**How to find AppPackage and Activity Name:**

Launch Command Propt and got to “D:\android\sdk\build-tools\24.0.3>” location and copy the application here in this path and from command prompt give the command “aapt dump badging ContactManager.apk”

If you want to download any app from play store

1:Got to <https://play.google.com/> and search for the appname and copy the URL

2:Launch <https://apps.evozi.com/apk-downloader/> and paste the URL here and generate it

3:Press “Click here to download “

If we want to add the java doc for appium->right click on JavaClient from the referenced libraries

Desired capabilities is a class ,as soon as we create an object we need to create some methods which gone to use for appium.

Gson and javaclient jars we need to add for appium and selenium standalone webdriver jar to detect selenium commands

**If you want to start appium server programmatically, using node.exe and appium.js**

**public** **class** startstopserver {

**public** **static** Process *process*;

**public** **static** AndroidDriver *driver*;

**public** **static** String *StartServer* = "D:\\AppiumForWindows-1.3.4.1\_001\\Appium\\node.exe D:\\AppiumForWindows-1.3.4.1\_001\\Appium\\node\_modules\\appium\\bin\\appium.js";

**public** **static** **void** startAppiumServer() **throws** IOException, InterruptedException {

Runtime runtime = Runtime.*getRuntime*();

*process* = runtime.exec(*StartServer*);

Thread.*sleep*(7000);

**if** (*process* != **null**) {

System.*out*.println("Appium server started");

}

}

//Stopping the Appium Server

**public** **static** **void** stopAppiumServer() **throws** IOException {

**if** (*process* != **null**) {

*process*.destroy();

}

System.*out*.println("Appium server stopped");

}

**public** **static** **void** main(String[] args) **throws** Exception{

*stopAppiumServer*();

*startAppiumServer*();

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

capabilities.setCapability("automationName", "Appium");

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

capabilities.setCapability("platformVersion","4.4.2");

capabilities.setCapability("deviceName","Galaxy Note3");

capabilities.setCapability("appPackage","io.selendroid.testapp");

capabilities.setCapability("appActivity","io.selendroid.testapp.HomeScreenActivity");

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

driver.manage().timeouts().implicitlyWait(30,TimeUnit.*SECONDS*);

driver.findElement(By.*id*("io.selendroid.testapp:id/my\_text\_field")).sendKeys("Hello Appium");

Thread.*sleep*(3000);

driver.quit();

*stopAppiumServer*();

}

}

AppiumDriver will not support on the latest release ,instead we use **AndroidDriver**.

Every testing scenario should be executed on some specific testing environment. The testing environment can be a web browser, mobile device, mobile emulator, mobile simulator, etc.

The **Desired Capabilities** Class helps us to tell the webdriver, which environment we are going to use in our test script.

It is used to set the browser properties (Ex. Chrome, IE), Platform Name (Ex. Linux, Windows) that are used while executing the test cases.

In the case of mobile automation, as we perform the tests on different varieties of mobile devices, the Mobile Platform (ex. iOS, Android) Platform Version (Ex. 3.x,4.x in Android) can be set.

**What is Desired Capability**

The desired capability is a series of key/value pairs that stores the browser properties like browsername, browser version, the path of the browser driver in the system, etc. to determine the behaviour of the browser at run time.

* Desired capability can also be used to configure the driver instance of Selenium WebDriver.
* We can configure driver instance like FirefoxDriver, ChromeDriver, InternetExplorerDriver by using desired capabilities.

Desired Capabilities are more useful in cases like:

* In mobile application automation, where the browser properties and the device properties can be set.
* In Selenium grid when we want to run the test cases on a different browser with different operating systems and versions.

**Different types of Desired Capabilities Methods**

Here we will see a different type of desired capabilities methods and see how to use one of this method "**setCapability Method".**

1. **getBrowserName()**

public java.lang.String getBrowserName()

1. **setBrowserName()**

public void setBrowserName(java.lang.String browserName)

1. **getVersion()**

public java.lang.String getVersion()

1. **setVersion()**

public void setVersion(java.lang.String version)

1. **getPlatform()**

public Platform getPlatform()

1. **setPlatform()**

public Platform getPlatform()

1. **getCapability Method**

The getCapability method of the Desired Capabilities class can be used to get the capability that is in use currently in the system.

public java.lang.Object getCapability(java.lang.String capabilityName)

1. **setCapabilityMethod**

The setCapability() method of the Desired Capabilities class can be used to set the device name, platform version, platform name, absolute path of the app under test (the .apk file of the app(Android) under test), app Activity (in Android) and appPackage(java).

**"setCapability method"** in java has the below declarations:

**UIAutomatorViewer:**

If we have more than one element with common properties (Class name,similar type elements in the screen use findElements instead of findElement

driver.findElements(By.className("android.widget.Button")).size());

size method displays the list of buttons or text field that to be there in the app using findElemets

Display only the button which is displayed on the screen

Import: import java.util.List;

List<WebElement> AllButton = driver.findElements(By.*className*("android.widget.Button"));

**for**(WebElement widbtn:AllButton){

System.*out*.println(widbtn.isDisplayed());

System.*out*.println(widbtn.getText());

}

Display only the Linear Layout which is displayed

List<WebElement> LinearLayout = driver.findElements(By.*className*("android.widget.LinearLayout"));

**for**(WebElement widbtn:LinearLayout){

} System.*out*.println(widbtn.isDisplayed());

If you want to click the button which is inside the Linear Layout then

WebElement finalIndex = (WebElement) driver.findElements(By.*className*("android.widget.LinearLayout")).get(6);

System.*out*.println(finalIndex.findElements(By.*className*("android.widget.Button")).size());

System.*out*.println("Linear Layout got clicked");

Thread.*sleep*(5000);

finalIndex.findElements(By.className("android.widget.Button")).get(1).click();;

ByXpath:

driver.findElement(By.xpath("//android.widget.Button[contains(@text,'Display Popup Window')]")).click();

or

driver.findElement(By.xpath("//android.widget.Button[@text='Display Popup Window']")).click();

**Another way of selecting findElements using AndroidUIAutomator—UI Selector**

driver.findElementByAndroidUIAutomator("UiSelector().className(\"android.widget.Button\").text(\"Display Popup Window\")").click();

driver.findElementByAndroidUIAutomator("UiSelector().className(\"android.widget.Button\").text(\"Display Popup Window\")").click();

// driver.findElementByAndroidUIAutomator("UiSelector().className(\"android.widget.CheckBox\").checked(true)").click();

/\*

System.out.println(driver.findElementsByAndroidUIAutomator("UiSelector().className(\"android.widget.LinearLayout\").className(\"android.widget.Button\").index(1)").size());

WebElement button = driver.findElementsByAndroidUIAutomator("UiSelector().className(\"android.widget.LinearLayout\").className(\"android.widget.Button\").index(1)").get(1);

button.click();\*/

using Thread.sleep(1000) is not a good practice,this will actually halt for 10 seconds

**Implicit Wait:**if we declare implicit wait in the start then where every driver call,if the element not found, execution will wait for 10 seconds

Driver.manage().timeout().ImplicitlyWait(10L,TimeInit.Seconds);

**Explit Wait:**will be tied to a particular element only.for example we have given imlicit waiting time as 10 sec and one element will display after 15 dec,then the script faill.inorder to come out we will use explicitly wait for the particlaur element.

WebDeriverWait wait = new WebDriverWait(driver.10L)

Wait.until(ExpectedConditions.presenceOfElementLocated(By.id(“”))).click();

It tied to the particular element thought it will not be used in any driver.

**How we do internal dialer application:First** import an project called C:\Users\mohan.m.gowda\AppiumproSeptember\ActivityAndPackageInfo to the eclipse and run it using androidApplication

2:in Console it gives all the Package and Activit names of the application

WebElement frameLayout = driver.findElement(By.className("android.widget.FrameLayout"));

WebElement HorizontalScrollView = frameLayout.findElement(By.className("android.widget.HorizontalScrollView"));

List<WebElement> actionTab = HorizontalScrollView.findElements(By.className("android.app.ActionBar$Tab"));

actionTab.get(0).click();

WebElement linearLayout = driver.findElement(By.id("com.android.contacts:id/dialpad"));

//9711191558

linearLayout.findElement(By.id("com.android.contacts:id/nine")).click();

linearLayout.findElement(By.id("com.android.contacts:id/seven")).click();

linearLayout.findElement(By.id("com.android.contacts:id/one")).click();

linearLayout.findElement(By.id("com.android.contacts:id/one")).click();

linearLayout.findElement(By.id("com.android.contacts:id/one")).click();

if more than one element present in the page then use list<element> as below.

driver.findElements(By.id("com.mobeta.android.demodslv:id/activity\_title")).get(0).click();

List<WebElement> element = driver.findElements(By.id("com.mobeta.android.demodslv:id/drag\_handle"));

**Touch Action for Drag and drop**

new TouchAction((MobileDriver)driver).longPress(element.get(0)).moveTo(element.get(3)).release().perform();

**If you want to do any action on AndroidKey pad then use**

*driver*.pressKeyCode(AndroidKeyCode.*HOME*);

*driver*.pressKeyCode(AndroidKeyCode.*BACK*);

PressKeycode used in 3.4.1 and above , below 3.4.1 used SendKeyEvent

**If you want to click some element using Root key then use below code**

WebElement frameLayout=*driver*.findElement(By.*className*("android.widget.FrameLayout"));

WebElement HorizontalScrollView= frameLayout.findElement(By.*id*("flipboard.app:id/main\_activity\_sliding\_tabs"));

WebElement LinearLayout = HorizontalScrollView.findElement(By.*className*("android.widget.LinearLayout")); List<WebElement> actiontab=LinearLayout.findElements(By.*className*("android.widget.ImageView"));

actiontab.get(2).click();

**Swipe using X and Y Axis**

Install flipboard app and swipe between x and y axis.For getting x and y axis one easy method is to turn on “Show Pointer Location” from Developer option



Touch the screen and it says the x and y axis of the screen then use

Driver.swipe(startx , starty , endx, endy,time)

EX:

*driver*.swipe(506, 1149, 506, 662, 1000);

Thread.*sleep*(5000);

*driver*.swipe(506, 1149, 506, 662, 1000);

**Appium GRID:**

The same test on multiple devices

Localhost:4444/grid/console#

1:Go to the Selenium stand alone location from the command prompt and give **java -jar selenium-server-standalone-2.52.0.jar -role hub** to start the hub

D:\SeleniumJars

2:after starting open browser and give <http://localhost:4444/grid/console>

3:create a file with the extension .js/.json(node1.js and node2.js) under C:\Users\mohan.m.gowda\Desktop\LearnForInterview and enter all the values

Both files are same but need to change device ID,Port and URL as below

First Port: "url":"http://127.0.0.1:4723/wd/hub","port": 4723,

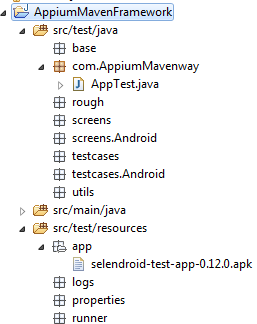
Second Port: "url":"http://127.0.0.1:4728/wd/hub", "port": 4728,

node.exe D:\AppiumForWindows-1.3.4.1\_001\Appium\node\_modules\appium\bin\appium.js --address 127.0.0.1 --port 4723 -bp 2251 --udid 498bee11 --nodeconfig C:\Users\mohan.m.gowda\Desktop\LearnForInterview\node1.js

node.exe D:\AppiumForWindows-1.3.4.1\_001\Appium\node\_modules\appium\bin\appium.js --address 127.0.0.1 --port 4728 -bp 2252 --udid 4d00c5fe2fee504f --nodeconfig C:\Users\mohan.m.gowda\Desktop\LearnForInterview\node2.js

PageObjectModel Automation Framework:

Create a folders as below and copy the app under app package.this entire code is there under C:\Users\mohan.m.gowda\AppiumproSeptember\AppiumMavenFramework



2:Create a Android\_selendroidtestapp.properties under properties to specify all the desired capabilities for android.

explicit.wait=120

default.wait=3

implicit.wait=10

application.path=C:\\selenium\\MobileFramework\\appiumframework\\src\\test\\resources\\app\\selendroid-test-app-0.12.0.apk

base.pkg=io.selendroid.testapp

appium.server.port=4723

application.activity=.HomeScreenActivity

automation.instumentation=Appium

browser.name=null

platform.name=Android

device.name=Android

platform.version=4.4.2

new.command.timeout=300

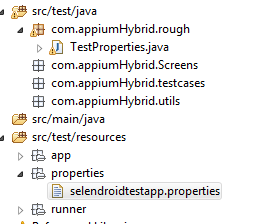
device.ready.timeout=300

3:Create a class under base package called “sreenbase” which contains all the common properties.

1:create a package under src/test/java as “com.w2a.appium.screens” where all business class will be saved here.for example :the calculator app,all the business logic relating to the functionality

will be comes under here

2:Crate the packages as below.



3:paste apk files under app package.

4:create .properties files and store all the delevirables as below.

explicit.wait=120

default.wait=3

implicit.wait=10

application.path=D:\\AppiumHybridFrameWork\\AppiumHybridFM\\src\\test\\resources\\app\\selendroid-test-app-0.12.0.apk

base.pkg=io.selendroid.testapp

appium.server.port=4723

application.activity=.HomeScreenActivity

automation.instumentation=Appium

browser.name=null

platform.name=Android

device.name=Android

platform.version=4.4.2

new.command.timeout=300

device.ready.timeout=300

5:if you want to fethch properties files data then use following code

**public** **static** Properties *prop* = **new** Properties();

**public** **static** **void** main(String[] args) **throws** IOException {

FileInputStream fls = **new** FileInputStream("D:\\AppiumHybridFrameWork\\AppiumHybridFM\\src\\test\\resources\\properties\\selendroidtestapp.properties");

*prop*.load(fls);

// System.out.println(fls.available());

System.*out*.println(*prop*.getProperty("base.pkg"));

System.*out*.println(*prop*.getProperty("platform.name"));

6:if you want to print the current directory path then use

System.out.println(**System.getProperty("user.dir"));**

Output: D:\AppiumHybridFrameWork\AppiumHybridFM

So the above FileInputStream can be gave as

System.getProperty(“user.dir”)+ [\\src\\test\\resources\\properties\\selendroidtestapp.properties](file:///\\src\\test\\resources\\properties\\selendroidtestapp.properties)

7: creat a class called CommonUtils under utils package where we can save all the common properties

**private** **static** Properties *prop* = **new** Properties();

**public** **static** **int** *EXPLICIT\_WAIT\_TIME*;

**public** **static** **int** *IMPLICIT\_WAIT\_TIME*;

**public** **static** **int** *DEFAULT\_WAIT\_TIME*;

**public** **static** String *APPLICATION\_NAME*;

**public** **static** String *BASE\_PKG*;

**public** **static** String *APP\_ACTIVITY*;

**public** **static** String *APP\_PASSWORD*;

**private** **static** String *APPIUM\_PORT*;

**public** **static** String *AUTOMATION\_INSTRUMENTATION*;

**public** **static** String *BROWSER\_NAME*;

**public** **static** String *PLATFORM\_NAME*;

**public** **static** String *NEW\_COMMAND\_TIMEOUT*;

**public** **static** String *PLATFORM\_VERSION*;

**public** **static** String *DEVICE\_READY\_TIMEOUT*;

**public** **static** String *DEVICE\_NAME*;

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

**private** **static** URL *serverUrl*;

**private** **static** AndroidDriver *driver*;

5:Create packages for particular screens like web,hybrid and native

6:under native app, create a class called SelendroidHomeScreen

7:initiate a loadfactory method for initializing PageFactory Object

**public** **void** loadFactory(){

PageFactory.*initElements*(**new** AppiumFieldDecorator(*driver*), **this**);

}

8:create a page factory element for the editfeild

@AndroidFindBy(id="io.selendroid.testapp:id/my\_text\_field")

**public** WebElement textfeild;

9:call the load factory by creating constructor.

**public** SelendoidHomeScreen(AndroidDriver driver){

**this**.*driver*=driver;

loadFactory();

}

10:Create another class for designing Webview Interaction Screen

11:we have to perform same actions that we did on SelendroidHomeScreen class like creating page factory ,Instantiating page factory and calling loadFactory() method using Costructor

So create a base class to access all common element

12:for this create a ScreenBase class

13:the main objective of Page object model ,method should have the return type and this Selendroid class “SelendroidScreen” will be the return method as it is not going to any other screen

14:so on the SelendroidScreen class create a return type as below.

**public** SelendoidHomeScreen typeData(String data){

textfeild.sendKeys(data);

**return** **this**;

}

15:Create another class under rough package and call “selendroidtestapp.properties” from the commonutil class and call setCapabilities also

**public** **static** AndroidDriver *driver*;

**public** **static** **void** main(String[] args) **throws** IOException {

// **TODO** Auto-generated method stub

CommonUtils.*loadConfigProp*("selendroidtestapp.properties");

CommonUtils.*setCapabilities*();

*driver*= CommonUtils.*getDriver*();

SelendoidHomeScreen sh = **new** SelendoidHomeScreen(*driver*);

sh.typeData("Mohan");

16:create as many methods for validations and call from the test class

17:Create another class for “WebViewInteractionScreen” and add the methods

@AndroidFindBys({

@AndroidFindBy(id="io.selendroid.testapp:id/tableHeader"),

@AndroidFindBy(className="android.widget.TextView")

})

**public** WebElement webTextview;

**public** WebViewInteractionScreen getText(){

System.*out*.println(webTextview.getText());

Assert.*assertEquals*(webTextview.getText(), "Web View Interaction");

System.*out*.println(*driver*.getPageSource());

**return** **this**;

18: Finally call WebViewInteractionScreen methods from testclass

sh.displayBtn();

WebViewInteractionScreen webView=sh.openWebview();

webView.getText();

**Adding Testcases to Testng**

Create a class called “selendroidHomeScreenTest.java” and WebVeiewScreenTest and add the code as below

**public** **class** selendroidHomeScreenTest {

**public** **static** AndroidDriver *driver*;

@BeforeTest

**public** **void** SetUp() **throws** IOException{

CommonUtils.*loadConfigProp*("selendroidtestapp.properties");

CommonUtils.*setCapabilities*();

*driver*= CommonUtils.*getDriver*();

}

@Test

**public** **void** validateHomeScreenTest(){

SelendoidHomeScreen sh = **new** SelendoidHomeScreen(*driver*);

sh.typeData("Mohan");

sh.getAllButtons();

sh.displayBtn();

WebViewInteractionScreen webView=sh.openWebview();

}

2:under WebVeiewScreenTest class put the code as below

**public** **static** AndroidDriver *driver*;

@Test

**public** **void** validateWebView(){

WebViewInteractionScreen webView= **new** WebViewInteractionScreen(*driver*);

webView.getText();

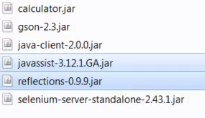
}

3:every time we have to initialize **public** **static** AndroidDriver *driver*; on every class so create a baseclass and add the following.whih is the super class for the selendroid and webViewscreetest classes

**Page Factory using Appium**

If you are using page Factory to find elements then use the below jars to highlight. Which I stored under appium jars.

The main objective to use page factory is , initially you declare all your element which to be used in your application and then use the variable only rather than using “driver.findElementBy” always



**1:**create an @AndroidFindElementBy as below

@AndroidFindBy(id="io.selendroid.testapp:id/my\_text\_field")

public WebElement textField;

2:Then initialize a page factory object and then call the variable and do action

public void doTesting(){

PageFactory.initElements(new AppiumFieldDecorator(driver,20,TimeUnit.SECONDS), this);

3:Call the variable and perform the action

textField.sendKeys("Rahul");

Example:

@AndroidFindBy(id="io.selendroid.testapp:id/my\_text\_field")

**public** WebElement textfeild;

@AndroidFindBy(id="io.selendroid.testapp:id/visibleButtonTest")

**public** WebElement displaybtn;

@AndroidFindBy(id="io.selendroid.testapp:id/visibleTextView")

**public** WebElement displayText;

@AndroidFindBy(className = "android.widget.Button")

**public** List<WebElement> button;

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

PageFactory.*initElements*(**new** AppiumFieldDecorator(driver, 20,TimeUnit.*SECONDS*), **this**);

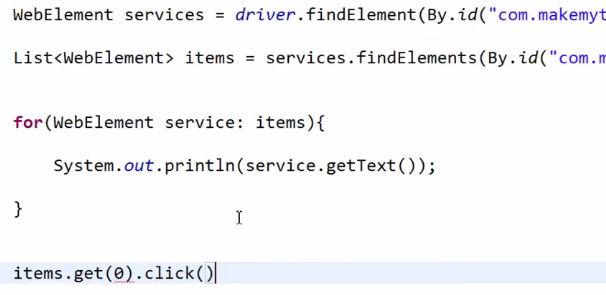
textfeild.sendKeys("Hello world");

displaybtn.click();

System.*out*.println(displayText.getText());

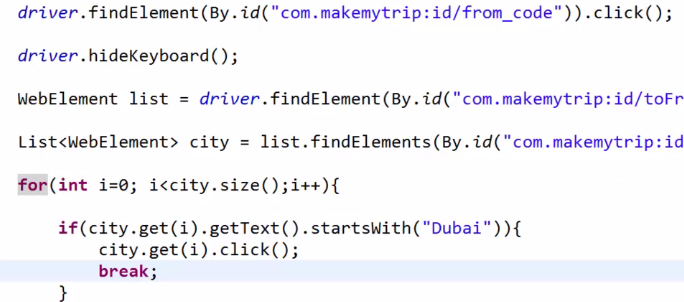
driver.hideKeyboard();

System.*out*.println(button.size());



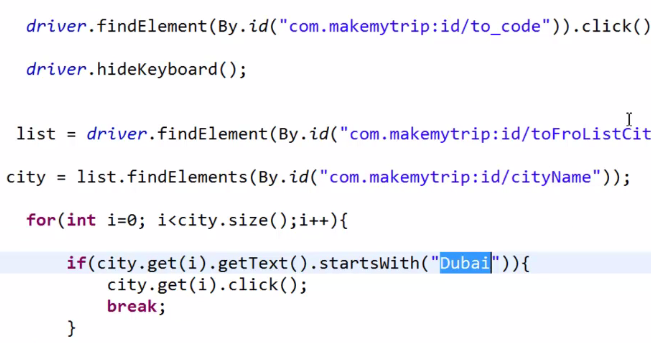
Testcase 2:

Select from dropdown and select dubai from the list



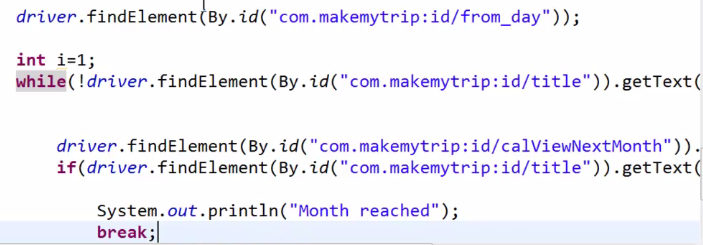
TestCase 3:

Select drinagar from the list



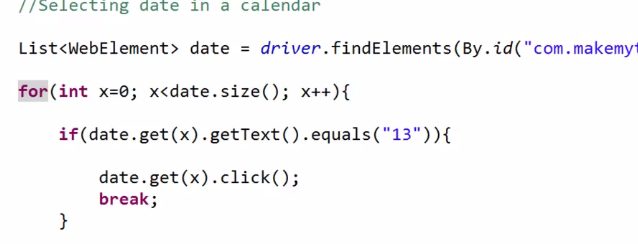
TestCase 4:

Select December month from the calendar

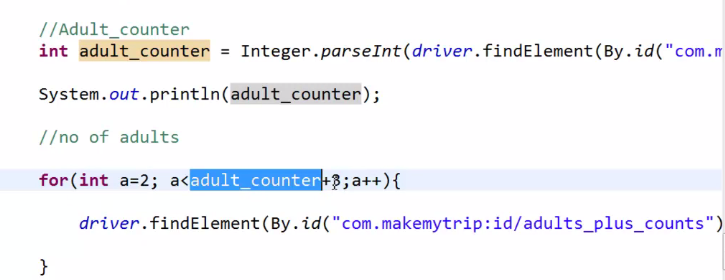


TestCase 5:

Select date in the calendar



Increase the adult count as 2



First line gettext

Inside for click();

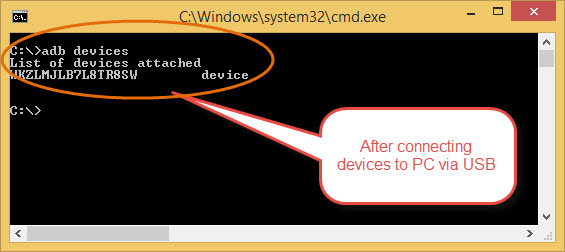
Under “ScreenBase” class write explicit wait function where will be used commonly whenever required

# Run Appium Tests On Real Android Device over Wifi

In the previous tutorials, we have seen executing Appium tests on a real devices by plugging USB to PC. Now here will execute Appium tests on wifi by enabling wireless adb on the device without connecting to PC with a USB cable.

Before proceeding further, you should make sure Android SDK is configured in your machine AND should have enabled ‘USB debugging’ from 'Developer Options' in your android device.

**Step 1** :- Connect your device to computer via USB and check if it is connected using 'adb devices' which will list you the devices connected to your machine.



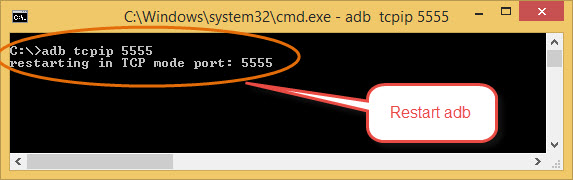
**Step 2** :- We should make sure that android device and the PC are connected to the same Wifi network. If you are trying to work on your desktop, you may need help with 'Wireless USB Adapters' which you can get them online and then connect your desktop also to the Wifi network.

**Step 3** :- Now we need to restart adb and make it work over tcpip by specifying the port value. If no port number is specified, Port 5555 is used by default.

adb tcpip <**port**>

***"adb tcpip 5555"****//- restarts the adb daemon listening on TCP on the specified port - 5555*

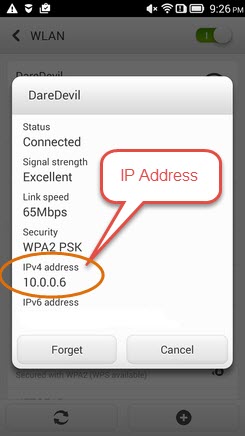
In the above command, 'adb tcpip' command actually tries to reconfigure and restart the adb daemon on the device. By default in most of the cases it will be using USB. After executing the above command adb daemon will now start listening to tcpip.



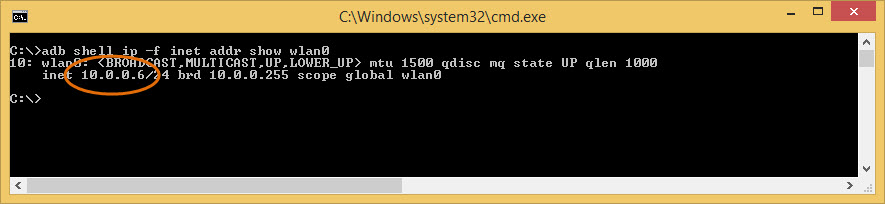
**Step 3** :- Now disconnect the device which is connected to PC

**Step 4** :- We need to get the IP address of the the device. There are two ways to get IP address of your device

Firstly, you can get that from your device -> Settings -> Wifi -> Wifi Settings / You can click on your wifi network which will popup details as shown below :

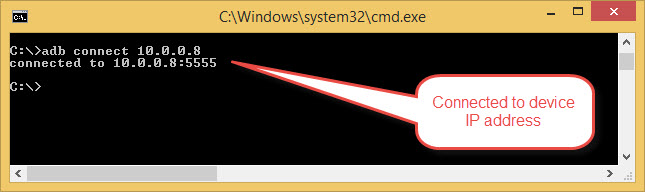


Alternatively, we can run the command 'adb shell ip -f inet addr show wlan0' which will show you the ip address of the connected device.



**Step 5** :- Run the below command to connect adb to your device over Wi-Fi using IP address of your device:

adb **connect** device\_IP\_Address



That's all, We are DONE!!!!!!!!!!. Now we can execute our tests on Real device over wifi.

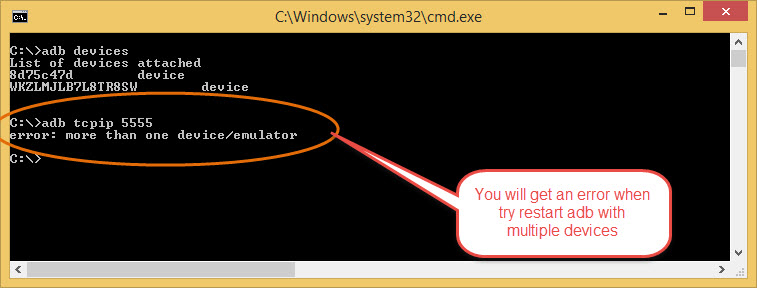
Say suppose, Now we want our *Appium tests to execute parallely on multiple devices over Wifi*. Is that Possible?? If so, how to do that?????.

Let us now connect multiple Android Devices over Wifi and execute our tests.

### How to execute Appium Tests On Real Android Device over Wifi with Multiple Devices?

We may need to follow the same steps as above, with a simple change, Let us do that in a simple way, as we have explained them in above steps.

If you try to restart adb with multiple devices connected to your PC at a time, it will throw an error as **"error: more than one device/emulator"**. To connect multiple devices over wifi, Follow the below steps :-



**Step 1** :- Connect your device ONE to PC  
**Step 2** :- "adb tcpip "  
**Step 3** :- Disconnect the device from the PC

**Step 4** :- Now Connect your device TWO to PC  
**Step 5** :- "adb tcpip "  
**Step 6** :- Disconnect the device from the PC

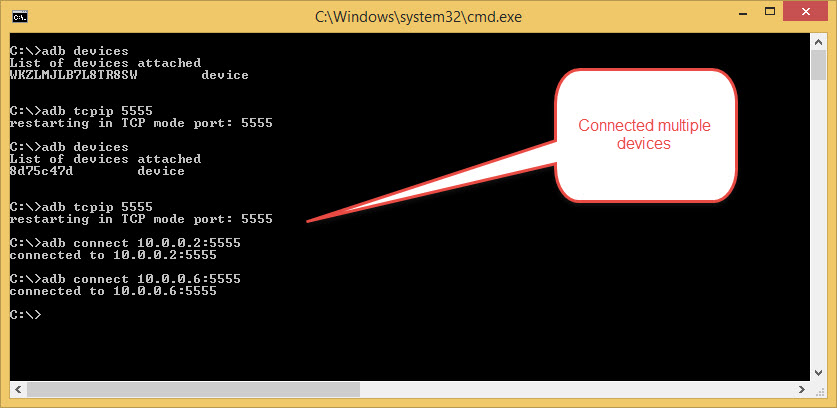
**Step 7** :- Get the IP addresses of the two devices [explained ways to get IP address above], and run below command to connect adb to your device over Wi-Fi using IP address of your device:

adb **connect** deviceONE\_IP\_Address

adb **connect** deviceTWO\_IP\_Address

Example : -  
adb connect 10.0.0.2 // IP address of device ONE  
adb connect 10.0.0.6 // IP address of device Two

In the below screen shot, "adb connect 10.0.0.2:5555" is IP address of one device and "adb connect 10.0.0.6:5555" which is the IP address of the other android device and 5555 is the same port number for two devices.



Be sure to replace 10.0.0.2 / 10.0.0.6 with the actual IP address that is assigned to your device.

Remember, Now again when we want to *switch to USB mode*, we have to tell the ADB daemon return to listening over USB. Enter below command:

#To Set back the port to USB, so the next time when ADB is started, it will start on USB again.

adb usb

After executing the above command, it should display message as 'restarting in USB mode'

When you connect your device over USB, Sometimes ADB loses connection to the device, even if you have everything working i.e USB driver installed, Developer settings enabled on your device. It suddenly stops working sometimes especially when the USB has been disconnected and reconnected and you may not be able to see your device listed when you run command 'adb devices'.

Try the following steps and it should bring your device back.

adb kill-server

adb **start**-server

adb devices

**adb start-server**- Which will ensure that there is a server running  
**adb kill-server**- This command Will kill the server if it is running

Hope this article helps you to connect your android devices / multiple devices over Wifi. Please let us know if you face any issues.