FREE eBook

LEARNING appium

#appium

Table of Contents

About

Chapter 1: Getting started with appium

Remarks

Appium is an open source, cross-platform test automation tool for native, hybrid and mobile web apps, tested on simulators (iOS, FirefoxOS), emulators (Android), and real devices (iOS, Android, FirefoxOS).

Why Appium?

1. You don't have to recompile your app or modify it in any way, due to use of standard automation APIs on all platforms.
2. You don't have to recompile your app or modify it in any way, due to use of standard automation APIs on all platforms. You can write tests with your favorite dev tools using any WebDriver-compatible language such as Java, Objective-C, JavaScript with Node.js (in promise, callback or generator flavors), PHP, Python, Ruby, C#, Clojure, or Perl with the Selenium WebDriver API and language-specific client libraries.
3. You can use any testing framework.

Investing in the WebDriver protocol means you are betting on a single, free and open protocol for testing that has become a defacto standard. Don't lock yourself into a proprietary stack.

If you use Apple's UIAutomation library without Appium you can only write tests using JavaScript and you can only run tests through the Instruments application. Similarly, with Google's UiAutomator you can only write tests in Java. Appium opens up the possibility of true cross-platform native mobile automation.

How It Works

Appium drives various native automation frameworks and provides an API based on Selenium's WebDriver JSON wire protocol.

Appium drives Apple's UIAutomation library for versions before iOS 10, which is based on Dan Cuellar's work on iOS Auto. With the deprecation of the UIAutomation library, all iOS 10 and future version are driven by the XCUITest framework.

Android support uses the UiAutomator framework for newer platforms and Selendroid for older Android platforms.

FirefoxOS support leverages Marionette, an automation driver that is compatible with WebDriver and is used to automate Gecko-based platforms.

Versions

Version	Release Date
1.6.3	2016-12-12
1.6.2	2016-12-02
1.6.1	2016-11-24
1.6.0	2016-10-10
1.5.3	2016-06-07
1.5.2	2016-04-20
1.5.1	2016-03-29
1.5.0	2016-02-26
1.4.16	2015-11-20
1.4.15	2015-11-18
1.4.14	2015-11-06
1.4.13	2015-09-30
1.4.11	2015-09-16
1.4.10	2015-08-07
1.4.8	2015-07-16
1.4.7	2015-07-02
1.4.6	2015-06-19
1.4.3	2015-06-09
1.4.1	2015-05-21
1.4.0	2015-05-09
1.3.7	2015-03-25
1.3.6	2014-12-01

Examples

Installation or Setup

Pre-requirements

Check the requirements for each device type you wish to automate and make sure they're installed before attempting to use Appium!

iOS Requirements

- Mac OS X 10.10 or higher, 10.11.1 recommended
- XCode >= 6.0, 7.1.1 recommended
- Apple Developer Tools (iPhone simulator SDK, command line tools)
- Ensure you read the documentation on setting yourself up for iOS testing!

Android Requirements

- Android SDK API >= 17 (Additional features require 18/19)

- Appium supports Android on OS X, Linux and Windows. Make sure you follow the directions for setting up your environment properly for testing on different OSes:

    - linux
    - osx
    - windows

FirefoxOS Requirements

- Firefox OS Simulator

Installation of Appium

Global installation using Node.js

```
$ npm install -g appium
$ appium
```

Local installation from Github's master branch

```
$ git clone git@github.com:appium/appium.git
$ cd appium
$ npm install
$ node .
```

Using the App for Mac or Windows

- Download the Appium app
- Run it!

Writing Tests for Appium

Formatted version of the Appium docs can be found here with the ability to choose code example language from the top right corner.

Launching Appium for Android platform and creating sample test

Environment Setup:

- Download android sdk of API level 17 or more
- Node.js (https://nodejs.org/)
- Appium software (http://appium.io/)
- Selenium jars (http://www.seleniumhq.org/download/)
- Appium jar (https://search.maven.org/#search%7Cga%7C1%7Cg%3Aio.appium%20a%3Ajava-client)
- .apk file of the application which needs to be tested

Preconditions:

- make sure Eclipse is downloaded from www.eclipse.org/downloads/
- java is installed (both jdk and jre)
- android sdk is installed
- Make sure your environment variable (Path) for Java, Android SDK, Platform and platform-tools is set.

Steps to set Path on windows OS: Right Click “My Computer”. “Properties” On left panel “Advance System Settings” Select Environment Variables System Variables-> Type Path-> “Path” double click Enter the path to JAVA jdk in your system followed by (;) then path to your android sdk (;) path to your android platform (;) path to your android platform tools-> Click OK.

- Make sure Eclipse Plug-in is installed

Steps to install Eclipse Plug-in for Android: Start Eclipse, then select Help > Install New Software. Click Add, in the top-right corner. In the Add Repository dialog that appears, enter "ADT Plugin" for the Name and the following URL for the Location: https://dl-ssl.google.com/android/eclipse/ Click OK (If you have trouble acquiring the plugin, try using "http" in the Location URL, instead of "https" (https is preferred for security reasons).

- Make sure ANDROID_HOME variable is set.

Steps to set ANDROID_HOME variable: Go to Eclipse->Window on top panel->Preferences-> Double click Android on left panel In the Android preferences, Copy the SDK Location Right Click “My Computer”. “Properties” On left panel “Advance System Settings” Select Environment Variables On the top User Variables-> Select new-> Variable Name, Enter ANDROID_HOME, Variable Path-> Enter copied SDK location from Eclipse-> Click OK Then System Variables-> Select new-> Variable Name, Enter ANDROID_HOME, Variable Path-> Enter copied SDK location from Eclipse-> Click OK Exit

- Make sure Android Virtual Device Manager can be launched. Eclipse->Window on top panel->Android Virtual Device Manager-> Click on the existing virtual device if it exists/ Create a new one with customized configurations.-> Click on “Start” on the right panel of the window.-

> Launch

Launching Appium:

- Install node.js ("http://nodejs.org/").
- Launch Appium from command line from the below location: Goto Appium folder node_modules appiumbinshift+right clickopen command prompttype node appiumenter

Following should be displayed: info: Welcome to Appium v1.3.4 (REV c8c79a85fbd6870cd6fc3d66d038a115ebe22efe) info: Appium REST http interface listener started on 0.0.0.0:4723 info: Console LogLevel: debug info: Appium REST http interface listener started on 0.0.0.0:4723info: Console LogLevel: debug

Write a Program to launch Appium in Eclipse: package appium.com;

import java.net.MalformedURLException; import java.net.URL;

import org.openqa.selenium.remote.CapabilityType; import org.openqa.selenium.remote.DesiredCapabilities; import org.openqa.selenium.remote.RemoteWebDriver;

public class AppiumLaunch { public static void main(String args[]) throws MalformedURLException { RemoteWebDriver driver; DesiredCapabilities capabilities =new DesiredCapabilities();

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

capabilities.setCapability("version","4.4.2");
capabilities.setCapability("device ID","");
capabilities.setCapability("app-package","");
capabilities.setCapability(CapabilityType.BROWSER_NAME, "");

capabilities.setCapability("app-activity","");
capabilities.setCapability("takesScreenshot",true);

capabilities.setCapability("app","C:/Users/.......apk");

driver=new RemoteWebDriver( new URL("http://127.0.0.1:4723/wd/hub"), capabilities);
System.out.println("app is launched on the device");



}
```

}

- Make sure the path of the apk file in the system is correct
- Make sure the path to the apk file in your system is correct in the program. Use correct package and activity which can be found by decompiling the apk file. For decompiling apk file, go to http://www.decompileandroid.com.

Steps to launch appium for android:

1. First start the appium server on command prompt or by running the appium.exe file.
2. Check whether the device is connected and displayed in adb: adb devices
3. Execute the program on the Eclipse. The program will get executed and .apk file which was installed in the device will launch the app.

Read Getting started with appium online: https://riptutorial.com/appium/topic/5122/getting-started-with-appium

Chapter 2: Java client

Remarks

Java Client API

Java Client Source Code

Examples

Android Play Store automation (Real device)

File Structure:

- pom.xml
- src/test/java/PlayStoreAutomation.java

Launch command:

```
mvn test -Dtest=PlayStoreAutomation
```

PlayStoreAutomation.java

```
import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;
import io.appium.java_client.android.AndroidDriver;
import io.appium.java_client.android.AndroidKeyCode;
import io.appium.java_client.MobileElement;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.By;
import java.util.concurrent.TimeUnit;
import java.net.URL;

public class PlayStoreAutomation {
    public static AndroidDriver<MobileElement> driver;

    @BeforeClass
    public static void setUp() throws Exception {
        DesiredCapabilities capabilities = new DesiredCapabilities();
        capabilities.setCapability("platformName", "Android");
        capabilities.setCapability("deviceName", "Android Device");
        capabilities.setCapability("appPackage", "com.android.vending");
        capabilities.setCapability("appActivity",
"com.google.android.finsky.activities.MainActivity");

        driver = new AndroidDriver<MobileElement>(new URL("http://localhost:4723/wd/hub"),
capabilities);
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
    }
    @AfterClass
    public static void tearDown() {
        driver.quit();
    }

    @Test
    public void testPlayStore() throws Exception {
        driver.findElement(By.id("com.android.vending:id/text_container")).sendKeys("Google");
        driver.pressKeyCode(AndroidKeyCode.ENTER);

        // First item in the search result by Xpath

driver.findElement(By.xpath("//android.support.v7.widget.RecyclerView[1]/android.widget.LinearLayout[1]

        // Confirm element found
        driver.findElement(By.xpath("//android.widget.TextView[@text='Google']"));
    }
}
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.tester.appium</groupId>
    <artifactId>tester-tests</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>io.appium</groupId>
            <artifactId>java-client</artifactId>
            <version>4.0.0</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.10</version>
                <configuration>
```

```
                    <reportsDirectory>${project.build.directory}/reports</reportsDirectory>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Read Java client online: https://riptutorial.com/appium/topic/6195/java-client

Chapter 3: Parallel Testing in Appium

Introduction

Parallel execution in appium using selenium GRID concept. Please find step by step process.

Examples

Step By Step process

Parallel Testing with Appium using GRID: I will describe the way which worked for me. Create Selenium Grid with Appium

1. Setup the selenium grid Download selenium standalone server jar on local file system Open your terminal and navigate to the directory to where you placed the jar file and execute the following command:

```
java -jar selenium-server-standalone-2.53.3.jar -role hub
Open http://localhost:4444/grid/console and you should be able to see GRID console in your
browser.
```

2. Setup the Appium Nodes Here you have to create the json files. Suppose you want to run on two devices then create two different json file. Here is one json file , I have as: { "capabilities": [{ "applicationName": "ONEPLUS A3003", "browserName": "ONEPLUS A3003", "platformName": "ANDROID", "maxInstances": 1 }], "configuration": { "cleanUpCycle": 2000, "timeout": 30000, "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy", "host": "127.0.0.1", "port": 4723, "maxSession": 1, "register": true, "registerCycle": 5000, "hubPort": 4444, "hubHost": "your ip address" } } save the above file as jasonFile1.json Here applicationName will be -> Your Mobile->settings->about phone->Model number Here hubHost will be your ip address Here note that you need to go as default cmd location then run below command

appium --nodeconfig C:/richa/jasonfile1.json -p 4723 -bp 4724 -U xxxx

i)Note that you need to provide the absolute pasth of the json file located ii) port as 4723 iii) Bootstrap port as 4724 iv) -U for example I have given as xxxx

you can find the device id as -> Your Mobile->settings->status->Serial number You can also do “adb device” and check this device id.

Then it will create the Selenium Grid with one device.

Now again run the second json file and you will get appium started Here is second json file:

{ "capabilities": [{ "applicationName": "Lenovo K50a40", "browserName": "Lenovo K50a40", "platformName": "ANDROID", "maxInstances": 1 }], "configuration": { "cleanUpCycle": 2000, "timeout": 30000, "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy", "host": "127.0.0.1", "port": 4730, "maxSession": 1, "register": true, "registerCycle": 5000, "hubPort": 4444, "hubHost": "your ip address" } } save the above file as jasonFile2.json

Start the second node with Lenovo mobile. appium --nodeconfig C:/richa/ jasonFile2.json -p 4730 -bp 4731 -U xxxx

Selenium Grid will look like this

3)Create TestNG parallel execution methods to run you test.

--> Please note value of device name will be the udid you provided earlier. You can get it by running adb devices on your command prompt.

4.

Now create SearchHotelTestCase.Java as below: package com.trivago.TestCases;

import java.net.MalformedURLException; import java.net.URL; import java.util.concurrent.TimeUnit;

import org.openqa.selenium.remote.DesiredCapabilities; import org.openqa.selenium.remote.RemoteWebDriver; import org.testng.annotations.BeforeMethod; import org.testng.annotations.Parameters; import org.testng.annotations.Test;

import com.trivago.pages.LocaleSelectionPage; import com.trivago.pages.SearchLocation; import com.trivago.pages.SplashScreenPage;

import io.appium.java_client.MobileElement; import io.appium.java_client.android.AndroidDriver;

public class SearchHotelTestCase { private AndroidDriver driver;

@Parameters({ "deviceName_","platformVersion_","applicationName_" }) @BeforeMethod public void beforeMethod(String deviceName_,String platformVersion_,String applicationName_) throws MalformedURLException, InterruptedException {

DesiredCapabilities capabilities = new DesiredCapabilities(); capabilities.setCapability("deviceName", deviceName_); capabilities.setCapability("platformVersion", platformVersion_); capabilities.setCapability("platformName", "Android"); capabilities.setCapability("applicationName", applicationName_); capabilities.setCapability("app", "/Users/richa.b.shrivastava/Downloads/com.trivago_2017-04-28.apk"); capabilities.setCapability("appPackage", "com.trivago"); capabilities.setCapability("appActivity", "com.trivago.activities.SplashActivity");

URL url = new URL("http://0.0.0.0:4723/wd/hub/"); System.out.println("before webdriver"); driver = new AndroidDriver(url, capabilities); System.out.println("after webdriver"); driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS); Thread.sleep(4000); }

@Test public void SearchHotel() { //Create the objects of Page Class LocaleSelectionPage

localeSelectionPage = new LocaleSelectionPage(driver); SplashScreenPage splashScreenPage = new SplashScreenPage(driver); SearchLocation searchLocation = new SearchLocation(driver);

//Call the methods of page class localeSelectionPage.selectLocale(); splashScreenPage.clickSplashSearchText(); searchLocation.inputSearchText("Paris"); searchLocation.selectSuggestions("Eiffel Tower, Paris");

}

}

Read Parallel Testing in Appium online: https://riptutorial.com/appium/topic/10016/parallel-testing-in-appium

Credits

S. No	Chapters	Contributors
1	Getting started with appium	BenJi, Community, Domestus, mrtuovinen, Priya, Richa Shrivastava
2	Java client	Domestus
3	Parallel Testing in Appium	Richa Shrivastava