**Creating First Appium Project**

**Desired Capabilities**

- Keys and values encoded in a JSON object, sent by Appium clients (e.g. Java Client) to the server when a new driver session is requested.

- Scripted through code or through the Appium Inspector

- Carries information like,

1. How you want your test to work

2. Device to connect

3. App to work with

Types

- General Capabilities (Common for all drivers)

- Android Only

- iOS Only

Links (Desired capabilities are located under respective GitHub pages of the drivers)

UiAutomator2 (Android) capabilities:

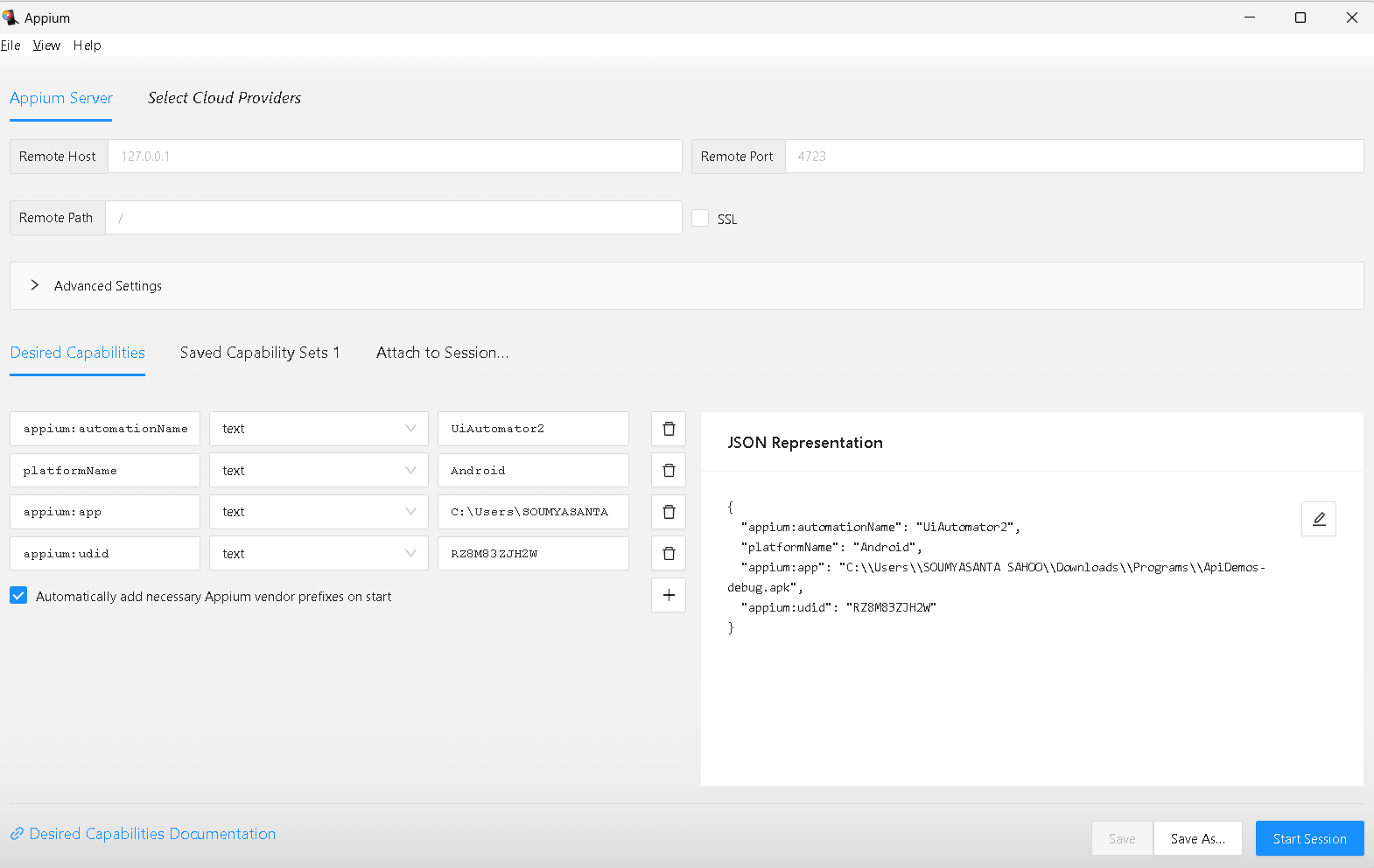
https://github.com/appium/appium-uiautomator2-driver?#capabilities

XCUITest (iOS) capabilities:

<https://github.com/appium/appium-xcuitest-driver#capabilities>

**Vendor Prefix in Appium**

* For adding Desired capabilities Appium have some Standard capabilities but when we add non-standard Desired capability then Appium adds some vendor prefix there
* In below image wherever appium: is added those are Vendor prefix.
* We can add Vendor prefix manually otherwise we have to click on the check box to add vendor prefix automatically.



Appium Vendor Prefix

**Why we use Maven Framework**

1. Dependency management

2. Test execution

3. Build Lifecycle Management (validate, compile, package, verify, install, deploy)

4. Supports CI/CD

5. Parallel Execution

**How to get appPackage & appActivity**

Android: appPackage and appActivity desired capabilities

How to find appPackage and appActivity?

Option 1

————

=> Launch app on the device and bring activity in focus

=> Launch terminal/CMD prompt and execute command (for older Android versions): adb shell dumpsys window | grep -E mCurrentFocus

=> Command for Android 10 and above: adb shell "dumpsys activity activities | grep mResumedActivity"

=> On windows, you might have to launch the adb shell first, using command “adb shell”. Then execute the remaining command: "dumpsys activity activities | grep mResumedActivity"

Option 2

————

Install APK Info app

**First Program to Open an App with Real Device**

**package** AppiumBasics;

**import** java.net.MalformedURLException;

**import** java.net.URL;

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

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

**import** org.testng.annotations.Test;

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

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

**public** **class** OpenApiDemoInRealDeviceTest {

@Test

**public** **void** openAppTest() **throws** MalformedURLException

{

DesiredCapabilities dc=**new** DesiredCapabilities();

//Common Desired Capabilities

dc.setCapability(MobileCapabilityType.***PLATFORM\_NAME***, "Android");

dc.setCapability(MobileCapabilityType.***DEVICE\_NAME***, "Galaxy M30s");

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

dc.setCapability(MobileCapabilityType.***UDID***, "RZ8M83ZJH2W");

//Desire Capabilities for Android

dc.setCapability("appPackage", "io.appium.android.apis");

dc.setCapability("appActivity", ".ApiDemos");

//Appium server URL

URL u=**new** URL("http://localhost:4723");

AndroidDriver driver=**new** AndroidDriver(u,dc);

driver.manage().timeouts().implicitlyWait (Duration.*ofSeconds*(30));

driver.quit();

}

}

**First Program to Open an App with Virtual Device (Emulator & Simulator)**

package org.appium.first;

import io.appium.java\_client.AppiumDriver;

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

import io.appium.java\_client.ios.IOSDriver;

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

import org.openqa.selenium.remote.DesiredCapabilities;

import java.io.File;

import java.net.MalformedURLException;

import java.net.URL;

public class CreateDriverSession {

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

DesiredCapabilities caps = new DesiredCapabilities();

caps.setCapability(MobileCapabilityType.PLATFORM\_NAME, "iOS");

caps.setCapability(MobileCapabilityType.DEVICE\_NAME, "iPhone 13 Pro Max");

caps.setCapability(MobileCapabilityType.AUTOMATION\_NAME, "XCUITest");

caps.setCapability(MobileCapabilityType.UDID, "FDDAF4BC-2C59-4E30-BC16-B05C16E3D29D");

String appUrl = System.getProperty("user.dir") + File.separator + "src" + File.separator + "main"

+ File.separator + "resources" + File.separator + "UIKitCatalog-iphonesimulator.app";

caps.setCapability(MobileCapabilityType.APP, appUrl);

URL url = new URL("http://0.0.0.0:4723");

AppiumDriver driver = new IOSDriver(url, caps);

/\* DesiredCapabilities caps = new DesiredCapabilities();

caps.setCapability(MobileCapabilityType.PLATFORM\_NAME, "Android");

caps.setCapability(MobileCapabilityType.DEVICE\_NAME, "Pixel\_5");

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

caps.setCapability(MobileCapabilityType.UDID, "emulator-5554");

String appUrl = System.getProperty("user.dir") + File.separator + "src" + File.separator + "main"

+ File.separator + "resources" + File.separator + "ApiDemos-debug.apk";

caps.setCapability(MobileCapabilityType.APP, appUrl);

URL url = new URL("http://0.0.0.0:4723");

AppiumDriver driver = new AndroidDriver(url, caps);\*/

}

}

**Open an App with Virtual Device (Emulator & Simulator) using Options class**

package org.appium.first;

import io.appium.java\_client.A

ppiumDriver;

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

import io.appium.java\_client.android.options.UiAutomator2Options;

import io.appium.java\_client.ios.IOSDriver;

import io.appium.java\_client.ios.options.XCUITestOptions;

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

import org.openqa.selenium.remote.DesiredCapabilities;

import java.io.File;

import java.net.MalformedURLException;

import java.net.URL;

public class CreateDriverSessionUsingOptions {

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

String appUrl = System.getProperty("user.dir") + File.separator + "src" + File.separator + "main"

+ File.separator + "resources" + File.separator + "UIKitCatalog-iphonesimulator.app";

XCUITestOptions options = new XCUITestOptions().

setDeviceName("iPhone 13 Pro Max").

setAutomationName("XCUITest").

setUdid("FDDAF4BC-2C59-4E30-BC16-B05C16E3D29D").

setApp(appUrl);

URL url = new URL("http://0.0.0.0:4723");

AppiumDriver driver = new IOSDriver(url, options);

/\* String appUrl = System.getProperty("user.dir") + File.separator + "src" + File.separator + "main"

+ File.separator + "resources" + File.separator + "ApiDemos-debug.apk";

UiAutomator2Options options = new UiAutomator2Options().

setDeviceName("Pixel\_5").

setAutomationName("UiAutomator2").

setApp(appUrl);

URL url = new URL("http://0.0.0.0:4723");

AppiumDriver driver = new AndroidDriver(url, options);\*/

}

}

**How to Launch Emulator Automatically**

Capabilities to use:

avd

avdLaunchTimeout

package org.appium.first;

import io.appium.java\_client.AppiumDriver;

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

import io.appium.java\_client.ios.IOSDriver;

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

import org.openqa.selenium.remote.DesiredCapabilities;

import java.io.File;

import java.net.MalformedURLException;

import java.net.URL;

public class CreateDriverSession {

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

DesiredCapabilities caps = new DesiredCapabilities();

caps.setCapability(MobileCapabilityType.PLATFORM\_NAME, "Android");

caps.setCapability(MobileCapabilityType.DEVICE\_NAME, "Pixel\_XL");

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

caps.setCapability(MobileCapabilityType.UDID, "emulator-5554");

caps.setCapability(capabilityName: “avd”, value: “Pixel\_XL”);

caps.setCapability(“avdLaunchTimeout”, 180000);

String appUrl = System.getProperty("user.dir") + File.separator + "src" + File.separator + "main"

+ File.separator + "resources" + File.separator + "ApiDemos-debug.apk";

caps.setCapability(MobileCapabilityType.APP, appUrl);

URL url = new URL("http://0.0.0.0:4723");

AppiumDriver driver = new AndroidDriver(url, caps);\*/

}

}

**(ANDROID DEBUG BRIDGE)**

It is a command line tool used to communicate with a device (mobile, emulators, set top box etc). (only for android). It allows us to do a variety of device actions such as installation and debugging apps.

It is a client server program that has 3 components

1. CLIENT: Sends the commands. The laptop/pc.

2. SERVER: Manages the communication between the client and the daemon.

3. DEAMON: It is a background process which runs on a device. Also called ADBD, it runs or executes the command.

**How adb works?**

When we start adb client, the client first checks whether there is an adb server process already running . If there isn’t than it starts the server process. When the server starts it binds to local TCP port 5037 and listens for commands sent from adb clients- all adb clients use port 5037 to communicate with the adb server.

The server than sets up connection to all running devices. Once ther server sets up the connections to all devices, we can use adb commands to access those devices.

adb = android debug bridge

adb is constant

adb devices--> It will fetch UDID.

Why we req. ADB?

-->To establish a communication between your system and device.

1. List connected devices

adb devices

1. If a device is connected start the adb server to be able to interact with the device.

adb start-server

adb kill-server

1. Install an App

adb install [apk\_path]

1. Uninstall an App

adb uninstall [package\_name]

Example:adb uninstall io.appium.android.apis

1. Pull files from device

adb pull [Mobile device file location] [local file location]

example: adb pull /storage/emulated/0/sportido\_ticket.jpg C:\Users\Srinidhi\Desktop\

1. Write files to device

adb push [local file location] [device file location]

example: adb push C:\Users\Srinidhi\Desktop\logcat.txt /storage/emulated/0/

1. Taking screenshot

Example: adb shell screencap -p /sdcard/screenshot1.png

adb pull /sdcard/screenshot1.png

adb shell rm /sdcard/screenshot1.png

or

adb exec-out screencap -p >screen.png

1. Capturing Videos

adb shell screenrecord /sdcard/NotAbleToLogin.mp4

adb pull /sdcard/NotAbleToLogin.mp4

adb shell rm /sdcard/NotAbleToLogin.mp4

or

adb exec-out screenrecord -p >screen.png

1. Adb logs

adb logcat

adb logcat>D:/April2020/pavan.txt

adb logcat package\_name

adb logcat \*:E | findstr com.example.package 🡪 to filter logs