**Mobile Testing:**

In mobile testing automation framework will be remain same only few android actions will change like **long press, swipe** etc.

Other than that all selenium methods will remain same like **close (), quite(), getText(), sendKeys(), clear()** etc.

**What is Emulator?**

An emulator is a software program that allows your mobile to imitate the features of another computer or mobile software you want them to imitate by installing them to your computer or Mobile. (**Virtual device**)

**-uiautomator2options**

The way to start a session using the UiAutomator2 driver is to **include the automation Name capability in your new session request, with the value UiAutomator2** .

UI Automator is a UI testing framework introduced by google to facilitate automation on a android device or emulator.

**-Below installations setup is need to be done for mobile testing using Appium.**

1. Appium server installation (https://github.com/appium/appium-desktop/releases)  
2. Android Studio (https://developer.android.com/studio)  
3. Appium Inspector  
4.Set Environment variable Path (JAVA HOME, Android HOME, NODE HOME, another SDK path)  
5. Install Node JS  
6.Eclipse  
7. Java home

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**-Post installation set setup**

1.Configration of Emulator (Virtual Device)

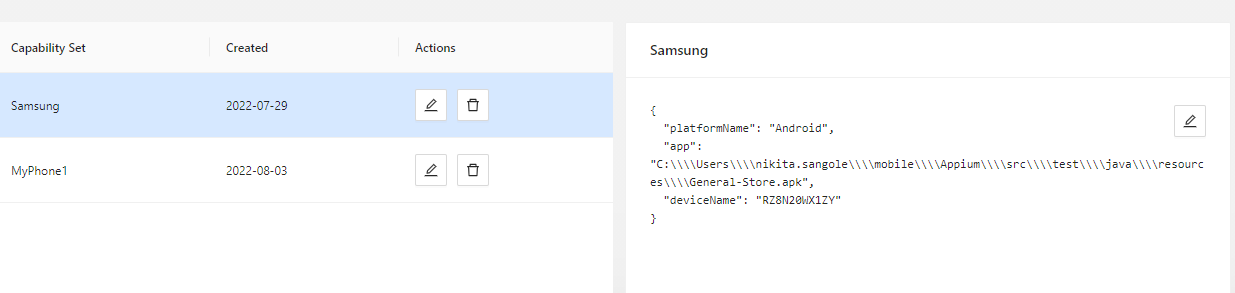
-How to create virtual device in Emulator and how to turn it on to connect it to inspector to locate the element.

2.Setup for Real Device

-How to enable developer option in real android device to do the automation

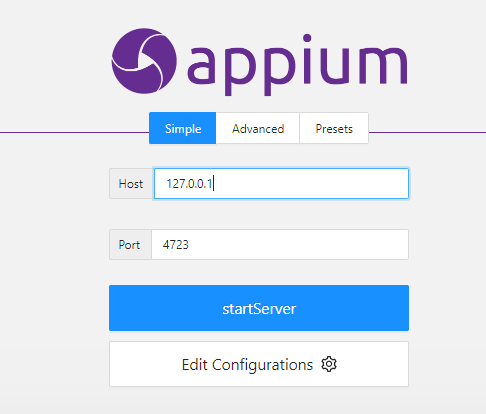
3.Set up device capabilities in inspector.

-set up capabilities in Appium inspector to connect device.

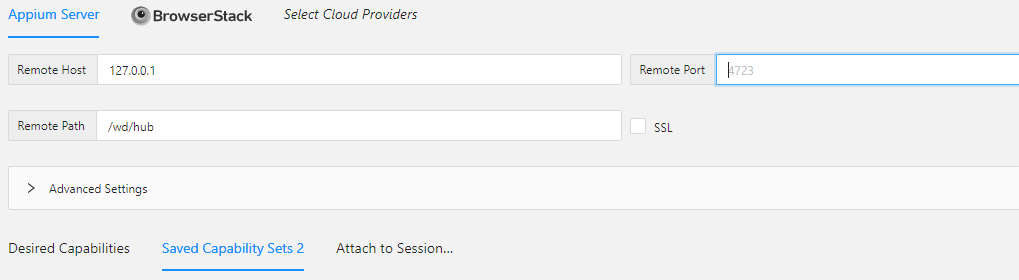


**How to identify element in Appium inspector?**

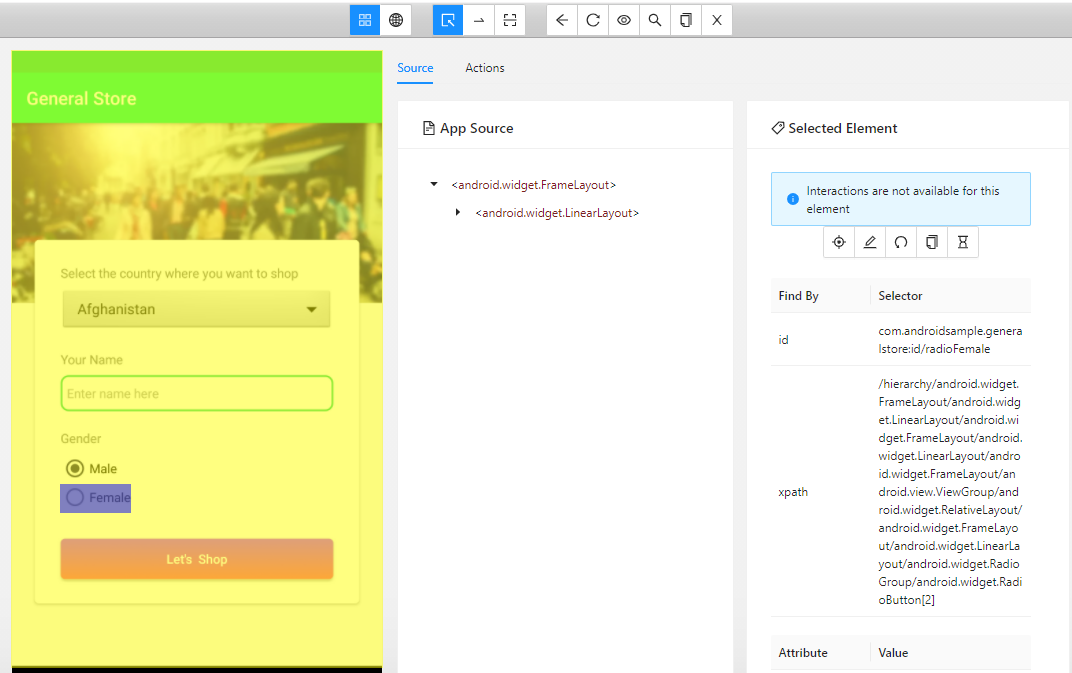
First turn Appium server on at IP 127.0.0.1 and port 4327 and click start Server.



Do below settings in Appium inspector (set remote Host and Remote path)



Add capabilities and then connect device to inspector and open the app in device on which you want to perform automation and click start session on inspector.



**-How to start Appium server Programmatically?**

* To start Appium server using script we have use **Appium service builder class**.

AppiumServiceBuilder Service= new ServiceBuilder().withAppiumjs(“path”).withIPAddress(“pass IP address”).usingPort(4723).build();

Service.Start();

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Locaters:**

-There are two additional locater are available in mobile testing **accessibilityID** and **AndroidUIAutomator**.

Xpath, ID, accessibilityID, classname, AndroidUIAutomator

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

driver.findElement(AppiumBy.*accessibilityId*("")); // in Appium to use accessibility ID and AndroidAutomator locator instead of only “BY” use “**AppiumBy**”.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Basic code to launch App in mobile**

We set up capabilities in mobile testing to tell Appium server on which device we want to perform the action.

Below are some capabilities:

* **Automation Name**
* **Platform version**
* **Device name**
* **Automation name etc**

**public** **class** appiumBasics

{

@Test

**public** **void** AppiumTest() **throws** MalformedURLException, InterruptedException

{

DesiredCapabilities obj=**new** DesiredCapabilities();

obj.setCapability(MobileCapabilityType.***AUTOMATION\_NAME***, "UiAutomator2");

obj.setCapability(MobileCapabilityType.***PLATFORM\_NAME***, "ANDROID");

obj.setCapability(MobileCapabilityType.***PLATFORM\_VERSION***, "11");

obj.setCapability(MobileCapabilityType.***DEVICE\_NAME***, "RZ8N20WX1ZY");

obj.setCapability(MobileCapabilityType.***APP***, "C:\\Users\\nikita.sangole\\mobile\\Appium\\src\\test\\java\\resources\\ApiDemos-debug.apk");

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

Thread.*sleep*(1000);

AndroidDriver driver=**new** AndroidDriver(url,obj);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Assertions are same as of same we used in selenium as we are using TestNg here.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### mobile: longPressGesture

### -Link to refer documentation:

### https://appium.io/docs/en/writing-running-appium/android/android-mobile-gestures/

### Java:

((JavascriptExecutor) driver).executeScript("mobile: longPressGesture", ImmutableMap.of(

"elementId", ((RemoteWebElement) element).getId()

));

### Python:

driver.execute\_script('mobile: longClickGesture', {'x': 100, 'y': 100, 'duration': 1000})

### Scrolling in Appium:

This gesture performs scroll gesture on the given element/area. Available since Appium v1.19

### Known till what you have to scroll

driver.findElement(AppiumBy.androidUIAutomator("new UiScrollable(new UiSelector()).scrollIntoView(text(\"WebView\"));"));

### Scrolling (no prior idea till where you have to scroll)

### boolean canScrollMore = (Boolean) ((JavascriptExecutor) driver).executeScript("mobile: scrollGesture", ImmutableMap.of( "left", 100, "top", 100, "width", 200, "height", 200, "direction", "down", "percent", 3.0 ));

**Swipe in Appium**

((JavascriptExecutor) driver).executeScript("mobile: swipeGesture", ImmutableMap.*of*(

"elementId",((RemoteWebElement)webelement).getId(),

"direction", left,

"percent", 0.75

### mobile: dragGesture

### ((JavascriptExecutor) driver).executeScript("mobile: dragGesture", ImmutableMap.of( "elementId", ((RemoteWebElement) element).getId(), "endX", 100, "endY", 100 ));

### How To roate Mobile :

DeviceRotation Landscap=**new** DeviceRotation(0, 0, 90);

### driver.rotate(Landscap);

* **How to perform Enter, back and click home button on mobile:**

driver.pressKey(**new** KeyEvent(AndroidKey.***HOME***));

driver.pressKey(**new** KeyEvent(AndroidKey.***BACK***));

driver.pressKey(**new** KeyEvent(AndroidKey.***ENTER***));

create object of KeyEvent class

**Gesture Documentation Link:**

<https://appium.io/docs/en/writing-running-appium/android/android-mobile-gestures/>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Native android app(which can be use only after installation)
* Hybrid android App (combination of native and web)

If any you wants check whether app is in native mode or web mode we use

**Driver.getContextHandles();** to check whether app is in native mode or web mode

**Framework**:

When you create object, constructer get executed automatically.

**How to Start application using app package and Activity name:**



**public** **void** preSetup()

{

//screen to home page

Activity activity = **new** Activity("com.androidsample.generalstore", "com.androidsample.generalstore.MainActivity");

driver.startActivity(activity);

}

**CMD:** adb shell dumpsys window | find "mCurrentFocus"

**appPackage**:

appPackage is the technical Java name of the application which is provided by its developers. It is a top-level package where all the code for the application resides.

**For Example**, appPackage name for ‘YouTube’ – Android is ‘com.google.android.youtube’. Similarly for WhatsApp, the appPackage is – ‘com.whatsapp’. So if you want to launch Whatsapp from Appium, it is required to provide its appPackage name as ‘com.whatsapp’ in Appium.

**appActivity**:

appActivity refers to the different functionalities that are performed by the app.

**For Example**, WhatsApp provides multiple functionalities such as conversations, video calls, profile information, setting profile photos, changing status, setting notifications, etc. All these functionalities are performed by different appActivity. Together with these activities, every application has a main activity which is the main screen you see when you launch the app.

**For Example**, For WhatsApp, the main screen is it is the Chats window, and for Facebook, the main screen is the Wall where we see posts.

Driver.getContextHandles(); to check whether app is in native mode or web mode

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name="Suite">

<test thread-count="5" name="Test">

<classes>

<class name="com.MobileProject.eCommrce\_TC\_2"/>

<class name="com.MobileProject.eCommrce\_TC\_4"/>

</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->

