUrl



To create global var: GlobalVariable.baseUrl = "xyz.com"

9. Debugging groovy files: switch from keyword perspective to debug perspective
10. putting debug point, checking variables(show var values in all statements) ,breakpoint(list breakpoints applied),expressions( a+10, value calculated is shown)
11. Log viewer reports are not very standard report so we can use Cucumber HTML Report for failure analysis, JSON report for pushing report into some tool.
12. To manage the requirements/TestIds, Pushing report in tool, Xray Tool( Jira plugin) was used.
13. Test Listeners: AfterTestSuiteetc to close the browser, similar to listeners in TestNG.  
Listener class was also utilized to push the test results into Xray weekly.  
Xray provides some Rest API and we can use report Json to push the result into Xray.

Failure Handling options in any step:


Optional will be shown as warning in report.

Continue on failure will mark the step as failed in report but continue executin remaining step

Stop on failure will not remaining steps(skipped) and mark it failed in report.

//check some common methods for wait and verify

```
1 import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint[]
24
25 class Listener {
26
27     /**
28      * Executes after every test case ends.
29      * @param testCaseContext related information of the executed test case.
30      */
31     @AfterTestCase
32     def sampleAfterTestCase(TestCaseContext testCaseContext) {
33         WebUI.closeBrowser();
34
35         WebUI.openBrowser("goo", FailureHandling.OPTIONAL)
36
37     }
38
39
40
41 }
```



Press 'Ctrl+Space' to show Katalon Proposals

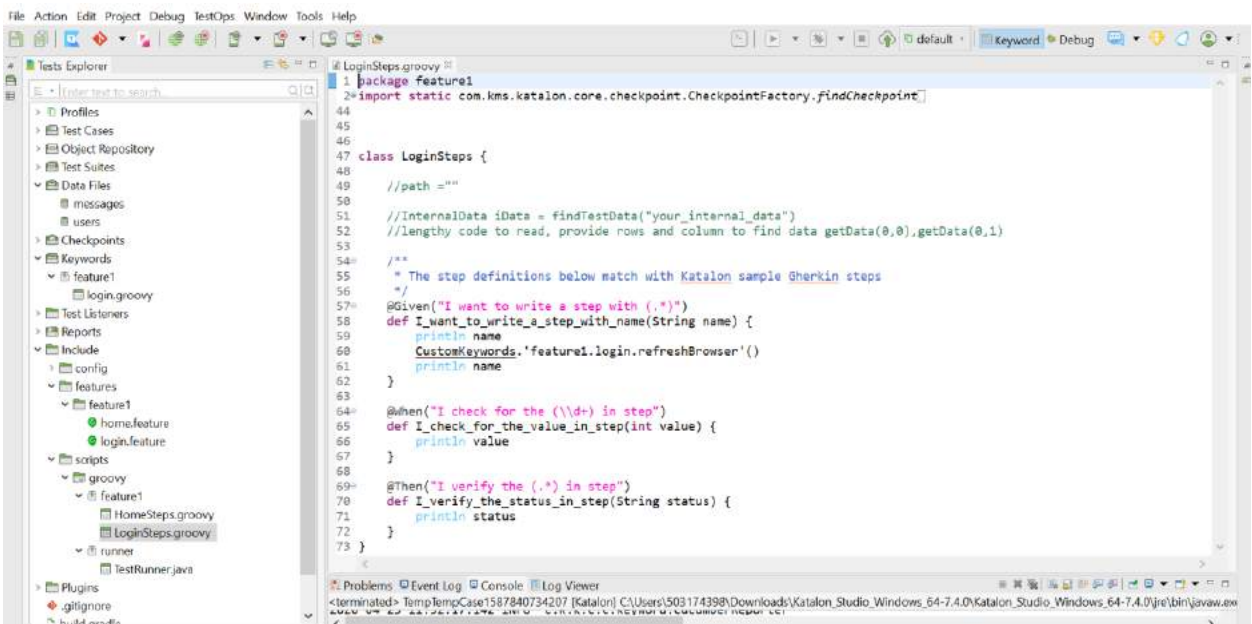
```

@tag
Feature: Title of your feature
  I want to use this template for my feature file

@tag1
Scenario Outline: Title of your scenario outline
  Given I want to write a step with <name>
  When I check for the <value> in step
  Then I verify the <status> in step

Examples:
  | name | value | status |
  | name1 | 5 | success |
  | name2 | 7 | Fail |

```



In case at any point do not want to use WebUI methods then you can get already launched driver instance and use normal selenium webdriver methods.

```

WebDriver driver = DriverFactory.getWebDriver()
driver.findElement(By.xpath(""))
driver.close()

```

More details on above:

[https://docs.katalon.com/katalon-studio/docs/using\\_selenium\\_webdriver\\_katalon\\_studio.html#multiple-webdrivers](https://docs.katalon.com/katalon-studio/docs/using_selenium_webdriver_katalon_studio.html#multiple-webdrivers)

**Mobile Automation(Android, 300/700 UI test cases):**

Separate project/repo at all.

We can use same feature files for some scenarios but step def n other locators has to be changed.

Appium desktop (.exe file) is used to connect with mobile, install apk on device & identify the locators in developer mode. An image of app is shown on Appium desktop.

We can locate the element with app in developer mode. When app goes into playstore we remove developer mode.

Katalon script methods start Appium server internally and launch the android app on connected device in execution mode & runs the script.

You want to start an application using absolute path and uninstall it after that.

```
import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
import static com.kms.katalon.core.testcase.TestCaseFactory.findTestCase
import static com.kms.katalon.core.testdata.TestDataFactory.findTestData
import static com.kms.katalon.core.testobject.ObjectRepository.findTestObject
import com.kms.katalon.core.checkpoint.Checkpoint as Checkpoint
import com.kms.katalon.core.checkpoint.CheckpointFactory as CheckpointFactory
import com.kms.katalon.core.mobile.keyword.MobileBuiltInKeywords as MobileBuiltInKeywords
import com.kms.katalon.core.mobile.keyword.MobileBuiltInKeywords as Mobile
import com.kms.katalon.core.model.FailureHandling as FailureHandling
import com.kms.katalon.core.testcase.TestCase as TestCase
import com.kms.katalon.core.testcase.TestCaseFactory as TestCaseFactory
import com.kms.katalon.core.testdata.TestData as TestData
import com.kms.katalon.core.testdata.TestDataFactory as TestDataFactory
import com.kms.katalon.core.testobject.ObjectRepository as ObjectRepository
import com.kms.katalon.core.testobject.TestObject as TestObject
import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WSBuiltInKeywords
import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WS
import com.kms.katalon.core.webui.keyword.WebUIBuiltInKeywords as WebUIBuiltInKeywords
import com.kms.katalon.core.webui.keyword.WebUIBuiltInKeywords as WebUI
import internal.GlobalVariable as GlobalVariable

'Start application on current selected android device'
Mobile.startApplication('C:\\Users\\admin\\androidfile.apk', true)

'Close application on current selected android device'
Mobile.closeApplication()
```

ation.html# int to start application using relative path and uninstall it after that.

## Scripts:

```
WebUI.openBrowser('')
```

```
WebUI.navigateToUrl('https://www.google.com/')
```

```
//give the path of file .rs file of locator inside OR folder
//findTestObject('Object Repository/feature1/login/searchBox')
//spaces are allowed here in the path
```

```
WebUI.setText(findTestObject('TC01_OR/Page_GE Single Sign On/input_GE Single Sign On_username'), '503174398')
```

```
WebUI.setEncryptedText(findTestObject('TC01_OR/Page_GE Single Sign On/input_GE Single Sign On_password'), 'sbubEADLgWFTRoFHF/aUjQ==')
```

```
WebUI.click(findTestObject('TC01_OR/Page_GE Single Sign On/input_GE Single Sign On_submitFrm'))
```

```
WebUI.setText(findTestObject('TC01_OR/Page_Google/input_Signin_q'), 'india')
```

```
WebUI.verifyTextPresent('Gmail', false)
```

Login:

```
WebUI.click(findTestObject('Page_CuraHomepage/btn_MakeAppointment'))
```

```
WebUI.setText(findTestObject('Page_Login/txt_UserName'), Username)
```

```
WebUI.setText(findTestObject('Page_Login/txt_Password'), Password)
```

```
WebUI.click(findTestObject('Page_Login/btn_Login'))
```

landingPage =

```
WebUI.verifyElementPresent(findTestObject('Page_CuraAppointment/div_Appointment'), GlobalVariable.G_Timeout)
```

```
WebUI.closeBrowser()
```

```
WebDriver driver = DriverFactory.getWebDriver()
```

```
List<WebElement>options=driver.findElements(By.xpath("//ul//li//span"))
```

```
for (WebElementoption: options){  
    printlnoption.getText()  
}
```

Loop:

```
for (int i = 0; i < max; i++) {  
    // interested only in p's  
    if (searchMe.charAt(i) != 'p') {  
        break;  
    }  
  
    // process p's  
    numPs++;  
}
```

Exception handling:

```

try {
    WebUI.openBrowser('')

    WebUI.navigateToUrl('katalon.com')

    if (WebUI.getText(findTestObject('Object Repository/txt_signin')).length() < 0){
        throw new com.kms.katalon.core.exception.StepErrorException('Value required!')
    }

}
catch (StepErrorException e) {
    this.println(e)
}
catch (Exception e) {
    this.println("General issue occurs.")
}
finally {
    this.println("Navigate to a page.")
}

```

Assert:

```

New Test Case
import com.kms.katalon.core.model.FailureHandling as FailureHandling
import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WSBuiltInKeywords
import com.kms.katalon.core.testcase.TestCaseFactory as TestCaseFactory
import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUiBuiltInKeywords
import com.kms.katalon.core.testdata.TestDataFactory as TestDataFactory
import com.kms.katalon.core.testobject.ObjectRepository as ObjectRepository
import com.kms.katalon.core.mobile.keyword.MobileBuiltInKeywords as MobileBuiltInKeywords

WebUiBuiltInKeywords.openBrowser('', FailureHandling.STOP_ON_FAILURE)
WebUiBuiltInKeywords.navigateToUrl(GlobalVariable.global_Gmail_Url, FailureHandling.STOP_ON_FAILURE)
assert varA == true

```

Method:

```

WebUiBuiltInKeywords.openBrowser('', FailureHandling.STOP_ON_FAILURE)
WebUiBuiltInKeywords.navigateToUrl(GlobalVariable.global_Gmail_Url, FailureHandling.STOP_ON_FAILURE)

Integer myMethod1(def param1, def param2) {
    return param1 + param2
}

myVar = myMethod1(varA, varB)

```

Method declaration

Method call

Using list:

```
List<WebElement>elements=driver.findElementsByClassName('android.widget.RadioButton')
```

```
for (WebElementradio : elements) {
```

'Get the text of each element in the list and store in to the "actual\_Text" variable.'

```
String actual_Text = radio.getText()
```

```

    'Here verifying the actual text with expected text of "Dinner" on every iteration'

    if(actual_Text.equals("Dinner"))
    {
        'Click on expected Element "Dinner" '
radio.click();

        'Break the loop'

        break;
    }
}

```

### Verify successful appointment:

```

WebUI.comment('Story: Book an appointment')

WebUI.comment('Given that the user has logged into their account')

WebUI.openBrowser(GlobalVariable.G_SiteURL)

WebUI.callTestCase(findTestCase('Common Test Cases/Login'), [('Username') : 'John Doe', ('Password') :
'ThisIsNotAPassword'],
FailureHandling.STOP_ON_FAILURE)

WebUI.comment('And Appointment page is displayed')

if (true) {
    WebUI.selectOptionByLabel(findTestObject('Page_CuraAppointment/lst_Facility'), 'Hongkong CURA
Healthcare Center', false)

    WebUI.check(findTestObject('Page_CuraAppointment/chk_Medicaid'))

        WebUI.check(findTestObject('Page_CuraAppointment/chk_Readmission'))

        WebUI.setText(findTestObject('Page_CuraAppointment/txt_VisitDate'), '27/12/2016')

        WebUI.setText(findTestObject('Page_CuraAppointment/txt_Comment'), 'Please make appointment as soon as
possible.')
    }

    WebUI.comment('When he fills in valid information in Appointment page')

    WebUI.click(findTestObject('Page_CuraAppointment/btn_BookAppointment'))

    WebUI.verifyTextPresent('Appointment Confirmation', false)

    WebUI.comment('Then he should be able to book a new appointment')

    if (true) {
        WebUI.verifyMatch('Hongkong CURA Healthcare Center',
WebUI.getText(findTestObject('Page_AppointmentConfirmation/lbl_Facility')),
        false)

        WebUI.verifyMatch('Yes',
WebUI.getText(findTestObject('Page_AppointmentConfirmation/lbl_HospitalReadmission')), false)
    }
}

```



```

WebUI.verifyMatch('Medicaid', WebUI.getText(findTestObject('Page_AppointmentConfirmation/lbl_Program')),
false)

WebUI.verifyMatch('27/12/2016',
WebUI.getText(findTestObject('Page_AppointmentConfirmation/lbl_VisitDate')), false)

WebUI.verifyMatch('Please make appointment as soon as possible.',
WebUI.getText(findTestObject('Page_AppointmentConfirmation/lbl_Comment')),
false)
}

WebUI.takeScreenshot()
WebUI.closeBrowser()

```

### BDD sample project:

Keywords/action methods/page class.

We can also use String/int/Boolean etc instead of def

```

@Keyword
def clickNumber(String number) {
    def digits = getDigits(number)
    for (def digit : digits) {
        digit = Integer.parseInt(String.valueOf(digit))
        switch (digit) {
            case 0:
                WebUI.click(findTestObject('Page_React
Calculator/button_digit_0'))
                break
            case 1:
                WebUI.click(findTestObject('Page_React
Calculator/button_digit_1'))
                break
        }
    }
}

def getDigits(String number) {
    return number.toCharArray()
}

```

## Test Listeners (Test Hooks)

```

class NewTestListener {
    /**
     * Executes before every test case starts.
     * @param testCaseContext related information of the executed test case.
     */
    @BeforeTestCase

```

```

def sampleBeforeTestCase(TestCaseContext testCaseContext) {
    println testCaseContext.getTestCaseId()
    println testCaseContext.getTestCaseVariables()
}

/**
 * Executes after every test case ends.
 * @param testCaseContext related information of the executed test case.
 */
@AfterTestCase
def sampleAfterTestCase(TestCaseContext testCaseContext) {
    println testCaseContext.getTestCaseId()
    println testCaseContext.getTestCaseStatus()
}

/**
 * Executes before every test suite starts.
 * @param testSuiteContext: related information of the executed test suite.
 */
@BeforeTestSuite
def sampleBeforeTestSuite(TestSuiteContext testSuiteContext) {
    println testSuiteContext.getTestSuiteId()
}

/**
 * Executes after every test suite ends.
 * @param testSuiteContext: related information of the executed test suite.
 */
@AfterTestSuite
def sampleAfterTestSuite(TestSuiteContext testSuiteContext) {
    println testSuiteContext.getTestSuiteId()
}
}

```

**In test Suite file:**

**2Setup annotated method is executed before all the suites is excuted**  
**@before suite method in lister is excitedbeore every suite.**

```
*/
@SetUp(skipped = false) // Please change skipped to be false to activate this method.
def setUp() {
    WebUI.openBrowser('')

    WebUI.navigateToUrl('https://www.google.com/')

    WebUI.setText(findTestObject('TC01_OR/Page_GE Single Sign On/input_GE
Single Sign On_username'), '503174398')

    WebUI.setEncryptedText(findTestObject('TC01_OR/Page_GE Single Sign
On/input_GE Single Sign On_password'), 'sbubEADLgWFTRoFHF/aUjQ==')

    WebUI.click(findTestObject('TC01_OR/Page_GE Single Sign On/input_GE
Single Sign On_submitFrm'))

    WebUI.setText(findTestObject('TC01_OR/Page_Google/input_Signin_q'),
'india')

    WebUI.verifyTextPresent('Gmail', false)
}

/**
 * Clean test suites environment.
 */
@TearDown(skipped = false) // Please change skipped to be false to activate this
method.
def tearDown() {
    WebUI.closeBrowser()
}

/**
 * Run before each test case starts.
 */
@SetupTestCase(skipped = true) // Please change skipped to be false to activate this
method.
def setupTestCase() {
    // Put your code here.
}

/**
 * Run after each test case ends.
 */
@TearDownTestCase(skipped = true) // Please change skipped to be false to activate
this method.
def tearDownTestCase() {
    // Put your code here.
}
```

Assert vs Verify in selenium:

In case of the “**Assert**” command, as soon as the validation fails the execution of that particular test method is stopped **and** the test method is marked as failed. Whereas, in case of “**Verify**”, the test method continues execution even after the failure of an **assertion** statement.

Verifications in Katalon:

verify (kind of assertion )can be handled using FailureHandling class.

```
WebUI.verifyEqual(actual object, expected object) //fails the step
```

```
WebUI.verifyEqual(actual, expected, FailureHandling.STOP_ON_FAILURE)
```

```
WebUI.verifyElementClickable(null) //Boolean
```

```
WebUI.verifyElementPresent(null, 0, FailureHandling.STOP_ON_FAILURE) //Boolean
```

#### **Waits(explicit wait):**

```
WebUI.waitForElementClickable(TestObject e, int timeoutInSec)
```

```
WebUI.waitForElementClickable(TestObject e, int timeoutInSec, FailureHandling.STOP_ON_FAILURE)
```

```
WebUI.waitForAlert(int Timeout in sec)
```

#### **Common methods:**

Dropdown

Alert

Switch to

Checkbox

Radiobutton

Exceptions:

NoSuchElementException, StaleElementReferenceException, or ElementNotVisibleException.

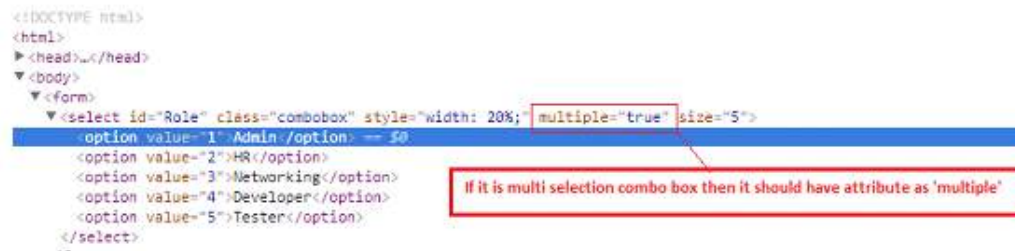
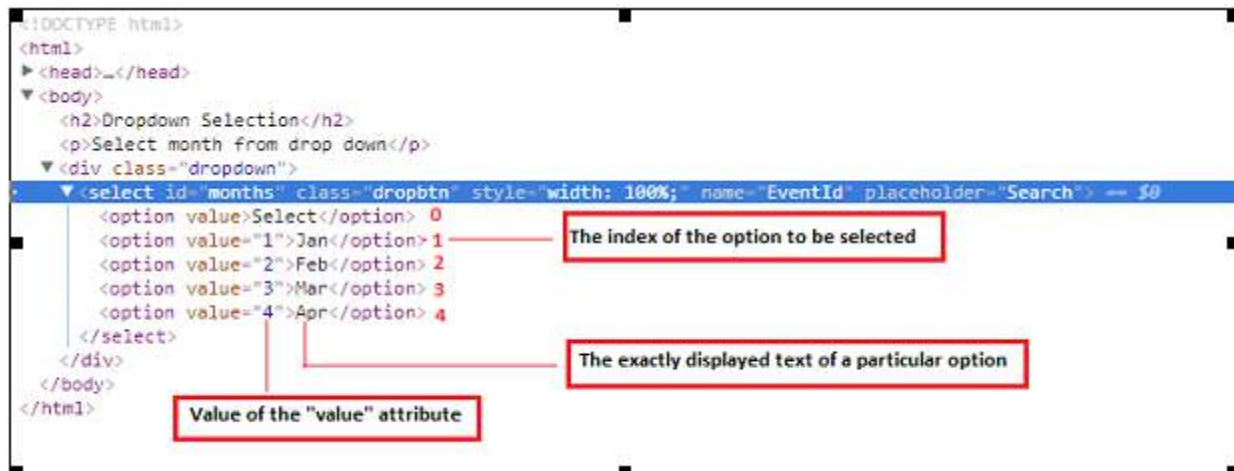
Warning in execution log:

```
11:15:41.957 INFO Unable to find the element located by 'By.xpath:  
//img[@name='isc_7main']'
```

**11:15:41.986 WARNING Web element with id:**

```
'Object Repository/wep_001_Login_Page/logo' located by 'By.xpath:  
//img[@name='isc_7main']' not found
```

Dropdown :



WebUI.selectOptionByIndex(TestObject e, Object range)

WebUI.selectOptionByLabel(TestObject e, Object label)

WebUI.selectOptionByValue(TestObject e, Object value)

SimilarlydeselectOptionBy()

- `to` - represent a web element
  - `range` - index range of the options to select. Index starts from 0.
- Example:
- 2 - index 2
  - "2,3" - index 2 and 3
  - "2-5" - index 2 to 5 (2, 3, 4, 5)

```

WebUI.maximizeWindow()

WebUI.selectAllOption(findTestObject('comboBox_Role'))

WebUI.deselectAllOption(findTestObject('comboBox_Role'))

NoOfSelectedOptions = WebUI.getNumberOfSelectedOption(findTestObject('comboBox_Role'))

WebUI.verifyEqual(NoOfSelectedOptions, 0)

```

### Checkbox/Radio Button:

```

WebUI.check(findTestObject('Pagereadmission'))

'verify Hospital readmission check box is checked'

WebUI.verifyElementChecked(findTestObject('P_readmission'), 30)

'un check Hospital readmission check box'

WebUI.uncheck(findTestObject('Page_CURA'))

'Verify uncheck Hospital readmission check box'

WebUI.verifyElementNotChecked(findTestObject('Page_CURA'), 30)

```

### Alert: accept/dismiss

```

WebUI.click(findTestObject('Alerts/button_ClickHere'))

'Accepting the Alert'

WebUI.acceptAlert()

```

### switchTo:

```

WebUI.navigateToUrl('https://www.google.com/')
WebUI.switch

```



The screenshot shows an IDE with a list of methods for the `WebUI.switch` command. The methods are:

- `switchToDefaultContent()` : void - WebUiBuiltInKeywords (Katalon)
- `switchToDefaultContent(FailureHandling flowControl)` : void - V
- `switchToFrame(TestObject to, int timeOut)` : boolean - WebUiBu
- `switchToFrame(TestObject to, int timeOut, FailureHandling flow`
- `switchToWindowIndex(Object index)` : void - WebUiBuiltInKeyw
- `switchToWindowIndex(Object index, FailureHandling flowContr`
- `switchToWindowTitle(String title)` : void - WebUiBuiltInKeywords
- `switchToWindowTitle(String title, FailureHandling flowControl)` :
- `switchToWindowUrl(String url)` : void - WebUiBuiltInKeywords (t
- `switchToWindowUrl(String url, FailureHandling flowControl)` : vc

### Andoid testing on APPIUM:



[https://www.youtube.com/watch?v=gsFXyUNiQFg&list=PLhW3qG5bs-L\\_D4ZePNNjvmIUluu6mBHbu&index=20](https://www.youtube.com/watch?v=gsFXyUNiQFg&list=PLhW3qG5bs-L_D4ZePNNjvmIUluu6mBHbu&index=20)

How to do mobile testing (android) on windows

---

Today we will learn

1. How to install Node.js
2. How to install Appium
3. How to setup Android device (mobile) for automation
4. Connection and Validation
5. Run TestCase on mobile

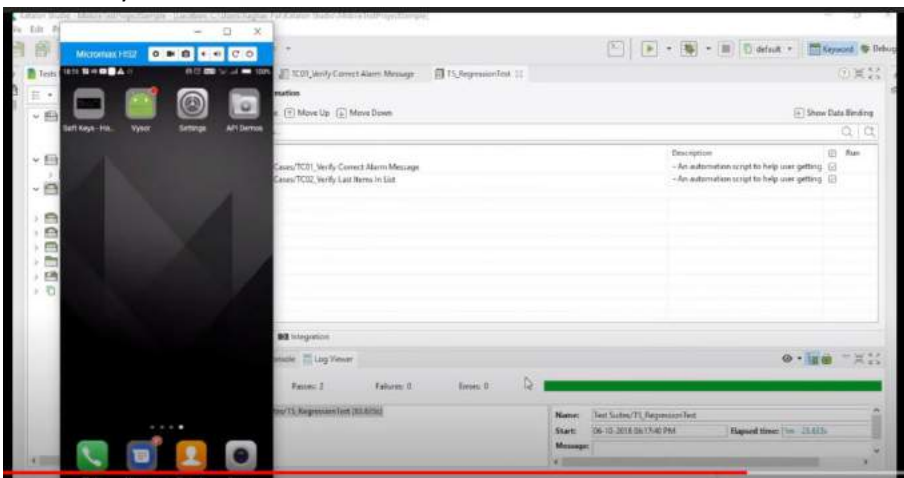
1) After npm install appium is successful, link Appium with katalon:  
Katalon studio preference->Katalon->Mobile->Appium directory: give the path of Appium installed inside node\_modules folder

2) set up android mobile device for automation:  
We can use emulator or real devices.

1. Settings > About Phone > tap 7 times on Build number
2. Settings > Developer Options > enable USB Debugging

Click 7 times on build number to enable developer mode.

Now connect mobile device with USB to system(enable usb debugging if you get prompt for it) and to show android screen on system we can use chrome plugin "Vysor".  
Click on Vysor ->click on View to view connected device via USB Cable.



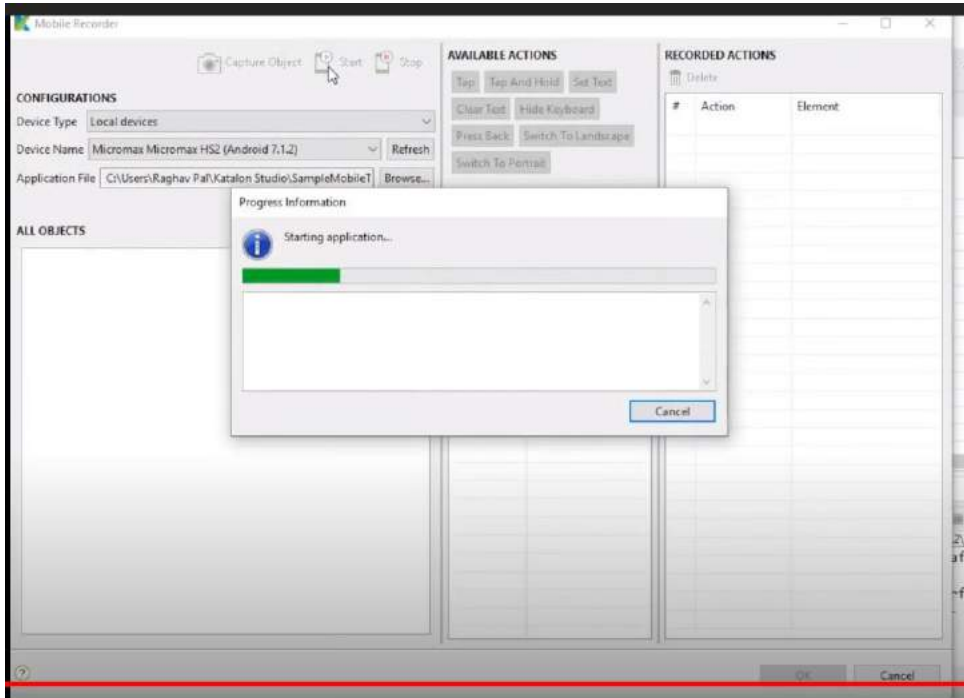
Run script on sample mobile project:

While running sample mobileproject select androidapp(instead of chrome or browser) it will then show the device name if connected properly. Select device n click ok.

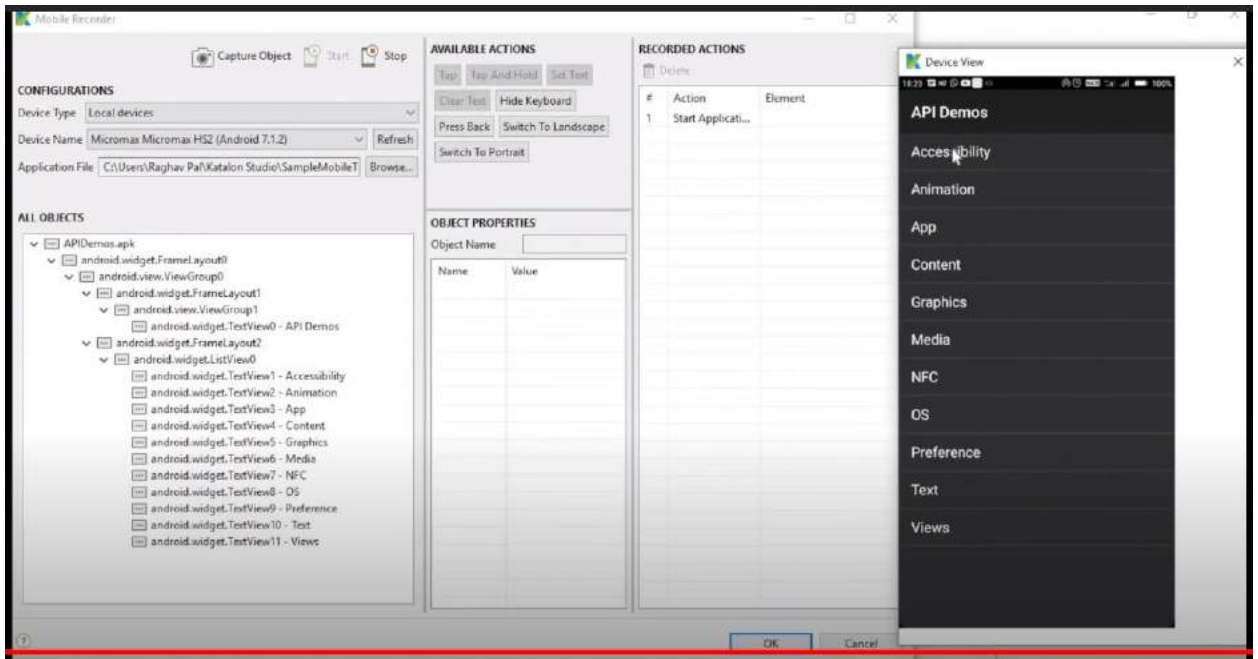
It will run the sample apk file and you can see on vysor screen.

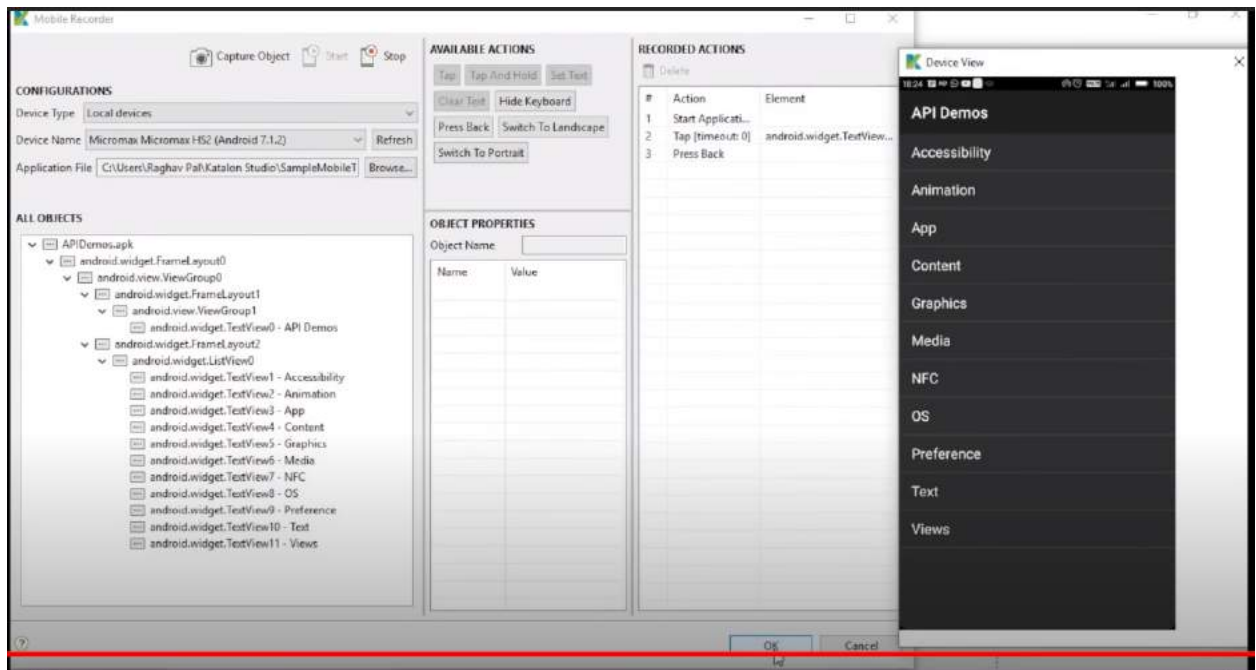
Create new mobile test cases by recording:

Record mobile(Android SDK has to be installed already)

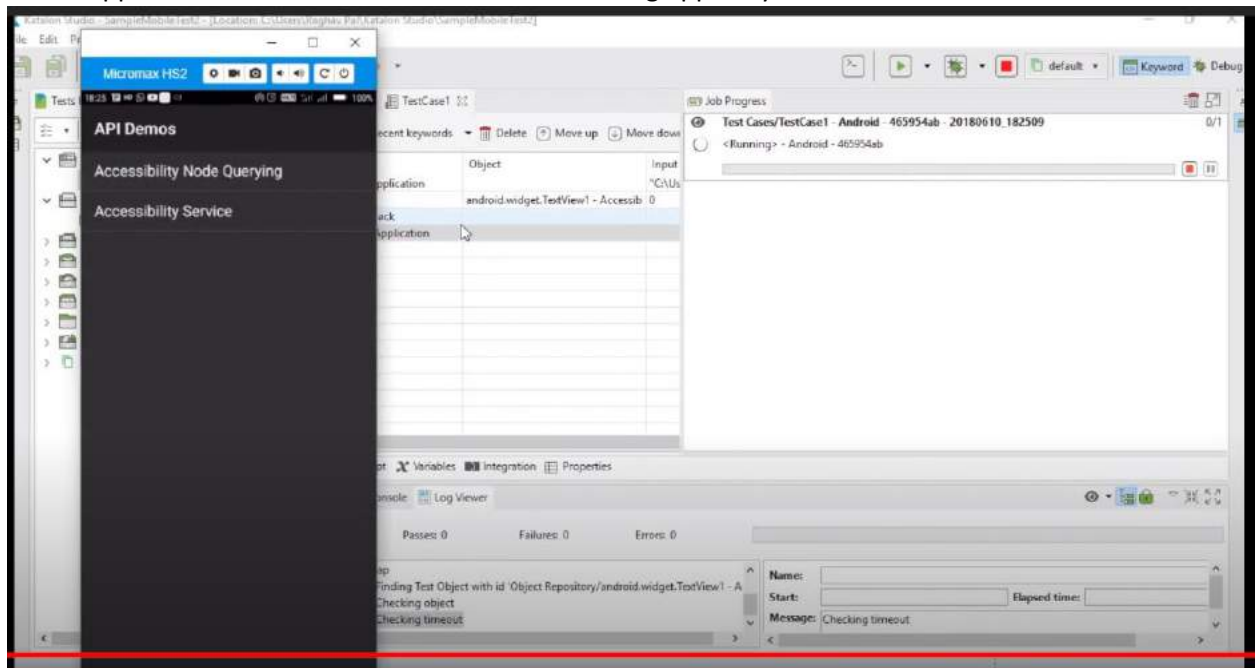


You will get device view inside katalon:





When Appium server is started we can see the running app in vsor:



Android APP automation:

```
Mobile.comment('Story: Verify correct alarm message')
```

```
Mobile.comment('Given that user has started an application')
```

```
'Get full directory's path of android application'
defappPath = PathUtil.relativeToAbsolutePath(GlobalVariable.G_AndroidApp,
RunConfiguration.getProjectDir())
```

```
Mobile.startApplication(appPath, false)

Mobile.comment('And he navigates the application to Graphics form')

Mobile.tap(findTestObject('Application/android.widget.TextView - Graphics'), GlobalVariable.G_Timeout)

Mobile.comment('When he scroll to Xfermodes text')

Mobile.scrollToText('Xfermodes')

Mobile.comment('Then the current screen should show Xfermodes text after scrolling')

'Get item\'s label'
def itemText = Mobile.getText(findTestObject('Application/Graphics/android.widget.TextView - Xfermodes'),
GlobalVariable.G_Timeout)

Mobile.verifyEqual(itemText, 'Xfermodes')

Mobile.closeApplication()
```