EBook Gratuito

APPRENDIMENTO

appium

Free unaffiliated eBook created from
Stack Overflow contributors.

#appium

Sommario

Di

Capitolo 1: Iniziare con Appium

Osservazioni

Appium è uno strumento di automazione di test open source e multipiattaforma per applicazioni web native, ibride e mobili, testato su simulatori (iOS, FirefoxOS), emulatori (Android) e dispositivi reali (iOS, Android, FirefoxOS).

Perché Appio?

1. Non è necessario ricompilare la tua app o modificarla in alcun modo, a causa dell'uso di API di automazione standard su tutte le piattaforme.
2. Non è necessario ricompilare la tua app o modificarla in alcun modo, a causa dell'uso di API di automazione standard su tutte le piattaforme. È possibile scrivere test con gli strumenti di sviluppo preferiti utilizzando qualsiasi linguaggio compatibile con WebDriver come Java , Objective-C , JavaScript con Node.js (in promessa, funzioni callback o generatore), PHP, Python , Ruby , C # , Clojure o Perl con l'API Selenium WebDriver e le librerie client specifiche della lingua.
3. Puoi usare qualsiasi framework di test.

Investire nel protocollo WebDriver significa che stai scommettendo su un singolo protocollo libero e aperto per i test che è diventato uno standard de facto. Non chiuderti in uno stack proprietario.

Se si utilizza la libreria UIAutomation di Apple senza Appium, è possibile scrivere solo test utilizzando JavaScript e si possono eseguire solo test attraverso l'applicazione Instruments. Allo stesso modo, con UiAutomator di Google è possibile scrivere solo test in Java. Appium offre la possibilità di un'autentica automazione mobile nativa multipiattaforma.

Come funziona

Appium supporta vari framework di automazione nativi e fornisce un'API basata sul protocollo wire JSON WebDriver di Selenium.

Appium guida la libreria UIAutomation di Apple per le versioni precedenti a iOS 10, basata sul lavoro di Dan Cuellar su iOS Auto. Con la deprecazione della libreria UIAutomation, tutte le versioni di iOS 10 e future sono guidate dal framework XCUITest.

Il supporto Android utilizza il framework UiAutomator per le piattaforme più recenti e Selendroid per le piattaforme Android precedenti.

Il supporto per FirefoxOS sfrutta Marionette , un driver di automazione compatibile con WebDriver e utilizzato per automatizzare le piattaforme basate su Gecko.

Versioni

Versione	Data di rilascio
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

Installazione o configurazione

Prerequisiti

Verifica i requisiti per ciascun tipo di dispositivo che desideri automatizzare e assicurati che siano installati prima di provare ad usare Appium!

Requisiti iOS

- Consigliato Mac OS X 10.10 o versione successiva, 10.11.1
- XCode> = 6.0, 7.1.1 raccomandato
- Strumenti per sviluppatori Apple (SDK simulatore iPhone, strumenti da riga di comando)
- Assicurati di leggere la documentazione su come impostarti per i test su iOS!

Requisiti Android

- API Android SDK > = 17 (Le funzionalità aggiuntive richiedono 18/19)
- Appium supporta Android su OS X, Linux e Windows. Assicurati di seguire le istruzioni per configurare correttamente il tuo ambiente per i test su diversi sistemi operativi:
    - linux
    - osx
    - finestre

Requisiti di FirefoxOS

- Firefox OS Simulator

Installazione di Appium

Installazione globale tramite Node.js

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

Installazione locale dal ramo principale di Github

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

Utilizzando l'app per Mac o Windows

- Scarica l'app Appium
- Eseguirlo!

Test di scrittura per Appium

La versione formattata dei documenti Appium può essere trovata qui con la possibilità di scegliere la lingua di esempio del codice dall'angolo in alto a destra.

Avvio di Appium per piattaforma Android e creazione di test di esempio

Impostazione dell'ambiente:

- Scarica Android SDK di livello API 17 o più
- Node.js (https://nodejs.org/)
- Software Appium (http://appium.io/)
- Barattoli di selenio (http://www.seleniumhq.org/download/)
- Barattolo di Appio (https://search.maven.org/#search%7Cga%7C1%7Cg%3Aio.appium%20a%3Ajava-client)
- .apk file dell'applicazione che deve essere testata

presupposti:

- assicurati che Eclipse sia scaricato da www.eclipse.org/downloads/
- java è installato (sia jdk che jre)
- Android SDK è installato
- Assicurati che la variabile di ambiente (Path) per Java, Android SDK, Platform e strumenti per piattaforma sia impostata.

Passi per impostare Path su Windows OS: Fare clic con il tasto destro del mouse su "Risorse del computer". "Proprietà" Sul pannello di sinistra "Impostazioni avanzate del sistema" Seleziona variabili d'ambiente Variabili di sistema-> Digita percorso-> "Percorso" doppio clic Inserisci il percorso di JAVA jdk nel tuo sistema seguito da (;) quindi percorso verso il tuo android sdk (;) percorso per la piattaforma Android (;) percorso per gli strumenti della piattaforma Android-> Fare clic su OK.

- Assicurarsi che sia installato il plug-in Eclipse

Passaggi per installare Eclipse Plug-in per Android: Avviare Eclipse, quindi selezionare Guida> Installa nuovo software. Fare clic su Aggiungi, nell'angolo in alto a destra. Nella finestra di dialogo Aggiungi repository visualizzata, inserire "ADT Plugin" per il Nome e il seguente URL per la Posizione: https://dl-ssl.google.com/android/eclipse/ Fare clic su OK (Se si riscontrano problemi nell'acquisizione il plugin, prova a utilizzare "http" nell'URL di posizione, invece di "https" (https è preferito per motivi di sicurezza).

- Assicurati che sia impostata la variabile ANDROID_HOME.

Passaggi per impostare la variabile ANDROID_HOME: Vai a Eclipse-> Finestra sul pannello superiore-> Preferenze-> Fai doppio clic su Android sul pannello sinistro Nelle preferenze Android, Copia posizione SDK Fai clic destro "Risorse del computer". "Proprietà" Nel pannello di sinistra "Impostazioni avanzate del sistema" Seleziona variabili d'ambiente In alto Variabili utente-> Seleziona nuovo-> Nome variabile, Inserisci ANDROID_HOME, Percorso variabile-> Inserisci posizione SDK copiata da Eclipse-> Fai clic su OK Quindi Variabili di sistema-> Seleziona nuovo-> Nome variabile, Inserisci ANDROID_HOME, Percorso variabile-> Inserisci

posizione SDK copiata da Eclipse-> Fai clic su OK Esci

- Assicurati che Android Virtual Device Manager possa essere avviato. Eclipse-> Finestra sul pannello superiore-> Android Virtual Device Manager-> Fare clic sul dispositivo virtuale esistente se esistente / Creane uno nuovo con configurazioni personalizzate .-> Fare clic su "Start" sul pannello di destra della finestra .-> Lanciare

Avvio di Appium:

- Installa node.js (" http://nodejs.org/ ").
- Avviare Appium dalla riga di comando dal percorso seguente: Goto Appium folder node_modules appiumbinshift + tasto destro del mouseopen prompt di comandotipo nodo appiumenter

A seguire dovrebbe essere visualizzato: info: Benvenuti in Appium v1.3.4 (REV c8c79a85fbd6870cd6fc3d66d038a115ebe22efe) informazioni: listino di interfaccia http di Appium REST avviato su 0.0.0.0:4723 informazioni: Console LogLevel: informazioni di debug: listener di interfaccia http di Appium REST avviato su 0.0.0.0: 4723info: Console LogLevel: debug

Scrivi un programma per lanciare Appium in Eclipse: pacchetto 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 []) lancia MalformedURLException {driver RemoteWebDriver; Funzionalità DesiredCapabilities = 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");



}
```

}

- Assicurati che il percorso del file apk nel sistema sia corretto

- Assicurati che il percorso del file apk nel tuo sistema sia corretto nel programma. Usa il pacchetto e l'attività corretti che possono essere trovati decompilando il file apk. Per la decompilazione del file apk, visitare http://www.decompileandroid.com .

Passaggi per lanciare appium per Android:

1. Innanzitutto avviare il server Appio al prompt dei comandi o eseguendo il file appium.exe.
2. Controlla se il dispositivo è connesso e visualizzato in adb: dispositivi adb
3. Esegui il programma su Eclipse. Il programma verrà eseguito e il file .apk installato nel dispositivo avvierà l'app.

Leggi Iniziare con Appium online: https://riptutorial.com/it/appium/topic/5122/iniziare-con-appium

Capitolo 2: Client Java

Osservazioni

API client Java

Codice sorgente client Java

Examples

Automazione Android Play Store (dispositivo reale)

Struttura del file:

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

Comando di lancio:

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>
```

Leggi Client Java online: https://riptutorial.com/it/appium/topic/6195/client-java

Capitolo 3: Test paralleli in Appium

introduzione

Esecuzione parallela in appium usando il concetto GRID del selenio. Si prega di trovare processo graduale.

Examples

Processo Step by Step

Test paralleli con Appium usando GRID: descriverò il modo in cui ha funzionato per me. Crea una griglia di selenio con Appium

1. Imposta la griglia del selenio Scarica il contenitore del server autonomo del selenio sul file system locale Apri il tuo terminale e vai alla directory in cui hai posizionato il file jar ed esegui il seguente comando:

```
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. Configurazione dei nodi Appium Qui è necessario creare i file JSON. Supponiamo di voler eseguire su due dispositivi, quindi creare due file json diversi. Ecco un file json, ho come: {"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", "porta": 4723, "maxSession": 1, "register": true, "registerCycle": 5000, "hubPort": 4444, "hubHost": "il tuo indirizzo ip"}} salva il file precedente come jasonFile1.json Qui applicationName sarà - > Il tuo cellulare-> impostazioni-> sul telefono-> Numero del modello Qui hubHost sarà il tuo indirizzo IP Qui tieni presente che devi andare come posizione predefinita per il cmd, quindi eseguire sotto il comando

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

i) Si noti che è necessario fornire il pasth assoluto del file json che si trova ii) porta come 4723 iii) Porta Bootstrap come 4724 iv) -U ad esempio ho dato come xxxx

puoi trovare l'ID del dispositivo come -> Il tuo cellulare-> impostazioni-> stato-> numero di serie Puoi anche fare "dispositivo adb" e controllare questo id dispositivo.

Quindi creerà la griglia del selenio con un dispositivo.

Ora esegui di nuovo il secondo file json e otterrai l'appium avviato Ecco il secondo file json:

{"capabilities": [{"applicationName": "Lenovo K50a40", "browserName": "Lenovo K50a40", "platformName": "ANDROID", "maxInstances": 1}], "configurazione": {"cleanUpCycle": 2000, "timeout": 30000, "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy", "host": "127.0.0.1", "port": 4730, "maxSession": 1, "registro ": true," registerCycle ": 5000," hubPort ": 4444," hubHost ":" il tuo indirizzo ip "}} salva il file precedente come jasonFile2.json

Avviare il secondo nodo con Lenovo mobile. appium --nodeconfig C: / richa / jasonFile2.json -p 4730 -bp 4731 -U xxxx

La griglia di selenio sarà simile a questa

3) Creare i metodi di esecuzione paralleli TestNG per eseguire il test.

-> Si prega di notare il valore del nome del dispositivo sarà il udid che hai fornito in precedenza. Puoi ottenerlo eseguendo dispositivi adb sul tuo prompt dei comandi.

4.

Ora crea SearchHotelTestCase.Java come di seguito: pacchetto 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 {driver AndroidDriver privato;

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

Funzionalità DesiredCapabilities = 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 = nuovo AndroidDriver (url, capacità); System.out.println ("after webdriver"); driver.manage (). timeouts (). implicitlyWait (10, TimeUnit.SECONDS); Thread.sleep (4000); }

@Test public void SearchHotel () {// Crea gli oggetti di Page Class LocaleSelectionPage localeSelectionPage = new LocaleSelectionPage (driver); SplashScreenPage splashScreenPage = new SplashScreenPage (driver); SearchLocation searchLocation = new SearchLocation (driver);

// Chiama i metodi della classe di pagina localeSelectionPage.selectLocale (); splashScreenPage.clickSplashSearchText (); searchLocation.inputSearchText ("Paris"); searchLocation.selectSuggestions ("Torre Eiffel, Parigi");

}

}

Leggi Test paralleli in Appium online: https://riptutorial.com/it/appium/topic/10016/test-paralleli-in-appium

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con Appium	BenJi, Community, Domestus, mrtuovinen, Priya, Richa Shrivastava
2	Client Java	Domestus
3	Test paralleli in Appium	Richa Shrivastava