**Android Automation:**

* Enable Developer options on the device.
* Connect the device to the machine using USB so the USB driver is installed.
* To check if the device is connected, open cmd and type adb devices. This displays list of attached devices.
* Download Android SDK – Unzip to local machine.
* Under Man Android SDK folder, go to C:\Program Files (x86)\Android\android-sdk\tools
* There will be a bat file called uiautomatorviewer. Double click the bat file to launch the UI automator.
* When the automator viewer is open, click on device screenshot to get the device.
* Every time the device screens changes, you got to click on the device screenshot.

**Appium**:

* Appium is an extended version of Selenium, cross-platform test automaton tool for native(mobile generic app only mobile view), hybrid (can switch to mobile view app or web view) and mobile web apps (eg: app that opens in chrome)
* Webdriver + mobile features = Appium
* Appium follows client-server architecture. When we code Appium, they communicate to the Appium server. So we need to first start the Appium server before even running the code.
* Appium server uses Json wire protocol so it reaches the Appium server in a http session object (Called desired concepts)which is headless which means independent of any language.
* Appium server will act as a proxy between the mobile you are testing and the client library.
* Appium can support Android (UI Automator)and iOS (iOS instrumentation framework)

**Appium pre-requisites: System environment variables**

* Install Android SDK. Unzip it. This has an editor to write an Appium code called eclipse.
* Set the system environmental variable from sdk folder to the path , go to the end put a semicolon and copy paste the path.
* Set the system variable from platform tools
* Set the System variable from tools folder

**Install Appium Server:**

* Download appium. Unzip it. Go to the unzipped and click on the Appium exe. This will open the Appium server.
* We need to get Appium and selenium jars
* Search for Appium jars above 2.1.0 and go to Mave n repository link.
* Import Selenium and Appium jars onto Eclipse
* Download latest selenium Jar file which looks like selenium-2.44.0, Import the 2 files within and also all the jars in the libs

**Desired Capabilities:**

* DesiredCapabilities are options that you can use to customise and configure a session with your device.
* DesiredCapabilities takes our inputs and talks to the Appium server to create a session.
* Syntax:

DesiredCapabilities capabilities = **new** DesiredCapabilities(); //Create an object for the class DesiredCapabilities

capabilities.setCapability(MobileCapabilityType.Browser\_Name, ""); //Setcapability method will start defining arguments related to the session

* Following url has the desired capabilities: <https://github.com/appium/appium/blob/master/docs/en/writing-running-appium/caps.md>

| **Capability** | **Description** | **Values** |
| --- | --- | --- |
| automationName | Which automation engine to use | Appium (default) or Selendroid |
| platformName | Which mobile OS platform to use | iOS, Android, or FirefoxOS |
| platformVersion | Mobile OS version | e.g., 7.1, 4.4 |
| deviceName | The kind of mobile device or emulator to use | iPhone Simulator, iPad Simulator,iPhone Retina 4-inch, Android Emulator, Galaxy S4, etc.... On iOS, this should be one of the valid devices returned by instruments withinstruments -s devices. On Android this capability is currently ignored. |
| app | The absolute local path *or* remote http URL to an.ipa or .apk file, or a .zip containing one of these. Appium will attempt to install this app binary on the appropriate device first. Note that this capability is not required for Android if you specifyappPackage and appActivity capabilities (see below). Incompatible with browserName. | /abs/path/to/my.apk orhttp://myapp.com/app.ipa |
| browserName | Name of mobile web browser to automate. Should be an empty string if automating an app instead. | 'Safari' for iOS and 'Chrome', 'Chromium', or 'Browser' for Android |
| newCommandTimeout | How long (in seconds) Appium will wait for a new command from the client before assuming the client quit and ending the session (wait for element) | e.g. 60 |
| autoLaunch | Whether to have Appium install and launch the app automatically. Default true | true, false |
| language | (Sim/Emu-only) Language to set for the simulator / emulator | e.g. fr |
| locale | (Sim/Emu-only) Locale to set for the simulator / emulator | e.g. fr\_CA |
| udid | Unique device identifier of the connected physical device | e.g. 1ae203187fc012g |
| orientation | (Sim/Emu-only) start in a certain orientation |  |

* The following are necessary to automate a native app on Android or iOS

AutomationName

platformName

platformVersison

deviceName

app

@Test

**public** **void** test() {

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

cap.setCapability(MobileCapabilityType.*PLATFORM\_NAME*, MobilePlatform.*ANDROID*); //you set capability key to the mobile type iOS or Android,

}

}

@Test

**public** **void** test() {

File appDir = **new** File("src"); //method to define a file

File app = **new** File(appDir, "EPiCSSActivity15\_RC2.apk");

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

cap.setCapability(MobileCapabilityType.*PLATFORM\_NAME*, MobilePlatform.*ANDROID*); //you set capability key to the mobile type iOS or Android,

cap.setCapability(MobileCapabilityType.*DEVICE\_NAME*, "Android Device"); //Android device device stands for real device

cap.setCapability(MobileCapabilityType.*APP*, app.getAbsolutePath()); //If you are automating App, the app must be present on the local machine or in the Eclipse. When we write this script, this will automatically open the app on the device.

}

///////// The following 5 steps stay common if automating the apps.

@Test

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

File appDir = **new** File("src"); //method to define a file

File app = **new** File(appDir, "EPiCSSActivity15\_RC2.apk");

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

cap.setCapability(MobileCapabilityType.*PLATFORM\_NAME*, MobilePlatform.*ANDROID*); //you set capability key to the mobile type iOS or Android,

cap.setCapability(MobileCapabilityType.*DEVICE\_NAME*, "Android Device"); //Android device device stands for real device

cap.setCapability(MobileCapabilityType.*APP*, app.getAbsolutePath()); //If you are automating App, the app must be present on the local machine or in the Eclipse. When we write this script, this will automatically open the app

//driver to run the scripts for android if Android OS and iOS if iOS.

//Create an object for the Android driver

AndroidDriver driver = **new** AndroidDriver(**new** URL("http://127.0.0.1:4723/wd/hub"),cap) ; // we need to pass 2 args, in what port the Appium server is configured and where it is hosted and the cap arg. The secon arg cap will handshake with the appium server and then it will provide cap info to the server

}

}

* //invoke the epic app . It is a must to start the appium server before we invoke the app
* To start the Appium server, go to the location where appium was, open the appium.exe and click on run
* Browser name : web application for a browser application. If using native app, it must be blank.
* App is the path where the apk file exists in the system. We need to point APK file to the script to install on the device.

**REAL DEVICES**

Connect the device

Cd to C:\Priya\LatestAndroidSDK\platform-tools and type adb devices.

That would bring up the devices attached.

To install apk on the device, this apk needs to be re-signed bcoz the apk has the dev’s signature who sent you, so if we re-sign to get our signature and then test it.

Re-signing APK:

* Signature of both AUT and Test project must match. Re-signing is necessary
* Unsign.apk
* Re-sign it with debug.keystore

Command: jarsigner –verbose – sigalg MD5with RSA – digestalg SHA1 –keystore.

**On a Mac:**

Apps Signing:

To test the Apps, the apps have to be signed by my debug key. Got o Eclipse -> Preferences -> Android - > Build - copy the debug keystore. Use this to sign the App that will be tested with the debug keystore.

Test ADB connections on a MAC: Make sure you copy the adb file from the platform tools – and copy to the root of the Android sdk folder and access to that location to run the adb command.

Silverbacks-iMac:~ silverback$ cd ~/Android-sdks

Silverbacks-iMac:Android-sdks silverback$ ./adb devices

List of devices attached

4178be5e device

The final code that installed the App for me

**public** **class** test2 {

@Test

**public** **void** apptest1() **throws** Exception{

File appDir = **new** File("src"); //method to define a file

File app = **new** File(appDir, "EPiCSSActivity15\_RC2.apk");

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

cap.setCapability(MobileCapabilityType.*PLATFORM\_NAME*, MobilePlatform.*ANDROID*); //you set capability key to the mobile type iOS or Android,

cap.setCapability(MobileCapabilityType.*DEVICE\_NAME*, "Android Device"); //Android device device stands for real device

cap.setCapability(MobileCapabilityType.*APP*, app.getAbsolutePath()); //If you are automating App, the app must be present on the local machine or in the Eclipse. When we write this script, this will automatically open the app on the device.

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

}

}

To find the elements on Android:

<http://appium.github.io/java-client/>

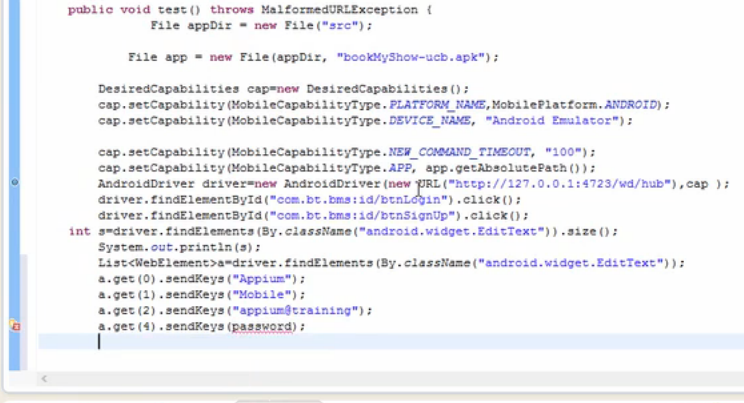
UI automator on the device.

go to C:\Priya\ADT\adt-bundle-windows-x86\_64-20140702\sdk\tools

bring up the UI automator.bat file. This takes a screenshot of the real device.

Resource id is id

If there are more paths with same class name, find by index.



Send keys to the device

@Test

**public** **void** apptest1() **throws** Exception{

File appDir = **new** File("src"); //method to define a file

File app = **new** File(appDir, "EPiCSSActivity15\_RC2.apk");

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

cap.setCapability(MobileCapabilityType.*PLATFORM\_NAME*, MobilePlatform.*ANDROID*); //you set capability key to the mobile type iOS or Android,

cap.setCapability(MobileCapabilityType.*DEVICE\_NAME*, "Android Device"); //Android device device stands for real device

cap.setCapability(MobileCapabilityType.*NEW\_COMMAND\_TIMEOUT*, "100");

cap.setCapability(MobileCapabilityType.*APP*, app.getAbsolutePath()); //If you are automating App, the app must be present on the local machine or in the Eclipse. When we write this script, this will automatically open the app on the device.

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

driver.findElement(By.*id*("com.silverbackmdm.epic.companion.ss:id/sbUsernameLabel")).sendKeys("Priya");

driver.findElementById("com.silverbackmdm.epic.companion.ss:id/tableRow1").click();

driver.findElement(By.*id*("com.silverbackmdm.epic.companion.ss:id/sbOtp")).clear();

driver.findElement(By.*id*("com.silverbackmdm.epic.companion.ss:id/sbOtp")).sendKeys("1587");

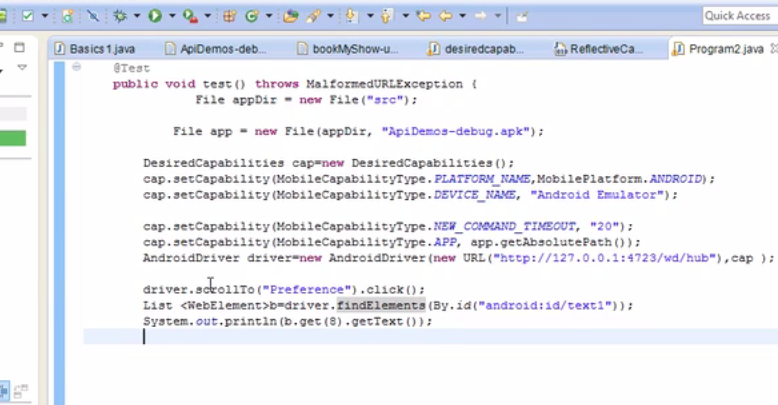
driver.findElementById("com.silverbackmdm.epic.companion.ss:id/tableRow1").click();

driver.findElementByClassName("android.widget.Button").click();

}

}

* **Driver.scrollto** is if you have to scroll down to options

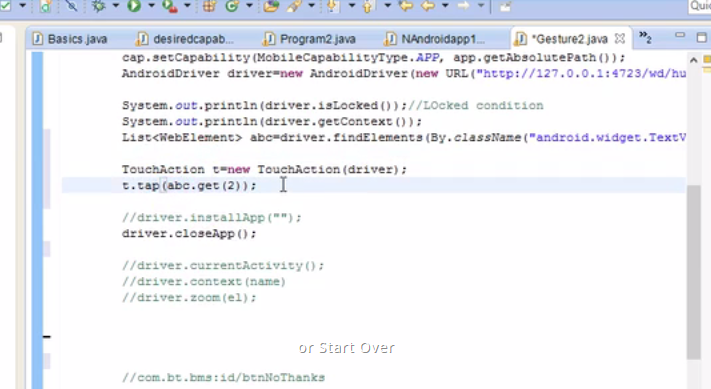


* **Zoom: Swipe**



* **Tap on a APP:**

Take the class name and tap at the index



* Locked/Unlocked: if device is in locked status

Driver.isLocked(); if it returns false means the device is not locked.

* Driver.getContext(); //whenever you open an App whether a mobile view or a web view. If it’s a hybrid app you can switch to web view or a mobile view example seek. Get context is used whenever the app is opened whether it opened in web context or mobile context

Syste,.out.println(Driver.getContext()); // prints the app mode.

* Driver. installAPp(“”); // installs App
* Driver.closeApp();
* Driver.currentActivity();

In an app there are multiple activities per page. Or whatever url is

* Touch Actions end with perform();

**Appium API’s for UI interaction:**

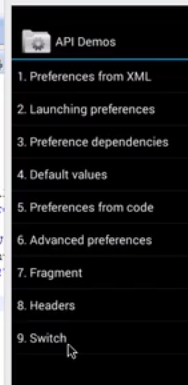
* To hide keyboard:

driver.hideKeyboard();

* To scroll down on App: (Appium specific)

Driver.ScrollTo(“Whatver”).click(); - This will scroll down to whatever and click.

* **To get the text from a list of elements: sample pic below. I want to get the position 9**



In this case , the class and is same. So achive it by index.

List <WebElement> priya = Driver.findelements(By.id(“whtver”);

System.out.printkn(priya.get(8).getText()

BROWSER:

Identifying elements on mobile browsers:

**User Agent:** Whenever you hit a url in a browser, there are some user agents set for the browser. If it’s a web browser user agent, the UI is displayed as web browser and if it’s a mobile browser user agent. So it will see what platform it is, Whether a web or a mobile platform. So we can set a platform number.

* Change the user agent of a browser to mobile to the user agent is rendered as a mobile UI
* Like firebug to identify objects on a normal browser, we cant use, so we trigger the browser to a mobile view and find the elements ☺
* So change to a mobile site and inspect with a firebug
* To switch from web browser to mobile browser, install the add on called “User agent switcher”
* Go to add ons – user agent, options it only has internet explorer
* Go to the link <http://techpatterns.com/forums/about304.html> and download xml user agent list. This has different browser and different OS.
* Go to add ons, import that xml file. It will download allOS

**Hybrid Apps:**

* To use hybrid apps, you need to switch context to native\_apps

dDriver.context("NATIVE\_APP");

**Issues**:

* When entering username and otp into the fields onto the web browser, the keyboard is not hidden and hence incorrect inputs were sent.

Fix: Check the behaviour and use the keys according

dDriver.getKeyboard().sendKeys(Keys.*ENTER*);

dDriver.getKeyboard().sendKeys(Keys.*TAB*);

* When switching from web browser to the system app, the elements were not recognized, hence I used switch context dDriver.context("NATIVE\_APP");

And used touch actions

dDriver.context("NATIVE\_APP");

TouchAction touch = **new** TouchAction(dDriver);

Thread.*sleep*(3000);

touch.tap(dDriver.findElementByClassName("android.widget.Button")).perform();//Tap on Install Companion

touch.tap(dDriver.findElementById("com.android.vending:id/continue\_button\_label")).perform();//Tap on Accept.

* Entering elements onto a webApp, elements were not recognised. So use mobile elements

MobileElement Username= (MobileElement)dDriver.findElementById("com.silverbackmdm.epic.companion.ss:id/sbUsername");

Username.sendKeys("sbtest1@airloom.com.au");

* Sending inputs to an app which has multiple edit fields were going only to a single edit box also use mobile elements.

Below is the whole code that works to enrol a device from ssp

package AndroidDevice;

import io.appium.java\_client.MobileElement;

import io.appium.java\_client.TouchAction;

import io.appium.java\_client.android.AndroidDriver;

import io.appium.java\_client.remote.MobileCapabilityType;

import io.appium.java\_client.remote.MobilePlatform;

import java.net.URL;

import org.junit.Test;

import org.openqa.selenium.By;

import org.openqa.selenium.Keys;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.remote.DesiredCapabilities;

import Utilities.xlReadWrite;

public class Enrollment {

@Test

public void EnrollAndroidfromssp() throws Exception{

DesiredCapabilities cap = new DesiredCapabilities();

cap.setCapability(MobileCapabilityType.PLATFORM\_NAME, MobilePlatform.ANDROID);

cap.setCapability(MobileCapabilityType.DEVICE\_NAME, "Android Device");

cap.setCapability(MobileCapabilityType.DEVICE\_NAME, "Galaxy S4");

cap.setCapability(MobileCapabilityType.BROWSER\_NAME, "Chrome");

cap.setCapability(MobileCapabilityType.NEW\_COMMAND\_TIMEOUT, "200");

cap.setCapability(MobileCapabilityType.APP\_PACKAGE, "com.android.vending");

cap.setCapability(MobileCapabilityType.APP\_ACTIVITY, "com.android.vending.AssetBrowserActivity");

cap.setCapability(MobileCapabilityType.APP\_PACKAGE, "com.silverbackmdm.epic.companion.ss");

cap.setCapability(MobileCapabilityType.APP\_ACTIVITY, "com.silverbackmdm.epic.companion.ss.EpicSSActivity");

AndroidDriver dDriver = new AndroidDriver(new URL("http://127.0.0.1:4723/wd/hub"),cap) ;

String[][] xTD, xTC; //2 dimensional variable to read/Store the test data results

String xlPath, xlResPath; //ExcelSpreadsheet path

String vError, vResult, vScreenshot, vTempRes;//Declaring the error columns on the excel

int iWait;

WebDriver wDriver = new FirefoxDriver();

xlPath = "C:\\Priya\\All Automation Tests class files\\AutomationTestData\\DeviceTests\\Tags\\Profile\\Passcode\\Passcode.xls";

xlResPath = "C:\\Priya\\All Automation Tests class files\\AutomationTestData\\DeviceTests\\Tags\\Profile\\Passcode\\Results";

//Read the entire TC

xTC = xlReadWrite.xlRead(xlPath, "TestCases");

xTD = xlReadWrite.xlRead(xlPath, "TestData");

for (int i = 1; i<xTC.length; i++) { //Go to each row in TC

if (xTC[i][3].equalsIgnoreCase("Y")) { //go to every test step if it is y

xTC[i][4] = "Pass"; //TC is pass by default on the excel, it updates the excel if the test case/step fails

for (int k=1; k<xTD.length; k++){

if (xTD[k][4].equalsIgnoreCase("y")){

vError = "-";

vResult = "Pass";

vScreenshot = "-";

try{

dDriver.get("https://qa.silverbackmdm.com/ssp");

dDriver.findElementById("Username").sendKeys("sbtest1@airloom.com.au");

dDriver.findElementById("Password").sendKeys("S1lverb@ck");

Thread.sleep(3000);

dDriver.getKeyboard().sendKeys(Keys.ENTER);

dDriver.getKeyboard().sendKeys(Keys.TAB);

String OTP = dDriver.findElement(By.cssSelector("fieldset > div > div > div")).getText();

System.out.println("OTP is" + OTP);

dDriver.findElement(By.linkText("https://qa.silverbackmdm.com/activate")).click();

Thread.sleep(3000);

dDriver.context("NATIVE\_APP");

TouchAction touch = new TouchAction(dDriver);

Thread.sleep(3000);

touch.tap(dDriver.findElementByClassName("android.widget.Button")).perform();//Tap on Install Companion

touch.tap(dDriver.findElementById("com.android.vending:id/continue\_button\_label")).perform();//Tap on Accept.

Thread.sleep(9000l);

Thread.sleep(9000l);

Thread.sleep(9000l);

dDriver.findElementByAndroidUIAutomator("new UiSelector().text(\"OPEN\")").click();

MobileElement Username= (MobileElement)dDriver.findElementById("com.silverbackmdm.epic.companion.ss:id/sbUsername");

Username.sendKeys("sbtest1@airloom.com.au");

dDriver.hideKeyboard();

MobileElement Password = (MobileElement)dDriver.findElementById("com.silverbackmdm.epic.companion.ss:id/sbOtp");

Password.sendKeys(OTP);

dDriver.hideKeyboard();

dDriver.findElementById("com.silverbackmdm.epic.companion.ss:id/sbActivateButton").click();

Thread.sleep(2000);

dDriver.findElementById("com.android.settings:id/action\_button").click();

//driver.quit();

String tagName;

tagName = "AutomatedTagAndroid";

//Step1: Log in into Console as an Administrator,

wDriver.get( "https://qa.silverbackmdm.com/admin");

wDriver.findElement(By.id("Username")).sendKeys("admin");

wDriver.findElement(By.id("Password")).sendKeys("S1lverb@ck");

wDriver.findElement(By.xpath("//input[@type='submit']")).click();

//Thread.sleep(iWait);

//Step2:Create an ADMIN Tag

wDriver.findElement(By.xpath("//a[@href='/admin/Tags']")).click();

wDriver.findElement(By.xpath("//input[@type='button']")).click();

wDriver.findElement(By.xpath("//input[@id='Name']")).sendKeys(tagName);

wDriver.findElement(By.xpath("//textarea[@id='Description']")).sendKeys("Test");

//Thread.sleep(iWait);

} catch (Exception mye){ //Unexpected error from the Keyword Executor method

vResult = "Fail";

vError = "Error is " + mye;

//vScreenshot = "xlResPath" + xTS[j][0] + "-" + xTS[j][1] + ".png";

//File scrFile = ((TakesScreenshot)myD).getScreenshotAs(OutputType.FILE); //code to capture screenshot

//FileUtils.copyFile(scrFile, new File(vScreenshot));

System.out.println("Step failed:" + vError);

xTC[i][4] = "Fail";

xTC[i][5] = vResult;

xTC[i][6] = vError;

//xTC[i][7] = vScreenshot;

} }}

//Write back the test Steps with the error

xlReadWrite.writeXL(xlResPath + "/TestCase-"+xTC[i][0]+".xls", "TestCase Results", xTC );

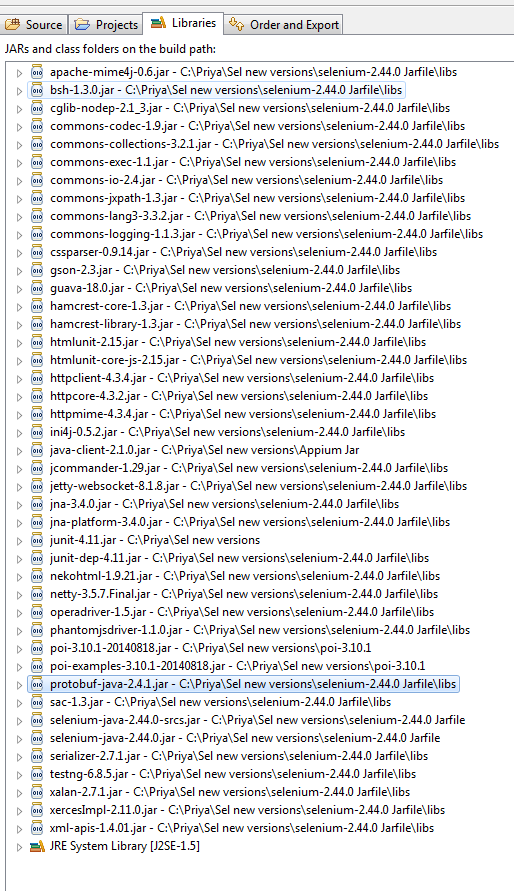
}

}

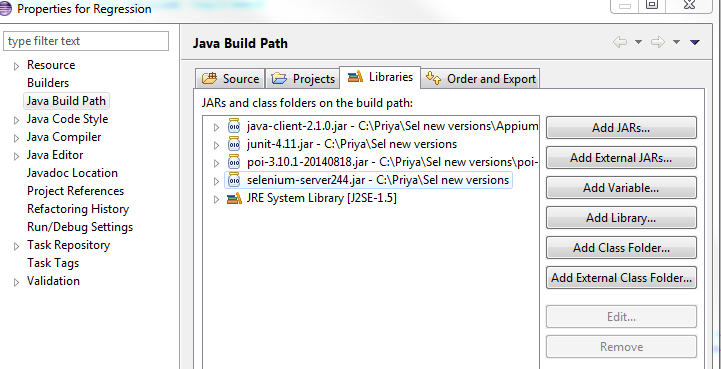
}

}

JAR files:



Browser:



**WAIT FOR THE ELEMENT TO BE VISBILE and click(example companion scan button)**

MobileElement companionScan= (MobileElement)dDriver.findElementById("com.silverbackmdm.epic.companion.ss:id/btn\_scan");

**while** (!(companionScan).isDisplayed()) ;

System.*out*.println("It's visible!");

WebDriverWait wait = **new** WebDriverWait(dDriver, 30);

//wait.until(ExpectedConditions.elementToBeClickable(By.name("somename")));

wait.until(ExpectedConditions.*elementToBeClickable*(By.*id*("com.silverbackmdm.epic.companion.ss:id/btn\_scan")));

*companionScan.click();*

Until a button is visible:

MobileElement companionScan= (MobileElement)dDriver.findElementById("com.silverbackmdm.epic.companion.ss:id/btn\_scan");

**while** (!(companionScan).isDisplayed()) ;

System.*out*.println("Companion Scan button is visible!");

To know all the app packages on the devices

Adb devices

Adb shell pm list packages // gives the list of all packages

//to get path

Adb shell pm path net.whatever ends with the package name after com.

Pull this app from the device to local machine

Adb pull /path it gave above c:\apps

Capture onscreen native app view : (the native settings are also considered as app, need to specify an app package”)

DesiredCapabilities cap2 = **new** DesiredCapabilities();

cap2.setCapability(MobileCapabilityType.*PLATFORM\_NAME*, MobilePlatform.*ANDROID*);

cap2.setCapability(MobileCapabilityType.*DEVICE\_NAME*, "Android Device");

cap2.setCapability(MobileCapabilityType.*DEVICE\_NAME*, "Galaxy S4");

cap2.setCapability(MobileCapabilityType.*DEVICE\_NAME*, "Galaxy S5");

cap2.setCapability(MobileCapabilityType.*NEW\_COMMAND\_TIMEOUT*, "200");

AndroidDriver dDriver2 = **new** AndroidDriver(**new** URL("http://127.0.0.1:4723/wd/hub"),cap2) ;

dDriver2.sendKeyEvent(AndroidKeyCode.*HOME*);

Thread.*sleep*(9000l);

System.*out*.println(dDriver2.getContext());

dDriver2.context("NATIVE\_APP");

dDriver2.findElementByAndroidUIAutomator("new UiSelector().text(\"PIN\")").click();

dDriver2.findElementById("com.android.settings:id/password\_entry").sendKeys("258");

String t1 = dDriver2.findElementByAndroidUIAutomator("new UiSelector().text(\"PIN must contain at least 4 numbers.\")").getText();

System.*out*.println(t1);

**Clear edit fields:**

String Pin = dDriver2.findElementById("com.android.settings:id/password\_entry").getText();

**int** maxChars = Pin.length();

**for** (**int** i = 0; i < maxChars; i++)

((AppiumDriver) dDriver2.findElementById("com.android.settings:id/password\_entry")).sendKeyEvent(67);

//TouchAction touch = new TouchAction(dDriver2);

Thread.*sleep*(3000);

//touch.tap(dDriver2.findElementByClassName("android.widget.Button")).longPress(Pin);

dDriver2.sendKeyEvent(AndroidKeyCode.*DEL*);

**Reset APP from code:**

@Test

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

File appDir = **new** File("src"); //method to define a file

File app = **new** File(appDir, "PriyaEPiCSSActivity.apk");

DesiredCapabilities cap2 = **new** DesiredCapabilities();

cap2.setCapability(MobileCapabilityType.*PLATFORM\_NAME*, MobilePlatform.*ANDROID*);

cap2.setCapability(MobileCapabilityType.*DEVICE\_NAME*, "Android Device");

cap2.setCapability(MobileCapabilityType.*DEVICE\_NAME*, "Galaxy S4");

cap2.setCapability(MobileCapabilityType.*DEVICE\_NAME*, "Galaxy S5");

cap2.setCapability(MobileCapabilityType.*NEW\_COMMAND\_TIMEOUT*, "200");

cap2.setCapability("app", app);

//capabilities.setCapability("appActivity", "com.android.settings.Settings");

cap2.setCapability("noReset", **true** );

//cap2.setCapability("noReset","com.silverbackmdm.epic.companion.ss.EpicSSActivity");

cap2.setCapability(MobileCapabilityType.*APP*, app.getAbsolutePath());

AndroidDriver dDriver2 = **new** AndroidDriver(**new** URL("http://127.0.0.1:4723/wd/hub"),cap2) ;

dDriver2.sendKeyEvent(AndroidKeyCode.*HOME*);

Thread.*sleep*(9000l);

System.*out*.println(dDriver2.getContext());

**OPENS SETTINGS:**

**public** **class** openSettings {

@Test

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

DesiredCapabilities cap2 = **new** DesiredCapabilities();

cap2.setCapability(MobileCapabilityType.*PLATFORM\_NAME*, MobilePlatform.*ANDROID*);

cap2.setCapability(MobileCapabilityType.*DEVICE\_NAME*, "Android Device");

cap2.setCapability(MobileCapabilityType.*DEVICE\_NAME*, "Galaxy S4");

cap2.setCapability(MobileCapabilityType.*DEVICE\_NAME*, "Galaxy S5");

cap2.setCapability(MobileCapabilityType.*NEW\_COMMAND\_TIMEOUT*, "200");

cap2.setCapability(MobileCapabilityType.*APP\_PACKAGE*, "com.android.settings");

cap2.setCapability(MobileCapabilityType.*APP\_ACTIVITY*, ".Settings");

//capabilities.setCapability("appActivity", "com.android.settings.Settings");

cap2.setCapability("noReset", "com.silverbackmdm.epic.companion.ss");

AndroidDriver dDriver2 = **new** AndroidDriver(**new** URL("http://127.0.0.1:4723/wd/hub"),cap2) ;

dDriver2.sendKeyEvent(AndroidKeyCode.*HOME*);

Thread.*sleep*(9000l);

System.*out*.println(dDriver2.getContext());

dDriver2.context("NATIVE\_APP");

dDriver2.launchApp();

dDriver2.sendKeyEvent(AndroidKeyCode.*SETTINGS*);

}

}