eBook Gratuit

APPRENEZ appium

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#appium

Table des matières

À propos

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

Chapitre 1: Démarrer avec Appium

Remarques

Appium est un outil d'automatisation de test multi-plateforme open source pour les applications Web natives, hybrides et mobiles, testé sur des simulateurs (iOS, FirefoxOS), des émulateurs (Android) et des périphériques réels (iOS, Android, FirefoxOS).

Pourquoi Appium?

1. Vous n'avez pas besoin de recompiler votre application ou de la modifier de quelque manière que ce soit, en raison de l'utilisation d'API d'automatisation standard sur toutes les plates-formes.
2. Vous n'avez pas besoin de recompiler votre application ou de la modifier de quelque manière que ce soit, en raison de l'utilisation d'API d'automatisation standard sur toutes les plates-formes. Vous pouvez écrire des tests avec vos outils de développement favoris en utilisant n'importe quel langage compatible WebDriver , tel que Java , Objective-C , JavaScript avec Node.js (promis, callback ou générateur), PHP, Python , Ruby , C # , Clojure ou Perl avec l'API Selenium WebDriver et les bibliothèques client spécifiques à une langue.
3. Vous pouvez utiliser n'importe quel framework de test.

Investir dans le protocole WebDriver signifie que vous pariez sur un protocole unique, gratuit et ouvert pour les tests, devenu un standard de facto. Ne vous enfermez pas dans une pile propriétaire.

Si vous utilisez la bibliothèque UIAutomation d'Apple sans Appium, vous ne pouvez écrire que des tests à l'aide de JavaScript et vous pouvez uniquement exécuter des tests via l'application Instruments. De même, avec UiAutomator de Google, vous ne pouvez écrire que des tests en Java. Appium ouvre la possibilité d'une véritable automatisation mobile native multiplateforme.

Comment ça marche

Appium pilote divers frameworks d'automatisation natifs et fournit une API basée sur le protocole de connexion WebDriver JSON de Selenium.

Appium pilote la bibliothèque UIAutomation d'Apple pour les versions antérieures à iOS 10, basée sur le travail de Dan Cuellar sur iOS Auto. Avec la dépréciation de la bibliothèque UIAutomation, toutes les versions d'iOS 10 et futures sont pilotées par le framework XCUITest.

Le support Android utilise le framework UiAutomator pour les nouvelles plates-formes et Selendroid pour les anciennes plates-formes Android.

La prise en charge de FirefoxOS tire parti de Marionette , un pilote d'automatisation compatible avec WebDriver et utilisé pour automatiser les plates-formes basées sur Gecko.

Versions

Version	Date de sortie
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

Pré-requis

Vérifiez les exigences pour chaque type d'appareil que vous souhaitez automatiser et assurez-vous qu'elles sont installées avant d'essayer d'utiliser Appium!

Configuration requise pour iOS

- Mac OS X 10.10 ou supérieur, 10.11.1 recommandé
- XCode> = 6.0, 7.1.1 recommandé
- Outils de développement Apple (SDK simulateur iPhone, outils de ligne de commande)
- Assurez-vous de lire la documentation pour vous installer pour les tests iOS!

Exigences Android

- Android SDK API> = 17 (Les fonctionnalités supplémentaires requièrent 18/19)
- Appium prend en charge Android sur OS X, Linux et Windows. Assurez-vous de suivre les instructions pour configurer correctement votre environnement pour tester sur différents systèmes d'exploitation:
    - linux
    - osx
    - les fenêtres

FirefoxOS exigences

- Firefox OS Simulator

Installation d'Appium

Installation globale à l'aide de Node.js

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

Installation locale à partir de la branche principale de Github

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

Utiliser l'application pour Mac ou Windows

- Téléchargez l'application Appium
- Exécuter!

Tests d'écriture pour Appium

La version formatée des documents Appium peut être trouvée ici avec la possibilité de choisir un langage de code à partir du coin supérieur droit.

Lancement de la plate-forme Appium pour Android et création d'un exemple de test

Configuration de l'environnement:

- Télécharger le SDK Android de l'API niveau 17 ou plus
- Node.js (https://nodejs.org/)
- Logiciel Appium (http://appium.io/)
- Bocaux de sélénium (http://www.seleniumhq.org/download/)
- Appium jar (https://search.maven.org/#search%7Cga%7C1%7Cg%3Aio.appium%20a%3Ajava-client)
- Fichier .apk de l'application à tester

Conditions préalables:

- assurez-vous que Eclipse est téléchargé depuis www.eclipse.org/downloads/
- java est installé (jdk et jre)
- Android sdk est installé
- Assurez-vous que votre variable d'environnement (Path) pour Java, Android SDK, Platform et Platform-Tools est définie.

Étapes pour définir le chemin d'accès sous Windows OS:: Cliquez avec le bouton droit sur “Poste de travail”. Properties "Propriétés" Dans le panneau de gauche "Paramètres système avancés" Sélectionner les variables d'environnement iables Variables système-> Chemin d'accès -> "Chemin" double-cliquez Entrez le chemin d'accès JAVA dans votre système suivi de (;) Android sdk (;) Chemin d'accès à votre plate-forme Android (;) Chemin d'accès à vos outils de plate-forme Android-> Cliquez sur OK.

- Assurez-vous que le plug-in Eclipse est installé

Procédure d'installation du plug-in Eclipse pour Android: Démarrez Eclipse, puis sélectionnez Aide> Installer un nouveau logiciel. Cliquez sur Ajouter, dans le coin supérieur droit. Dans la boîte de dialogue Ajouter un référentiel qui apparaît, entrez "ADT Plugin" pour le nom et l'URL suivante pour l'emplacement: https://dl-ssl.google.com/android/eclipse/ Cliquez sur OK (Si vous rencontrez des difficultés pour acquérir Dans le plug-in, essayez d'utiliser "http" dans l'URL d'emplacement, au lieu de "https" (https est préférable pour des raisons de sécurité).

- Assurez-vous que la variable ANDROID_HOME est définie.

Étapes pour définir la variable ANDROID_HOME: to Accédez à Eclipse-> Fenêtre sur le panneau supérieur-> Préférences-> Double-cliquez sur Android dans le panneau de gauche Dans les préférences Android, copiez l'emplacement du SDK Click Cliquez avec le bouton droit sur «Poste de travail». Properties “Propriétés” Dans le panneau de gauche “Paramètres système avancés”

Sélectionner les variables d'environnement top En haut Variables utilisateur-> Sélectionnez nouveau-> Nom de la variable, entrez ANDROID_HOME, chemin de variable-> entrez l'emplacement du SDK copié d'Eclipse-> cliquez sur OK Puis variables système-> Sélectionnez nouveau-> Nom de la variable, entrez ANDROID_HOME, chemin de la variable-> entrez l'emplacement du SDK copié à partir d'Eclipse-> cliquez sur OK quitter

- Assurez-vous que Android Virtual Device Manager peut être lancé. Eclipse-> Fenêtre du panneau supérieur-> Android Virtual Device Manager-> Cliquez sur le périphérique virtuel existant s'il existe / Créez-en un avec des configurations personnalisées .-> Cliquez sur «Démarrer» dans le volet droit de la fenêtre .-> lancement

Lancement d'Appium:

- Installez node.js (“ http://nodejs.org/ ”).
- Lancez Appium à partir de la ligne de commande à partir de l’emplacement ci-dessous:

Les informations suivantes doivent être affichées: info: Bienvenue dans Appium v1.3.4 (REV c8c79a85fbd6870cd6fc3d66d038a115ebe22efe) info: le port d'écoute de l'interface http AppEST REST a démarré sur 0.0.0.0:4723 info: Console LogLevel: info de débogage: 4723info: Console LogLevel: debug

Rédiger un programme pour lancer Appium dans 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;

classe publique AppiumLaunch {public static void main (String args []) lève MalformedURLException {RemoteWebDriver driver; DesiredCapabilities capacités = 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");



}
```

}

- Assurez-vous que le chemin du fichier apk dans le système est correct
- Assurez-vous que le chemin d'accès au fichier apk de votre système est correct dans le programme. Utilisez le package et l'activité corrects qui peuvent être trouvés en décompilant le fichier apk. Pour décompiler le fichier apk, rendez-vous sur http://www.decompileandroid.com .

Étapes pour lancer appium pour Android:

1. Commencez par démarrer le serveur appium à l'invite de commande ou en exécutant le fichier appium.exe.
2. Vérifiez si le périphérique est connecté et affiché dans les périphériques adb: adb
3. Exécutez le programme sur l'Eclipse. Le programme sera exécuté et le fichier .apk installé sur l'appareil lancera l'application.

Lire Démarrer avec Appium en ligne: https://riptutorial.com/fr/appium/topic/5122/demarrer-avec-appium

Chapitre 2: Client Java

Remarques

API du client Java

Code source du client Java

Examples

Android Play Store automatisation (appareil réel)

Structure de fichier:

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

Commande de lancement:

test mvn -Destest = 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>
```

Lire Client Java en ligne: https://riptutorial.com/fr/appium/topic/6195/client-java

Chapitre 3: Tests parallèles dans Appium

Introduction

Exécution parallèle en appium utilisant le concept GRID de sélénium. Veuillez trouver étape par étape le processus.

Examples

Processus pas à pas

Tests parallèles avec Appium en utilisant GRID: Je décrirai la manière dont cela a fonctionné pour moi. Créer une grille de sélénium avec Appium

1. Configurer la grille de sélénium Télécharger le fichier de serveur autonome de sélénium sur le système de fichiers local Ouvrez votre terminal et accédez au répertoire où vous avez placé le fichier jar et exécutez la commande suivante:

```
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. Configurer les nœuds Appium Ici, vous devez créer les fichiers json. Supposons que vous voulez exécuter sur deux appareils, puis créez deux fichiers json différents. Voici un fichier json, j'ai comme: {"capacités": [{"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": "votre adresse IP"}} enregistrez le fichier ci-dessus en tant que jasonFile1.json > Votre mobile-> paramètres-> à propos de téléphone-> numéro de modèle Ici, hubHost sera votre adresse IP Notez ici que vous devez vous rendre par défaut à l'emplacement de cmd, puis exécuter la commande ci-dessous

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

i) Notez que vous devez fournir le pas absolu du fichier json situé ii) port sous le numéro 4723 iii) Port Bootstrap sous la forme 4724 iv) -U par exemple j'ai donné comme xxxx

Vous pouvez trouver l'identifiant de l'appareil comme -> Votre Mobile-> Paramètres-> Status-> Numéro de série Vous pouvez également faire “adb device” et vérifier cet identifiant d'appareil.

Ensuite, il créera la grille Selenium avec un seul appareil.

Maintenant, exécutez à nouveau le deuxième fichier json et vous obtiendrez l'application lancée. Voici le deuxième fichier json:

{"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 ":" votre adresse IP "}} enregistrez le fichier ci-dessus en tant que jasonFile2.json

Démarrez le deuxième nœud avec Lenovo mobile. appium --nodeconfig C: / richa / jasonFile2.json -p 4730 -bp 4731 -U xxxx

Selenium Grid ressemblera à ceci

3) Créez des méthodes d'exécution parallèle TestNG pour exécuter votre test.

-> S'il vous plaît noter la valeur du nom de périphérique sera le udid que vous avez fourni plus tôt. Vous pouvez l'obtenir en exécutant des périphériques adb sur votre invite de commande.

4.

Maintenant, créez SearchHotelTestCase.Java comme ci-dessous: 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;

classe publique SearchHotelTestCase {pilote AndroidDriver privé;

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

DesiredCapabilities capacités = new DesiredCapabilities (); capabilities.setCapability ("deviceName", deviceName_); capabilities.setCapability ("platformVersion", platformVersion_); capabilities.setCapability ("platformName", "Android"); capabilities.setCapability ("nom_application", nom_application_); 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 = nouvelle URL (" http://0.0.0.0:4723/wd/hub/ "); System.out.println ("avant webdriver"); pilote = nouveau AndroidDriver (URL, capacités); System.out.println ("après webdriver"); driver.manage (). timeouts (). implicitlyWait (10, TimeUnit.SECONDS); Thread.sleep (4000); }

@Test public void SearchHotel () {// Crée les objets de la classe de page LocaleSelectionPage localeSelectionPage = new LocaleSelectionPage (pilote); SplashScreenPage splashScreenPage = new SplashScreenPage (pilote); SearchLocation searchLocation = new SearchLocation (pilote);

// Appelle les méthodes de la classe de page localeSelectionPage.selectLocale (); splashScreenPage.clickSplashSearchText (); searchLocation.inputSearchText ("Paris"); searchLocation.selectSuggestions ("Tour Eiffel, Paris");

}

}

Lire Tests parallèles dans Appium en ligne: https://riptutorial.com/fr/appium/topic/10016/tests-paralleles-dans-appium

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec Appium	BenJi, Community, Domestus, mrtuovinen, Priya, Richa Shrivastava
2	Client Java	Domestus
3	Tests parallèles dans Appium	Richa Shrivastava