#### **Know Activity & Package name:**

C:\Users\parag.mehta>adb shell

shell@geefhd:/ $ dumpsys window windows | grep -E 'mCurrentFocus|FocusedApp'

dumpsys window windows | grep -E 'mCurrentFocus|FocusedApp'

mCurrentFocus=Window{43a3bc18 u0 com.jio.mhood.jionet/com.jio.android.jionet.w

ankhade.WankhadeLoginActivity}

mFocusedApp=AppWindowToken{45d2dd68 token=Token{43bbc6c0 ActivityRecord{43836c

38 u0 com.jio.mhood.jionet/com.jio.android.jionet.wankhade.WankhadeLoginActivity

t104}}}

#### **Advantages  of  using  Appium :**

* It uses vendor-provided automation frameworks –
  + For iOS – Apple’s [UIAutomation](https://developer.apple.com/library/ios/documentation/DeveloperTools/Reference/UIAutomationRef/_index.html);
  + Android 4.2+ – Google’s [UiAutomator](http://developer.android.com/tools/help/uiautomator/index.html);
  + Android 2.3+ – Google’s [Instrumentation](http://developer.android.com/reference/android/app/Instrumentation.html)
* It uses WebDriver API (i.e ‘[Selenium WebDriver](http://docs.seleniumhq.org/projects/webdriver/)‘) which provides a client-server protocol (known as the [JSON Wire Protocol](https://code.google.com/p/selenium/wiki/JsonWireProtocol)) thus making it possible to write the test code in any language.
* Appium is a server which is written in [node.js](http://nodejs.org/) and can be built and installed from source or directly from [npm](https://www.npmjs.com/).
* There are [client libraries](http://appium.io/slate/en/v1.1.0/?ruby#list-of-client-libraries-with-appium-server-support) (in Java, Ruby, Python, PHP, JavaScript, and C#) which can be used instead of regular WebDriver client.
* Last but not the least – it is Open-Source.

#### Appium  on  Windows  (Requirements) :

Following things will be required as we work on appium android automation on windows platform –

1. [Appium.exe](https://bitbucket.org/appium/appium.app/downloads/)
2. [Android SDK](http://developer.android.com/sdk/index.html) API >= 17
3. [Java JDK](http://www.oracle.com/technetwork/java/javase/downloads/index.html)
4. [Apache Ant](http://ant.apache.org/bindownload.cgi)
5. [Eclipse](http://www.eclipse.org/downloads/?)
6. [Android Emulator](http://developer.android.com/tools/help/emulator.html) or an Android Device
7. Selenium Server (formerly the Selenium RC Server) version ([download link](http://docs.seleniumhq.org/download/))
8. Appium Client Library jar file (according to your language choice) ([download link](http://appium.io/downloads))
9. ‘[.apk](http://en.wikipedia.org/wiki/Android_application_package)‘ file of the app on which tests will be executed.

We would be using [java-client](https://github.com/appium/java-client) here for client library as we discuss further.

#### Pre – Conditions –

##### [i]  Set  Environment  Variables –

Make sure you set up environment variables correctly before executing any script. All these variables should be set under ‘System variables’ not in ‘User variables’.

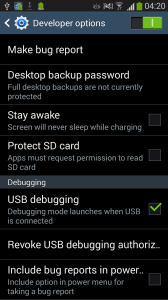
Depending on the location where you have jdk, android-sdk and ant installed in your machine, path should be specified . For example ,in my machine , path is specified as –

* **JAVA\_HOME** –
  + C:\Program Files\Java\jdk1.8.0\_25
* **ANDROID\_HOME** –
  + D:\Android\_Development\android\_sdk
* **ANT\_HOME** –
  + C:\apache-ant-1.9.4
* **PATH** variable (would contain System32 , platform-tools , tools ,ant bin and jdk bin set) –
  + C:\Windows\System32;D:\Android\_Development\android\_sdk\platform-tools;%JAVA\_HOME%\bin;D:\Android\_Development\android\_sdk\tools;%ANT\_HOME%\bin;

Class: Enter JDK path here

##### [ii]  USB Debugging ‘ON’ –

Make sure the device which you are connecting to your PC to run tests on has Developer Options enabled and USB debugging option is checked there as shown below –

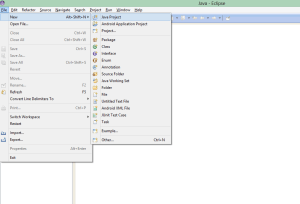
[](http://i1.wp.com/smritituteja.in/blog/wp-content/uploads/2014/12/usb_debugging.png)

#### Executing   Basic   Script –

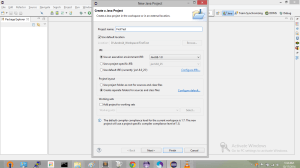
##### **FIRST   STEP  :  Creating   a   new   project**

[1] Create a new java project in Eclipse :

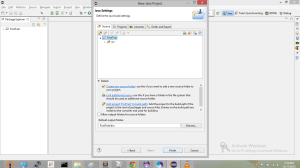
* Go to ‘File’ -> ‘New’ -> ‘Java Project’

[](http://i0.wp.com/smritituteja.in/blog/wp-content/uploads/2013/11/new_java_project.png)

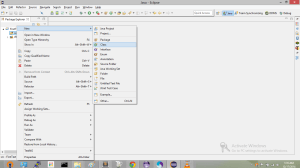
[2] Add project name and click ‘Next’ –

[](http://i1.wp.com/smritituteja.in/blog/wp-content/uploads/2014/12/new_java_project_name.png)

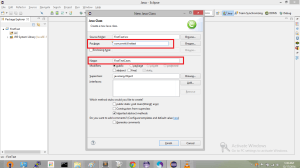
[3] Click ‘Finish’ –

[](http://i1.wp.com/smritituteja.in/blog/wp-content/uploads/2014/12/new_java_project_finish.png)

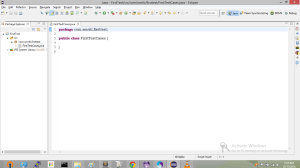
[4] Expand your project. Right click on src folder -> ‘New’ -> ‘Class’ –

[](http://i0.wp.com/smritituteja.in/blog/wp-content/uploads/2014/12/new_java_project_class.png)

[5] Enter ‘Name’ for class.  Also mention ‘Package’ and then click ‘Finish’ –

[](http://i0.wp.com/smritituteja.in/blog/wp-content/uploads/2014/12/new_java_project_class_name.png)

[6] You will see something like –

[](http://i2.wp.com/smritituteja.in/blog/wp-content/uploads/2014/12/new_java_project_empty_created.png)

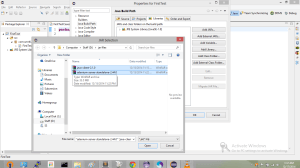
##### **SECOND   STEP  :  Adding   required   jar   files**

[1] You will need to export required selenium jar files in your project.

* Selenium Server (formerly the Selenium RC Server) version [here](http://docs.seleniumhq.org/download/).
* Appium Client Library jar file (according to your language choice) from [here](http://appium.io/downloads).

[2] Import all these jar files in your project.

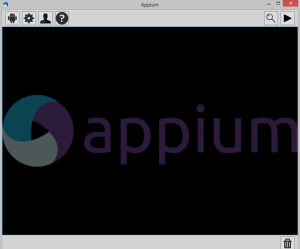
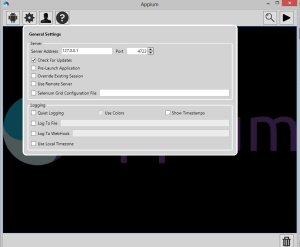
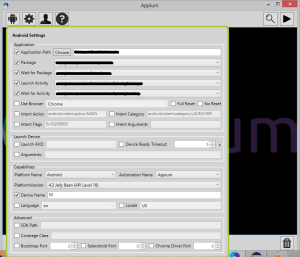
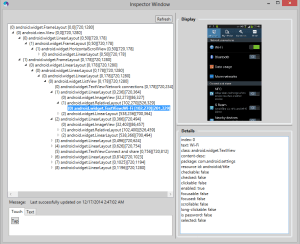
* Right click on your project -> Properties.
* Go to Libraries section -> Add External JARs (Note: I am using [java client](https://github.com/appium/java-client) here)
* Select both jar files which needs to be added -> click on ‘Open’ -> then tap ‘OK’.

[](http://i0.wp.com/smritituteja.in/blog/wp-content/uploads/2014/12/add_jar_files.png)

##### **THIRD   STEP  :  Download  Appium**

Download Appium for Windows zip file [here](https://bitbucket.org/appium/appium.app/downloads/) and extract the zip file contents. After Appium is downloaded , you will find an “Appium.exe” application inside extracted ‘Appium’ folder.

##### **FOURTH   STEP  :  Understanding   Appium   Parameters**

1. Launch Appium.exe and you would see following buttons –
   1. Android button
   2. Settings button
   3. Developer Settings
   4. About
   5. Inspector (magnifying glass icon)
   6. Launch/Stop button
   7. Delete (trash icon)
2. [](http://i0.wp.com/smritituteja.in/blog/wp-content/uploads/2014/12/launch_appium.png)Hover over gear icon and you would see “General Settings”. Tap on it. Some important points to note here –
   1. You would see “Server Address” (IP address on which you want the Appium server to run) and “Port” (port on which the Appium server will listen for WebDriver commands)  as it shows “127.0.0.1” (localhost) and “4723” respectively here –
   2. “Check for Updates” is to check regular Appium updates.
   3. “Pre-Launch Application” is to prelaunch the application before any commands start executing.
   4. “Override Existing Session” can be checked to override any existing Appium sessions(if any).
3. Now hover over android icon and you would see “Android Settings. Tap on it and you will see a window opens up with too many fields.
4. [](http://i2.wp.com/smritituteja.in/blog/wp-content/uploads/2014/12/appium_android_settings1.png)Under “Application” section , some fields which we can consider here –
   1. “Application Path” – You should check this option and can tap on ‘Choose’ and give the path of your ‘.apk’ file downloaded which you want to test.
   2. “Package” – This is the package name of your test application (ex – com.example.testApp)
   3. “Launch Activity” – Activity name for the Android activity which needs to be launched from your package.
5. Under “Capabilities” section –
   1. “Platform Name” – mobile platform.
   2. “Automation Name” – automation tool (Android or Selendroid).
   3. “Platform Version” – mobile platform version.
   4. “Device Name” – name of the device you would be testing on.
6. Hover on the magnifying glass and you can all the elements accessible on the screen. Appium should be running for inspector to work. An example screenshot of “Settings” section of an android device can be viewed in Inspector as –
7. Through appium inspector – you can view name , value , label , xpath of different elements on the screen. Given below are some examples how to use elements from the screen using xpath , tagName, name, attribute etc –
8. //webdriver declaration
9. public WebDriver driver ;
10. //using xpath
11. driver.findElement(By.xpath("//window[1]/scrollview[1]/button[1]"));
12. //using name
13. driver.findElement(By.name("Go"));
14. //using attribute –
15. WebElement x = driver.findElement( By.xpath("//window[1]/tableview[1]/cell[2]/text[1]");
16. x.getAttribute("name"); //get name of the element
17. //using tagName
18. driver.findElement(By.tagName("button"));
19. //sendkeys can be used for giving some input in any field
20. WebElement login\_email = driver.findElement(By.xpath("//window[1]/scrollview[1]/textfield[1]"));
21. login\_email.click();
22. login\_email.sendKeys("smriti@abc.com");
23. //provide delay between two event clicks
24. Thread.sleep(time in milliseconds)

**Note** – You can run your test scripts either in simulator or on real device. Refer about Appium parameters detailed information [here](https://github.com/appium/appium-dot-exe).

##### **FIFTH   STEP  :  Writing  Script**

Now we will have a look at one sample script.

Following desired capabilities need to be taken care of while writing script for android automation ([Desired capabilities](http://appium.io/slate/en/master/?ruby#appium-server-capabilities) are a set of keys to communicate with Appium server to tell the server what kind of automation session you are interested in) –

* platformName (mobile platform)
* platformVersion (mobile platform version)
* deviceName (name of device on which you are running test)
* app (path to your ‘.apk’ file)
* appPackage (app package)
* appActivity (main activity class)

1. package com.smriti.firsttest;
2. import java.net.URL;
3. import org.openqa.selenium.By;
4. import org.openqa.selenium.WebDriver;
5. import org.openqa.selenium.remote.DesiredCapabilities;
6. import org.openqa.selenium.remote.RemoteWebDriver;
7. public class FirstTestCases {
8. // webdriver declaration
9. public WebDriver driver;
10. public static void main(String[] args) throws Exception {
11. // creating a class object
12. FirstTestCases testcases = new FirstTestCases();
13. // call launch app method
14. testcases.launchApp();
15. // call add name method
16. testcases.addName();
17. }
18. public void launchApp() throws Exception {
19. System.out.println("Launching App");
20. DesiredCapabilities capabilities = new DesiredCapabilities();
21. capabilities.setCapability("platformName", "Android");
22. capabilities.setCapability("platformVersion", "4.3");
23. capabilities.setCapability("deviceName", "S3");
24. capabilities.setCapability("app", "D:\\Apps\\TestApp.apk");
25. capabilities.setCapability("app-package", "com.example.testApp");
26. capabilities.setCapability("app-activity", ".SplashActivity");
27. driver = new RemoteWebDriver(new URL("http://127.0.0.1:4723/wd/hub"),
28. capabilities);
29. Thread.sleep(20000); // delay of 20s
30. System.out.println("App Launched");
31. }
32. public void addName() throws InterruptedException {
33. System.out.println("Adding Name");
34. String myname = "Smriti";
35. driver.findElement(By.name("Add")).click();
36. driver.findElement(By.xpath("//textfield[1]")).sendKeys(myname);
37. driver.findElement(By.name("Save")).click();
38. Thread.sleep(5000); // 5s delay
39. // write all your tests here
40. }
41. }

**This worked:**

**package** Test;

**import** java.io.File;

**import** java.net.MalformedURLException;

**import** java.net.URL;

**import** java.util.concurrent.TimeUnit;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.remote.DesiredCapabilities;

**import** org.openqa.selenium.remote.RemoteWebDriver;

**public** **class** FirstTest {

// webdriver declaration

**public** WebDriver driver;

@Before

**public** **void** setUp() **throws** Exception {

// set up appium

File appDir = **new** File("D:\\Appium\_jars\\Test\_apk\\");

File app = **new** File(appDir, "Jionet\_1.6.10.apk"); //my case “demo1.apk”

DesiredCapabilities capabilities = **new** DesiredCapabilities();

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

capabilities.setCapability("platformVersion", "4.3");

capabilities.setCapability("deviceName", "LGE988dc16d10e");

capabilities.setCapability("app", "D:\\Appium\_jars\\Test\_apk\\Jionet\_1.6.10.apk");

capabilities.setCapability("app-package", "com.jio.mhood.jionet");

capabilities.setCapability("app-activity", "/com.jio.android.jionet.wankhade.WankhadeLoginActivity");

driver = **new** RemoteWebDriver(**new** URL("http://127.0.0.1:4723/wd/hub"), capabilities);

Thread.*sleep*(20000); // delay of 20s

System.*out*.println("App Launched");

}

@After

**public** **void** tearDown() **throws** Exception {

driver.quit();

}

@Test

**public** **void** addName() **throws** InterruptedException {

// find user-name input field

WebElement ByPass = driver.findElement(By.*id*("com.jio.mhood.jionet:id/gotit"));

ByPass.click();

WebElement OpenLogin = driver.findElement(By.*className*("android.widget.ImageButton"));

OpenLogin.click();

WebElement LoginScreen = driver.findElement(By.*id*("com.jio.mhood.jionet:id/imageButton1"));

LoginScreen.click();

WebElement userNameInput = driver.findElement(By.*id*("login\_username"));

userNameInput.sendKeys("pujapriya");

WebElement passwordInput = driver.findElement(By.*id*("com.jio.mhood.jionet:id/login\_password"));

passwordInput.click();

passwordInput.sendKeys("puja-12345");

driver.navigate().back();

WebElement login = driver.findElement(By.*id*("com.jio.mhood.jionet:id/btnLogin"));

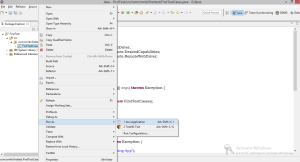
login.click();

}

}

##### **SIXTH   STEP  :  Run  Script**

After writing the test script , follow these steps to run your test –

1. Open Appium.exe.
2. Go to ‘General Settings’ and check that ‘Server Address’ and ‘Port’ should be set as “127.0.0.1” and “4723” respectively.
3. Go to ‘Android Settings’ add your ‘.apk’ file path(path where you downloaded the apk file in your machine) under “Application Path”. You will notice that it takes up ‘Package’ and ‘Launch Activity’ paths automatically when you enter the apk file path.
4. Under ‘Capabilities’ section –
   1. Make sure ‘Platform Name’ and  ‘Automation Name’ are set to ‘Android’ and ‘Appium’ respectively considering you aren’t using older android versions. If yes , chose ‘Selendroid’.
   2. Select ‘PlatformVersion’ based on mobile platform version you are working on.
   3. Select checkbox next to Device Name and mention your device name there say “S3″ if my Samsung S3 device name is S3.
5. Now trigger Appium server by tapping on launch button.
6. Now go to Eclipse and run your project as a ‘Java Application’ –
7. [](http://i0.wp.com/smritituteja.in/blog/wp-content/uploads/2014/12/run_java_file.png)Once Appium server is triggered and your test app launches , you can tap on Appium Inspector and view all accessible elements on any active screen and write test cases accordingly and execute.

 Frameworks:

<http://www.oracle.com/technetwork/articles/entarch/shrivastava-automated-frameworks-1692936.html>

<http://www.testingexcellence.com/test-automation-framework-designs/>

<http://simpleprogrammer.com/2011/03/16/getting-up-to-bat-designing-an-automation-framework/>