====================== APPIUM / Testing world =========================

1. download Genymotion for personal use with virtual box. and signup.

2. create virtual device: click add > select a device > finish. You will see a virtual device.

3. start the virtual device: select the device > click start > ok.

4. download the app.

============================== APPIUM================================

**1.** Select **File** -> **New** -> **Project…** option from Eclipse IDE .

**2.** In the New Project popup box, select Java Project and then click on Next button.

**3.** In the New Java Project popup box, write the project name as AppiumSetup. Leave all the other fields as default and then click on Finish button .

**4.** If the panel at the left side is collapsed, then you can restore it as shown below.

**5.** Eclipse IDE should now look like something as shown below. It would display your newly created project in the left side pane.

-2 different methods using which you can find appPackage and appActivity name of your app under test.

-In very basic terms, appPackage is the technical name of the app which is provided by its developers. It’s actually a top level package under which all the code for the app resides.

-in very basic terms appActivity refers to the different functionalities that are provided by the app.

-Method command prompt : Using ‘mCurrentFocus’ or ‘mFocusedApp’ in Command Prompt.

it will provide the appPackage and appActivity name of the app which is currently in focus.

Step 1: Unlock your mobile device and connect it to your computer using USB cable.

**Step 2:** Open Command Prompt and run ‘adb devices’ command. We are running this command to just make sure that your mobile is properly connected.

**Step 3:** Once you run ‘adb devices’ command, you should see that it displays the list of attached devices as shown in the below image (the actual device name that you see would be different based on what mobile phone you use) –

**Step 4:** Run ‘adb shell’ command. After running this command.

**Step 5:** Now in your mobile phone, open the app for which you want to find the appPackage and appActivity.

Step 6: Now run this command: dumpsys window windows | grep -E ‘mCurrentFocus’.

Step 7: The above command would display the details of the app which is currently in focus. From that, you can figure out the appPackage and appActivity name as per the below image –

appPackage starts with com. and ends before backshash (/). So from the above image, appPackage name is – com.android.vending

appActivity starts after the backslash (/) and goes till the end. From the above image, appActivity name is – com.google.android.finsky.activities.MainActivity

**Step 8:** There is one more similar command that provides the appPackage and appActivity name. This command adds some additional details before and after the package name & activity name, but you can still try it out just to verify that the results from the above command are same. This command is – dumpsys window windows | grep -E ‘mFocusedApp’ and the output of this command is shown below –

## Method 2: Using APK Info app

APK Info is an app which will provide the appPackage and appActivity name of any app which is installed on your mobile device.

Step 1: Download “APK Info” app from Google Play Store on your android mobile.

Step 2: Once you have successfully installed APK Info app, open it and check that it lists down all the apps that you have on your phone.

Step 3: Long press on the “Google Play Store” application icon inside the APK Info app till it displays the list of options as shown below –

Step 4: Click the option “Detailed Information” option. It would show the detailed log for the app.

Here, check the APK path section. This sections displays the “appPackage” name.

Note: Skip any number at the postfix of the name (eg: here its “-2”). So, the appPackage name in this case is – com.android.vending

Step 5: Then to find the appActivity name of the app, scroll down to the sub-section “Activities”. This sub-section displays all the activities that are available for the app. From this list, you have to look for the activity which has “MainActivity” or “Main” or “Login” in the activity name.

Here “com.google.android.finsky.activities.MainActivity” is the appActivity name for the Play Store app.

== First Appium test script==

1. Go to Start -> All Programs and look for Appium.

2. Click on Appium to open Appium Desktop (it might take 20-30 seconds for Appium Desktop to open)

**3.** Once Appium Desktop opens, navigate to the **Advanced** tab as shown below.

4. Enter the following values in the fields as shown in the below image –

* Server Address – 0.0.0.0
* Server Port – 4723
* Allow Session Override – tick the checkbox.

Note: Why did you tick “Allow Session Override” checkbox? When you run an Appium test script, it creates a new session on the Appium server. If this session doesn’t close properly, then your script would fail when you re-run it. This is because the previous session still exists and thus, Appium server is not able to create a new session.

And when you tick the checkbox, you provide Appium the capability to override the existing session when you re-run scripts. This would ensure that your scripts don’t fail due to session related issues.

**5.** Leave all other fields as it is, and then click on Start Server v1.7.2 button.

**6.** Appium server would start and you would see the screen as shown below.

## Get your Mobile Phone’s Device ID

Appium identifies your mobile phone using its Device Id (also called as its UDID). For example, consider a scenario where you have multiple mobile phones connected to your machine using USB cables. Now when you run your Appium test script, Appium would detect that you have multiple devices connected. So it would need the Device ID, so that it can connect to the correct device and run the scripts on that device. Let us see how you can find the Device ID (or UDID) of your mobile phone –

**1.** Connect your mobile device to your computer using USB cable. (If your mobile device shows some popup after connecting, then accept that popup)

**2.** Now open command prompt and run this command: adb devices

**3.** Once you run this command, you would see the details of the device as shown in the below image (if you see some other response such as “daemon not running. daemon started successfully”, then run the command again).

**4.** From the above screenshot the Device ID is – ENUL6303030010. Note down the device ID that you get for your phone, because you would need to provide that in your Appium test script.

**5.** Keep your mobile device connected to the USB cable. It needs to remain this way when you run your script.

## Get your Mobile Phone’s Android version

You would also need to provide your mobile phone’s Android version number in the Appium script. To find this, open your mobile phone and go toSetting > About Phone. In About Phone screen, you can see the Android version as shown below –

## Find out appPackage and appActivity names of the mobile app.

-appPackage name as – com.android.vending

and appActivity name as – com.google.android.finsky.activities.MainActivity

= **Write Appium Test Script to launch Play Store app**

**1.** Open the Appium project that we had created in this article.

**2.** In the Appium project in Eclipse, right click on src folder. Then select New > Package.

3. In the popup window, enter package name as tests and then click on Finish button.

4. You can see that a new package called tests is created under src folder. Right click on this package and select New > Class option.

**5.** In the popup window, enter the class name as AppiumTest and then click on Finish button.

package tests;

public class AppiumTest {

public static void main(String[] args) {

//Set the Desired Capabilities

DesiredCapabilities caps = new DesiredCapabilities();

caps.setCapability("deviceName", "My Phone");

caps.setCapability("udid", "ENUL6303030010"); //Give Device ID of your mobile phone

caps.setCapability("platformName", "Android");

caps.setCapability("platformVersion", "6.0");

caps.setCapability("appPackage", "com.android.vending");

caps.setCapability("appActivity", "com.google.android.finsky.activities.MainActivity");

caps.setCapability("noReset", "true");

//Instantiate Appium Driver

try {

AppiumDriver<MobileElement> driver = new AndroidDriver<MobileElement>(new URL("http://0.0.0.0:4723/wd/hub"), caps);

} catch (MalformedURLException e) {

System.out.println(e.getMessage());

}

}

}

You have written your Appium test script. Its now time to run it. Please make sure that your mobile device is connected to your computer with USB and its unlocked.

========================== **Appium Desktop features** ===========================

Two most important features of Appium Desktop Client are –

* Appium Desktop UI and its ability to start and stop Appium server
* Appium Desktop Inspector to inspect objects in your mobile app.

- **Tabs in Appium Desktop UI-**

### 1. Simple Tab : **Appium server requires minimum 2 parameters to run. And Simple tab provided these 2 parameters – Host and Port.** The default values for these parameters is “0.0.0.0” (for Host) and “4723” (for Port). So if you don’t provide any specific values for these parameters, Appium would take the default ones to start the server.

-You can now click on Stop Server (3rd button with pause icon) and then Close Log button (one with X icon), to navigate back to the Simple tab.

### 2. Advanced Tab :

General section : This section contains the options which are required for both Android and iOS

* iOS section : This section contains iOS only options
* Android section : Here, you have options needed for Android automation.

Please note that the options in iOS and Android sections are not always mandatory.

#### Start Appium Server from Advanced tab-

Go ahead and enter values “127.0.0.1” and “4723” in Server Address and Server Port fields. Also, tick the Allow Session Override checkbox as shown in the below image. Now click on Start Appium button .

-‘Start Inspector Session’. This is the button from where you can start Appium Desktop Inspector.

-Save As Preset button. With this functionality, you can save the inputs that you are providing to launch Appium server. So the next time you want to start the Appium server, you can just select the Preset and run it. This way you can avoid providing all the options every time you want to start the server.

-**1.** Navigate back to the Advanced tab in Appium Desktop (if you are on the logs screen, you can stop the server and then close the logs to navigate back to the Advanced tab)

**2.** Provide Server Address as “127.0.0.1”, Server Port as “4723” and tick Allow Session Override checkbox, as shown below.

**3.** Now click on Save As Preset… button. A popup would be displayed as shown below.

================= how connect mobile to windows ===============================

1. Download & Open eclipse IDE 2. Create Java Project 3. Add selenium & appium libraries

4. Connect mobile device 5. Start appium server 6. Write code to start a mobile application

7. Use appium desktop client to find element locators

Step 1: Download Eclipse or any other IDE for Java [https://www.youtube.com/playlist?list...](https://www.youtube.com/playlist?list=PLhW3qG5bs-L_qj1L5hnHvJYeFpQ_g4UuU)

Step 2 : Create a Java project

Step 3 : Add libraries Selenium java Appium java client [https://mvnrepository.com/](https://www.youtube.com/redirect?event=video_description&v=N7vY3cPSo8g&q=https%3A%2F%2Fmvnrepository.com%2F&redir_token=v8e0KZgLSH6EWcf6q0buHjGlJbZ8MTU4ODc1NzUyOEAxNTg4NjcxMTI4)

Step 4 : Connect device run command : adb devices

Step 5 : Start appium server can do from command line or appium desktop client

Step 6 : Add code to start automation on mobile device Set desired capabilities deviceName udid platformName platformVersion appPackage, appActivity [http://127.0.0.1:4723/wd/hub](https://www.youtube.com/redirect?event=video_description&v=N7vY3cPSo8g&q=http%3A%2F%2F127.0.0.1%3A4723%2Fwd%2Fhub&redir_token=v8e0KZgLSH6EWcf6q0buHjGlJbZ8MTU4ODc1NzUyOEAxNTg4NjcxMTI4) Start calculator application

Step 7 : Run and validate

======================  **How to use Appium Inspector** ===========================

Appium inspection is the process using which you can identify or find elements in your mobile app.

* Appium Desktop Inspector: The latest version of Appium GUI ([Appium Desktop](http://www.automationtestinghub.com/appium-desktop/)) comes with an inspector. You can use this inspector for both Android and iOS apps (for iOS apps, you would need a Mac).

1. Open Appium desktop client 2. Start appium server

3. Create new connection 4. Provide desired capabilities

5. Start session 6. Tap on objects to view the details

7. Add scripts in project and validate

**Step 1:** Add all the desiredcapabilities**,** then save it if we want to.. . start session, it will start the app.

**Step 2:** click on anything on device screen it will popup app source and Selected Element. we can find the elements from here.

**Step 3:** click on tap option to execute on the screen.

**Step 4:** write script :

DesiredCapabilities cap = **new** DesiredCapabilities();

cap.setCapability("deviceName", "G4");

cap.setCapability("udid", "VS98679aea3f3");

cap.setCapability("platformName", "android");

cap.setCapability("platformVersion", "6.0");

cap.setCapability("appPackage", "com.android.calculator2");

cap.setCapability("appActivity", "com.android.calculator2.Calculator");

URL url = **new** URL("http://127.0.0.1:4723/wd/hub");

driver = **new** AppiumDriver<MobileElement>(url, cap);

MobileElement two = driver.findelement(By.id("id value");

MobileElement plus = driver.findelement(By.id("id value");

MobileElement three = driver.findelement(By.id("id value");

MobileElement equals = driver.findelement(By.id("id value");

MobileElement result = driver.findelement(By.id("id value");

two.click();

plus.click();

three.click();

equals.click();

String resultValue = equals.getText();

System.out.println("result : "+ resultvalue);

=> we can perform the operation on popup\_appScreen using **tap** button.

==> Different Appium Inspectors that help you identify elements in mobile app.

**==> how to find MobileElement by using appium inspector: By Raghab Pal**

1. open appium - start server & click on start inspector button (top right corner )

2. add desired capabilities. click on (+) to add more desired capabilities.

3. click on save button - click on start session button.

4. click on any element on left side (device window), can see element source in middle

and select element at the right side.

5. get the ID, name, className, xpath from the right side window.

6. do desired operation on the left side mobile screen then click on tap button to execute the operation.

7. get all the locators and write the code.

===================== **How to use UIAutomatorViewer** ==========================

-UIAutomatorViewer: This is a tool provided by Android Studio that lets you inspect elements in your mobile app

There are 2 ways in which you can open UIAutomatorViewer. You can follow any of these methods to open it –

Method 1: Open UIAutomatorViewer from Command Prompt This is the easiest way of opening UIAutomatorViewer. Just follow the steps given below to open it –

**Step 1.1:** Open command prompt

**Step 1.2:** Type uiautomatorviewer and then hit Enter.

Method 2: Open UIAutomatorViewer from its folder location doing double click.

**Step 2.1:** Connect your mobile phone to your machine using USB

**Step 2.2 :** To make sure that the phone is connected properly, open command prompt and run adb devices command.

**Step 3:** Make sure phone is unlocked and **app** is opened

**Step 4:** Now click on Device Screenshot icon in UI Automator Viewer (this is the second icon at the top-left corner).

**Step 5:** Wait for a few seconds UI AUtomator Viewer will load the screenshot of your phone .

**Step 6:** Now, hover your mouse pointer anywhere on the device screenshot. You will see that some information gets displayed on the right side panes .

= 3 main sections in UI Automator Viewer =

**1. Left Side Pane:** This pane shows the device screenshot. This is an interactive image, where you can click or hover your mouse pointer over any element on the screenshot.

**2. Right Side Top Pane:** This pane shows the entire XML structure of the screen. This XML structure shows very basic details about each control – control name, its text (if available) and its coordinates.

**3. Right Side Bottom Pane:** This is an important pane which shows all the details of a control. Whatever element you select from the screenshot, this pane will show its entire details such as its text, class name, package name etc. This is the pane from where you will get information on how to identify an element uniquely.

= UIAutomatorViewer – Inspect mobile elements =

## Different ways to identify elements using Appium...

Find element by ID

* Find element by ClassName
* Find element by Tag Name
* Find element by Accessibility ID/ resource-id.
* Find element by XPath

Click on any element in UIAutomatorViewer, so that its details are displayed.

driver.findElement(By.id("com.android.vending:id/search\_box\_idle\_text")).click();

driver.findElementById("com.android.vending:id/search\_box\_idle\_text").click();

xpath : //className[@attribute='value']

List<MobileElement> elements = driver.findElements(By.className("android.widget.ImageView"));

driver.findElement(By.xpath("//android.widget.EditText[@text = 'Search Google Play']")).sendKeys("Google");

driver.findElement(By.xpath("//\*[@text = 'Search Google Play']")).sendKeys("Google"); ?

List<MobileElement> elements = driver.findElements(By.xpath("//android.widget.EditText"));

= Appium in Android provides a method called findElementByAndroidUIAutomator() which uses Android’s UiSelector to identify the mobile elements.

textStartsWith:

List<MobileElement> elementsStartingWithT = ((AndroidDriver<MobileElement>)driver).findElementsByAndroidUIAutomator("new UiSelector().textStartsWith(\"T\")");

textContains() method:

List<MobileElement> elementsTextContainsTO = ((AndroidDriver<MobileElement>)driver).findElementsByAndroidUIAutomator("new UiSelector().textContains(\"TO\")");

textMacthes() method:

List<MobileElement> elementsWith2Words = ((AndroidDriver<MobileElement>)driver).findElementsByAndroidUIAutomator("new UiSelector().textMatches(\"\\w+\\s{1}\\w+\")");

Using UiSelector with Resource ID

This method allows you to identify an element with its id or resource-id. This method provides two variations using which you can identify mobile elements – exact match and regular expression match.

//Identify an element using Resource ID (exact match)

MobileElement searchBox = ((AndroidDriver<MobileElement>)driver).findElementByAndroidUIAutomator("new UiSelector().resourceId(\"com.android.vending:id/search\_box\_idle\_text\")");

System.out.println("Search Box Name - " + searchBox.getAttribute("name"));

//Identify an element using Resource ID Matches (Regular Expression)

searchBox = ((AndroidDriver<MobileElement>)driver).findElementByAndroidUIAutomator("new UiSelector().resourceIdMatches(\".\*:id/search\_box\_idle\_text\")");

System.out.println("Search Box Name [RegEx] - " + searchBox.getAttribute("name"));

//Identify List of elements using Resource ID

List<MobileElement> allTitleElements = ((AndroidDriver<MobileElement>)driver).findElementsByAndroidUIAutomator("new UiSelector().resourceId(\"com.android.vending:id/title\")");

//Using UiSelector with Description:

These 4 variations are – description(), descriptionContains(), descriptionStartsWith() and descriptionMatches(). Since this method works the same way as text(), we will just show one example of how it works

//Identify element using its Content Description

MobileElement elementContentDesc = ((AndroidDriver<MobileElement>)driver).findElementByAndroidUIAutomator("new UiSelector().description(\"Search\")");

System.out.println("Element Class Name - " + elementContentDesc.getAttribute("className"));

== Save UI Automator Viewer Screenshot and Object Hierarchy Dump ==

**\*** Click on Save icon. It will ask for the location, where the screenshot has to be saved.

=== Open saved screenshot and UI dump ===

**\*** Click on folder icon from UI Automator Viewer window menu. It will display Open UI Dump Files popup.

**\*** Click on the Browse button for Screenshot section, navigate to the folder where you saved the screenshot and select the screenshot.

**\*** Do the same for UI XML dump as well. Once you select both the files, then the OK button would get enabled.

**\*** Click on OK button. The screenshot and its UI dump would be displayed in UI Automator Viewer window.

# Scroll :

TouchActions action = **new** TouchActions(driver);

action.scroll(element, 10, 100);

action.perform();

**# appium test framework setup:**

Step 1 : Open Eclipse

Step 2 : Install TestNG in eclipse[https://www.youtube.com/watch?v=yyUyi...](https://www.youtube.com/watch?v=yyUyi8s42dE&list=PLhW3qG5bs-L8dQYImOOArI1jaofMCc4hb)

Step 3 : Create a new maven project

Step 4 : Add required lib/dependencies Appium java client Selenium java TestNG

Step 5 : Save - Clean - Build project

Step 6 : Create 2 folders under src/test/java tests pages

Step 7 : User src/test/resources create a folder apps will keep application files (.apk, .ipa, .app)

Step 8 : Inside tests folder create a class - BaseClass and a Test class.

Step 9 : Create setup & teardown functions Annotate with testng annotations

Step 10 : Add code for desired capabilities

Step 11 : Attach mobile device (or make emulator ready) adb devices get id and platform version and add in desired capabilities.

Step 12 : Start appium server

Step 13 : Run & test

Step 14 : Add test class and extend BaseClass

Step 15 : Create some more sample tests

Step 16 : Run & Test .

==================== **how to connect mobile using wifi** ==========================

Step 1 : Ensure your computer and mobile phone are connected to same wifi network

Step 2 : Connect mobile with computer using cable

Step 3 : On cmd run adb devices. This will list the devices connected

Step 4 : Now on CMD run adb tcpip 5555 You can use any other port also You should get output like - restarting in TCP mode port: 5555 FREE - [https://automationstepbystep.com/onli...](https://www.youtube.com/redirect?event=video_description&redir_token=DSG3sBFMwLwxy1MSjfyxluiFHSN8MTU4ODc2NjgwNUAxNTg4NjgwNDA1&q=https%3A%2F%2Fautomationstepbystep.com%2Fonline-courses%2F&v=uw1Inr1wAiA)

Step 5 : Disconnect mobile cable

Step 6 : Get your mobile ip - Settings - About Phone - Status OR Settings - WiFi - your connected network - and note down the IP address

Step 7 : Run on cmd adb connect ip adb connect 192.168.1.4 if any issues can do add kill-server adb connect 192.168.1.4 OR add kill-server adb connect 192.168.1.4:5555

Step 8 : Now you can use your mobile without cable

Step 9 : To verify run adb devices command and it should list your device

# Chrome browser example:

System.setProperty("webdriver.chrome.driver","C:\\selenium\_drivers\\chromedriver.exe");

driver = new AndroidDriver<MobileElement>(new URL("http//127.0.0.1:4723"), caps);