Kostenloses eBook

LERNEN

appium

Free unaffiliated eBook created from
Stack Overflow contributors.

#appium

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: appium

It is an unofficial and free appium ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official appium.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit Appium

Bemerkungen

Appium ist ein plattformübergreifendes Open Source-Testautomatisierungs-Tool für native, Hybrid- und mobile Web-Apps, das auf Simulatoren (iOS, FirefoxOS), Emulatoren (Android) und realen Geräten (iOS, Android, FirefoxOS) getestet wurde.

Warum Appium?

1. Sie müssen Ihre App nicht neu kompilieren oder in irgendeiner Weise ändern, da auf allen Plattformen standardisierte Automatisierungs-APIs verwendet werden.
2. Sie müssen Ihre App nicht neu kompilieren oder in irgendeiner Weise ändern, da auf allen Plattformen standardisierte Automatisierungs-APIs verwendet werden. Sie können Tests mit Ihren bevorzugten Entwicklertools in jeder WebDriver- kompatiblen Sprache wie Java , Objective-C , JavaScript mit Node.js (in Versprechungs-, Rückruf- oder Generator-Varianten), PHP, Python , Ruby , C # , Clojure oder Perl schreiben mit der Selenium WebDriver API und sprachspezifischen Client-Bibliotheken.
3. Sie können jedes Testframework verwenden.

Wenn Sie in das WebDriver-Protokoll investieren, setzen Sie auf ein einziges, freies und offenes Protokoll für Tests, die zum Defacto-Standard geworden sind. Schließ dich nicht in einen proprietären Stack ein.

Wenn Sie die UIAutomation-Bibliothek von Apple ohne Appium verwenden, können Sie Tests nur mit JavaScript schreiben und Sie können nur Tests über die Instruments-Anwendung ausführen. Ebenso können Sie mit dem UiAutomator von Google nur Tests in Java schreiben. Appium eröffnet die Möglichkeit einer echten plattformübergreifenden nativen mobilen Automatisierung.

Wie es funktioniert

Appium unterstützt verschiedene native Automatisierungsframeworks und stellt eine API bereit, die auf dem WebDriver JSON-Drahtprotokoll von Selenium basiert.

Appium führt die UIAutomation-Bibliothek von Apple für Versionen vor iOS 10 aus, die auf der Arbeit von Dan Cuellar für iOS Auto basiert. Mit der Ablehnung der UIAutomation-Bibliothek werden alle iOS 10- und zukünftigen Versionen vom XCUITest-Framework gesteuert.

Die Android-Unterstützung verwendet das UiAutomator-Framework für neuere Plattformen und Selendroid für ältere Android-Plattformen.

Die FirefoxOS-Unterstützung nutzt Marionette , einen mit WebDriver kompatiblen Automatisierungstreiber, der zur Automatisierung von Gecko-basierten Plattformen verwendet wird.

Versionen

Ausführung	Veröffentlichungsdatum
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 oder Setup

Voraussetzungen

Überprüfen Sie die Anforderungen für jeden Gerätetyp, den Sie automatisieren möchten, und stellen Sie sicher, dass sie installiert sind, bevor Sie Appium verwenden.

iOS-Anforderungen

- Mac OS X 10.10 oder höher, 10.11.1 empfohlen
- XCode> = 6.0, 7.1.1 empfohlen
- Apple Developer Tools (iPhone Simulator SDK, Befehlszeilentools)
- Stellen Sie sicher, dass Sie die Dokumentation zum Einrichten des iOS-Tests gelesen haben!

Android-Anforderungen

- Android SDK- API> = 17 (Zusätzliche Funktionen erfordern 18/19)
- Appium unterstützt Android unter OS X, Linux und Windows. Stellen Sie sicher, dass Sie die Anweisungen zum Einrichten Ihrer Umgebung zum Testen auf verschiedenen Betriebssystemen befolgen:
    - Linux
    - osx
    - Fenster

FirefoxOS-Anforderungen

- Firefox OS Simulator

Installation von Appium

Globale Installation mit Node.js

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

Lokale Installation von Githubs Hauptzweig

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

App für Mac oder Windows verwenden

- Laden Sie die Appium App herunter
- Starte es!

Schreibtests für Appium

Formatierte Version der Appium docs finden sich hier mit der Fähigkeit, Codebeispiel Sprache aus der rechten oberen Ecke zu wählen.

Appium für Android-Plattform starten und Beispieltest erstellen

Umgebungseinstellung:

- Laden Sie android sdk von API Level 17 oder mehr herunter
- Node.js (https://nodejs.org/)
- Appium-Software (http://appium.io/)
- Selengläser (http://www.seleniumhq.org/download/)
- Appium jar (https://search.maven.org/#search%7Cga%7C1%7Cg%3Aio.appium%20a%3Ajava-client)
- .apk-Datei der Anwendung, die getestet werden muss

Voraussetzungen:

- Stellen Sie sicher, dass Eclipse von www.eclipse.org/downloads/ heruntergeladen wird.
- Java ist installiert (sowohl jdk als auch jre)
- Android SDK ist installiert
- Stellen Sie sicher, dass Ihre Umgebungsvariable (Pfad) für Java, Android SDK, Plattform und Plattform-Tools festgelegt ist.

Schritte zum Einstellen des Pfads unter Windows OS: Klicken Sie mit der rechten Maustaste auf „Arbeitsplatz“. Properties „Eigenschaften“ Auf der linken Seite „Erweiterte Systemeinstellungen“ Umgebungsvariablen auswählen Systemvariablen-> Typpfad-> „Pfad“ Doppelklick Geben Sie den Pfad zu JAVA jdk in Ihrem System ein, gefolgt von (;) und dann dem Pfad zu Ihrem android sdk (;) Pfad zu Ihrer Android-Plattform (;) Pfad zu Ihren Android-Plattform-Tools-> Klicken Sie auf OK.

- Stellen Sie sicher, dass das Eclipse-Plug-In installiert ist

Schritte zum Installieren des Eclipse-Plug-Ins für Android: Starten Sie Eclipse und wählen Sie Hilfe> Neue Software installieren. Klicken Sie oben rechts auf Hinzufügen. Geben Sie im angezeigten Dialogfeld Repository hinzufügen "ADT Plugin" als Namen und die folgende URL als Standort ein: https://dl-ssl.google.com/android/eclipse/ Klicken Sie auf OK (Wenn Sie Probleme beim Abrufen haben Verwenden Sie für das Plugin "http" als "URL" anstelle von "https" (https wird aus Sicherheitsgründen bevorzugt).

- Stellen Sie sicher, dass die Variable ANDROID_HOME gesetzt ist.

So stellen Sie die Variable ANDROID_HOME ein: Gehen Sie zu Eclipse-> Fenster im oberen Bedienfeld-> Voreinstellungen-> Doppelklicken Sie im linken Bedienfeld auf Android. Kopieren Sie den SDK-Speicherort in den Android-Einstellungen. Klicken Sie mit der rechten Maustaste auf „Arbeitsplatz“. Properties „Eigenschaften“ Auf der linken Seite „Erweiterte Systemeinstellungen“. Environment Umgebungsvariablen auswählen. The Oberste Benutzervariablen-> Neue

auswählen> Variablenname, ANDROID_HOME, Variablenpfad eingeben.> Geben Sie den kopierten SDK-Speicherort von Eclipse- ein Dann Systemvariablen-> Wählen Sie Neu-> Variablenname aus, geben Sie ANDROID_HOME, Variablenpfad-> Geben Sie den kopierten SDK-Speicherort von Eclipse ein. -> Klicken Sie auf OK Beenden

- Stellen Sie sicher, dass der Android Virtual Device Manager gestartet werden kann. Eclipse-> Fenster im oberen Bedienfeld-> Android Virtual Device Manager-> Klicken Sie auf das vorhandene virtuelle Gerät, falls vorhanden, / Erstellen Sie ein neues mit benutzerdefinierten Konfigurationen. -> Klicken Sie im rechten Fensterbereich auf "Start".> Starten

Appium starten:

- Installieren Sie node.js (" http://nodejs.org/ ").
- Starten Sie Appium von der Befehlszeile aus dem folgenden Speicherort: Springen Sie in den Appium-Ordner node_modules appiumbinshift + Rechtsklickopen-Eingabeaufforderung type node appiumenter

Folgendes sollte angezeigt werden: info: Willkommen bei Appium v1.3.4 (REV c8c79a85fbd6870cd6fc3d66d038a115ebe22efe) info: Appium REST-http-Listener wurde mit 0.0.0.0:4723 gestartet. Info: Console LogLevel: debug info: Appium REST-http-Listener mit 0.0.0.0 4723info: Console LogLevel: Debuggen

Schreiben Sie ein Programm zum Starten von 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;

öffentliche Klasse AppiumLaunch {public static void main (String args []) löst MalformedURLException {RemoteWebDriver driver; DesiredCapabilities-Fähigkeiten = 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");



}
```

}

- Stellen Sie sicher, dass der Pfad der APK-Datei im System korrekt ist
- Stellen Sie sicher, dass der Pfad zur APK-Datei in Ihrem System im Programm korrekt ist. Verwenden Sie das richtige Paket und die richtige Aktivität, die durch Dekompilieren der APK-Datei gefunden werden kann. Um eine apk-Datei zu dekompilieren, gehen Sie zu http://www.decompileandroid.com .

Schritte zum Starten von appium für Android:

1. Starten Sie den Appium-Server zunächst an der Eingabeaufforderung oder indem Sie die Datei appium.exe ausführen.
2. Prüfen Sie, ob das Gerät angeschlossen und in adb: adb-Geräten angezeigt wird
3. Führen Sie das Programm auf der Eclipse aus. Das Programm wird ausgeführt und die .apk-Datei, die auf dem Gerät installiert wurde, startet die App.

Erste Schritte mit Appium online lesen: https://riptutorial.com/de/appium/topic/5122/erste-schritte-mit-appium

Kapitel 2: Java-Client

Bemerkungen

Java-Client-API

Java-Client-Quellcode

Examples

Android Play Store-Automatisierung (Echtgerät)

Dateistruktur:

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

Startbefehl:

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>
```

Java-Client online lesen: https://riptutorial.com/de/appium/topic/6195/java-client

Kapitel 3: Parallele Prüfung in Appium

Einführung

Parallele Ausführung in Appium mit Selen-GRID-Konzept. Bitte finden Sie Schritt für Schritt.

Examples

Schritt für Schritt

Parallele Tests mit Appium mit GRID: Ich werde den Weg beschreiben, der für mich funktioniert hat. Erstellen Sie mit Appium ein Selen-Gitter

1. Selenium-Grid einrichten Selenium Standalone-Server-JAR im lokalen Dateisystem herunterladen Öffnen Sie Ihr Terminal und navigieren Sie zu dem Verzeichnis, in dem Sie die JAR-Datei abgelegt haben, und führen Sie den folgenden Befehl aus:

```
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. Appium-Knoten einrichten Hier müssen Sie die Json-Dateien erstellen. Angenommen, Sie möchten auf zwei Geräten laufen und dann zwei verschiedene Json-Dateien erstellen. Hier ist eine Json-Datei, die ich habe: {"Fähigkeiten": [{"Anwendungsname": "ONEPLUS A3003", "Browsername": "ONEPLUS A3003", "Plattformname": "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": "Ihre IP-Adresse"}} Speichern Sie die obige Datei als jasonFile1.json. Hier wird applicationName - > Ihr Handy-> Einstellungen-> Über Telefon-> Modellnummer Hier wird HubHost Ihre IP-Adresse sein. Beachten Sie, dass Sie als Standard-Cmd-Speicherort gehen und dann unter dem Befehl ausführen müssen

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

i) Beachten Sie, dass Sie den absoluten Pasth der json-Datei angeben müssen. ii) port als 4723 iii) Bootstrap-Port als 4724 iv) -U zum Beispiel habe ich als xxxx angegeben

Sie finden die Geräte-ID unter -> Ihr Handy-> Einstellungen-> Status-> Seriennummer Sie können auch "ADB-Gerät" ausführen und diese Geräte-ID überprüfen.

Dann wird das Selen-Gitter mit einem Gerät erstellt.

Führen Sie nun erneut die zweite Json-Datei aus, und Sie können Appium starten. Hier ist die zweite Json-Datei:

{"Fähigkeiten": [{"Anwendungsname": "Lenovo K50a40", "Browsername": "Lenovo K50a40", "Plattformname": "ANDROID", "maxInstances": 1}], "Konfiguration": {"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 ":" Ihre IP-Adresse "}} Speichern Sie die obige Datei als" jasonFile2.json "

Starten Sie den zweiten Knoten mit Lenovo Mobile. appium --nodeconfig C: / richa / jasonFile2.json -p 4730 -bp 4731 -U xxxx

Selen-Gitter wird so aussehen

3) Erstellen Sie parallele TestNG-Ausführungsmethoden, um Ihren Test auszuführen.

-> Bitte beachten Sie, dass der Wert des Gerätenamens das zuvor angegebene udid ist. Sie können es erhalten, indem Sie Adb-Geräte an der Eingabeaufforderung ausführen.

4.

Erstellen Sie nun SearchHotelTestCase.Java wie folgt: 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;

öffentliche Klasse SearchHotelTestCase {privater AndroidDriver-Treiber;

@Parameters ({"deviceName _", "platformVersion _", "applicationName_"}) @BeforeMethod public void beforeMethod (String deviceName_, String platformVersion_, String applicationName_) löst MalformedURLException, InterruptedException

DesiredCapabilities-Fähigkeiten = new DesiredCapabilities (); Capabilities.setCapability ("Gerätename", Gerätename_); Capabilities.setCapability ("Plattformversion", Plattformversion_); Capabilities.setCapability ("Plattformname", "Android"); Capabilities.setCapability ("applicationName", applicationName_); Capabilities.setCapability ("app", "/Users/richa.b.shrivastava/Downloads/com.trivago_2017-04-28.apk"); Capabilities.setCapability ("appPackage", "com.trivago"); Fähigkeiten.setCapability ("appActivity", "com.trivago.activities.SplashActivity");

URL-URL = neue URL (" http://0.0.0.0:4723/wd/hub/ "); System.out.println ("before webdriver"); Treiber = neuer AndroidDriver (URL, Fähigkeiten); System.out.println ("after webdriver"); driver.manage (). timeouts (). implicitWait (10, TimeUnit.SECONDS); Fadenschlaf (4000); }

@Test public void SearchHotel () {// Objekte der Seitenklasse LocaleSelectionPage erstellen localeSelectionPage = new LocaleSelectionPage (Treiber); SplashScreenPage splashScreenPage = new SplashScreenPage (Treiber); SearchLocation searchLocation = neuer SearchLocation (Treiber);

// Aufruf der Methoden der Seitenklasse localeSelectionPage.selectLocale (); splashScreenPage.clickSplashSearchText (); searchLocation.inputSearchText ("Paris"); searchLocation.selectSuggestions ("Eiffelturm, Paris");

}

}

Parallele Prüfung in Appium online lesen: https://riptutorial.com/de/appium/topic/10016/parallele-prufung-in-appium

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Appium	BenJi, Community, Domestus, mrtuovinen, Priya, Richa Shrivastava
2	Java-Client	Domestus
3	Parallele Prüfung in Appium	Richa Shrivastava