EBook Gratis

APRENDIZAJE

appium

Free unaffiliated eBook created from
Stack Overflow contributors.

#appium

Tabla de contenido

Acerca de

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

Capítulo 1: Empezando con el appium

Observaciones

Appium es una herramienta de automatización de pruebas de código abierto y multiplataforma para aplicaciones web nativas, híbridas y móviles, probada en simuladores (iOS, FirefoxOS), emuladores (Android) y dispositivos reales (iOS, Android, FirefoxOS).

¿Por qué Appium?

1. No tiene que recompilar su aplicación o modificarla de ninguna manera, debido al uso de API de automatización estándar en todas las plataformas.
2. No tiene que recompilar su aplicación o modificarla de ninguna manera, debido al uso de API de automatización estándar en todas las plataformas. Puedes escribir pruebas con tus herramientas de desarrollo favoritas utilizando cualquier lenguaje compatible con WebDriver , como Java , Objective-C , JavaScript con Node.js (en promesa, devolución de llamada o generadores), PHP, Python , Ruby , C # , Clojure o Perl con la API de Selenium WebDriver y las bibliotecas cliente específicas del idioma.
3. Puede utilizar cualquier marco de prueba.

Invertir en el protocolo WebDriver significa que está apostando a un protocolo único, gratuito y abierto para pruebas que se ha convertido en un estándar de facto. No te encierres en una pila propietaria.

Si utiliza la biblioteca UIAutomation de Apple sin Appium, solo puede escribir pruebas con JavaScript y solo puede ejecutar pruebas a través de la aplicación Instruments. Del mismo modo, con UiAutomator de Google solo puede escribir pruebas en Java. Appium abre la posibilidad de una verdadera automatización nativa multiplataforma.

Cómo funciona

Appium controla varios marcos de automatización nativos y proporciona una API basada en el protocolo de conexión Web Jr. de WebDriver de Selenium.

Appium impulsa la biblioteca UIAutomation de Apple para versiones anteriores a iOS 10, que se basa en el trabajo de Dan Cuellar en iOS Auto. Con la desaprobación de la biblioteca de UIAutomation, todo el iOS 10 y la versión futura están controlados por el marco XCUITest.

El soporte de Android usa el marco UiAutomator para las plataformas más nuevas y Selendroid para las plataformas más antiguas de Android.

El soporte de FirefoxOS aprovecha Marionette , un controlador de automatización que es compatible con WebDriver y se usa para automatizar plataformas basadas en Gecko.

Versiones

Versión	Fecha de lanzamiento
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

Instalación o configuración

Pre requisitos

¡Compruebe los requisitos para cada tipo de dispositivo que desea automatizar y asegúrese de que estén instalados antes de intentar utilizar Appium!

Requisitos de iOS

- Mac OS X 10.10 o superior, se recomienda 10.11.1
- XCode> = 6.0, 7.1.1 recomendado
- Herramientas para desarrolladores de Apple (iPhone simulador SDK, herramientas de línea de comandos)
- ¡Asegúrese de leer la documentación sobre cómo prepararse para las pruebas de iOS!

Requisitos de Android

- Android SDK API> = 17 (las funciones adicionales requieren 18/19)
- Appium es compatible con Android en OS X, Linux y Windows. Asegúrese de seguir las instrucciones para configurar su entorno correctamente para realizar pruebas en diferentes sistemas operativos:
    - linux
    - osx
    - ventanas

Requisitos de FirefoxOS

- Firefox OS Simulator

Instalación de Appium

Instalación global utilizando Node.js

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

Instalación local desde la rama maestra de Github.

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

Usando la aplicación para Mac o Windows

- Descarga la aplicación Appium
- ¡Ejecutarlo!

Pruebas de Escritura para Appium

La versión con formato de los documentos de Appium se puede encontrar aquí con la capacidad de elegir el idioma del ejemplo de código en la esquina superior derecha.

Lanzamiento de la plataforma Appium para Android y creación de prueba de muestra.

Configuración del entorno:

- Descargar sdk de Android de nivel API 17 o más
- Node.js (https://nodejs.org/)
- Software Appium (http://appium.io/)
- Tarros de selenio (http://www.seleniumhq.org/download/)
- Tarro de Appium (https://search.maven.org/#search%7Cga%7C1%7Cg%3Aio.appium%20a%3Ajava-client)
- Archivo .apk de la aplicación que necesita ser probado

Condiciones previas:

- asegúrese de que Eclipse se descargue de www.eclipse.org/downloads/
- java está instalado (tanto jdk como jre)
- sdk de android esta instalado
- Asegúrese de que su variable de entorno (Ruta) para Java, Android SDK, Plataforma y herramientas de plataforma esté configurada.

Pasos para configurar la ruta en el sistema operativo Windows: Haga clic derecho en "Mi PC". “Propiedades” En el panel izquierdo “Configuración avanzada del sistema” Seleccione Variables de entorno Varia Variables del sistema-> Tipo de ruta-> “Ruta” haga doble clic en Ingrese la ruta a JAVA jdk en su sistema seguido de (;) luego la ruta a su android sdk (;) ruta a su plataforma android (;) ruta a las herramientas de su plataforma android-> Haga clic en Aceptar.

- Asegúrate de que Eclipse Plug-in esté instalado

Pasos para instalar Eclipse Plug-in para Android: Inicie Eclipse, luego seleccione Ayuda> Instalar nuevo software. Haga clic en Agregar, en la esquina superior derecha. En el cuadro de diálogo Agregar repositorio que aparece, ingrese "ADT Plugin" para el Nombre y la siguiente URL para la Ubicación: https://dl-ssl.google.com/android/eclipse/ Haga clic en Aceptar (si tiene problemas para adquirir el complemento, intente usar "http" en la URL de la ubicación, en lugar de "https" (https se prefiere por razones de seguridad).

- Asegúrese de que la variable ANDROID_HOME esté establecida.

Pasos para configurar la variable ANDROID_HOME: Vaya a Eclipse-> Ventana en el panel superior-> Preferencias-> Haga doble clic en Android en el panel izquierdo En las preferencias de Android, copie la ubicación del SDK Haga clic derecho en “Mi computadora”. “Propiedades” En el panel izquierdo “Configuración avanzada del sistema” Seleccione Variables de entorno En la parte superior Variables de usuario-> Seleccione nuevo-> Nombre de variable, ingrese

ANDROID_HOME, ruta de variable-> Ingrese la ubicación del SDK copiado desde Eclipse-> Haga clic en Aceptar Luego, Variables del sistema-> Seleccionar nuevo-> Nombre de variable, ingrese ANDROID_HOME, Ruta variable-> Ingrese la ubicación del SDK copiado desde Eclipse-> Haga clic en Aceptar Salir

- Asegúrese de que se pueda iniciar el Administrador de dispositivos virtuales de Android. Eclipse-> Ventana en el panel superior-> Administrador de dispositivos virtuales de Android-> Haga clic en el dispositivo virtual existente si existe / Cree uno nuevo con configuraciones personalizadas .-> Haga clic en “Inicio” en el panel derecho de la ventana .-> Lanzamiento

Lanzamiento de Appium:

- Instale node.js (" http://nodejs.org/ ").
- Inicie Appium desde la línea de comando desde la siguiente ubicación: Ir a la carpeta Appium node_modules appiumbinshift + right clickopen command prompttype node appium enter

Se debe mostrar lo siguiente: información: Bienvenido a Appium v1.3.4 (REV c8c79a85fbd6870cd6fc3d66d038a115ebe22efe) información: Appium REST http interface listener comenzó en 0.0.0.0:4723 información: Console LogLevel: debug info: Appium REST http interface listener comenzó en 0.0.0.0.0.04.03 4723info: Console LogLevel: debug

Escriba un programa para lanzar Appium en Eclipse: package appium.com;

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

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

la clase pública AppiumLaunch {public static void main (String args []) lanza MalformedURLException {controlador RemoteWebDriver; Capacidades de DesiredCapabilities = nuevas 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");



}
```

}

- Asegúrese de que la ruta del archivo apk en el sistema es correcta
- Asegúrese de que la ruta al archivo apk en su sistema sea correcta en el programa. Use el paquete y la actividad correctos que se pueden encontrar al descompilar el archivo apk. Para descompilar el archivo apk, vaya a http://www.decompileandroid.com .

Pasos para lanzar Appium para Android:

1. Primero inicie el servidor appium en el símbolo del sistema o ejecutando el archivo appium.exe.
2. Compruebe si el dispositivo está conectado y se muestra en dispositivos adb: adb
3. Ejecutar el programa en el eclipse. El programa se ejecutará y el archivo .apk que se instaló en el dispositivo iniciará la aplicación.

Lea Empezando con el appium en línea: https://riptutorial.com/es/appium/topic/5122/empezando-con-el-appium

Capítulo 2: Cliente Java

Observaciones

API de cliente de Java

Código fuente del cliente Java

Examples

Android Play Store automatización (dispositivo real)

Estructura del archivo:

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

Comando de lanzamiento:

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>
```

Lea Cliente Java en línea: https://riptutorial.com/es/appium/topic/6195/cliente-java

Capítulo 3: Pruebas paralelas en Appium

Introducción

Ejecución paralela en appium utilizando concepto GRID de selenio. Por favor encuentre el proceso paso a paso.

Examples

Proceso paso a paso

Pruebas paralelas con Appium usando GRID: describiré la forma en que funcionó para mí. Crea Grilla De Selenio Con Appium

1. Configure la cuadrícula de selenio Descargue el servidor independiente de Selenium jar en el sistema de archivos local Abra su terminal y navegue al directorio donde colocó el archivo jar y ejecute el siguiente 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. Configure los nodos de Appium Aquí tiene que crear los archivos json. Supongamos que desea ejecutar en dos dispositivos y luego crear dos archivos json diferentes. Aquí hay un archivo json, tengo como: {"capacidades": [{"applicationName": "ONEPLUS A3003", "browserName": "ONEPLUS A3003", "platformName": "ANDROID", "maxInstances": 1}] , "configuración": {"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"}} guarde el archivo anterior como jasonFile1.json Aquí applicationName será: > Su Móvil-> configuración-> sobre el teléfono-> Número de modelo Aquí hubHost será su dirección IP Aquí tenga en cuenta que debe ir como ubicación predeterminada de cmd y luego ejecute el siguiente comando

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

i) Tenga en cuenta que debe proporcionar el pasaje absoluto del archivo json ubicado ii) puerto como 4723 iii) Bootstrap puerto como 4724 iv) -U, por ejemplo, lo he dado como xxxx

puede encontrar la identificación del dispositivo como -> Su Móvil-> configuraciones-> estado-> Número de serie También puede hacer "adb device" y verificar esta identificación del dispositivo.

Luego creará la cuadrícula de selenio con un dispositivo.

Ahora vuelva a ejecutar el segundo archivo json y obtendrá el inicio de appium. Aquí está el segundo archivo json:

{"capacidades": [{"applicationName": "Lenovo K50a40", "browserName": "Lenovo K50a40", "platformName": "ANDROID", "maxInstances": 1}], "configuración": {"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 ":" your ip address "}} guarda el archivo anterior como jasonFile2.json

Iniciar el segundo nodo con Lenovo móvil. appium --nodeconfig C: / richa / jasonFile2.json -p 4730 -bp 4731 -U xxxx

Selenium Grid se verá así

3) Cree los métodos de ejecución paralelos de TestNG para ejecutar su prueba.

-> Tenga en cuenta que el valor del nombre del dispositivo será el udid que proporcionó anteriormente. Puede obtenerlo ejecutando dispositivos adb en el símbolo del sistema.

4.

Ahora cree SearchHotelTestCase.Java como se indica a continuación: package com.trivago.TestCases;

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

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

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

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

clase pública SearchHotelTestCase {controlador privado de AndroidDriver;

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

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

URL url = new URL (" http://0.0.0.0:4723/wd/hub/ "); System.out.println ("before webdriver"); controlador = nuevo AndroidDriver (url, capacidades); System.out.println ("after webdriver");

driver.manage (). timeouts (). implitlyWait (10, TimeUnit.SECONDS); Thread.sleep (4000); }

@Test public void SearchHotel () {// Cree los objetos de Page Class LocaleSelectionPage localeSelectionPage = new LocaleSelectionPage (driver); SplashScreenPage splashScreenPage = new SplashScreenPage (controlador); SearchLocation searchLocation = new SearchLocation (controlador);

// Llame a los métodos de la clase de página localeSelectionPage.selectLocale (); splashScreenPage.clickSplashSearchText (); searchLocation.inputSearchText ("París"); searchLocation.selectSuggestions ("Torre Eiffel, París");

}

}

Lea Pruebas paralelas en Appium en línea: https://riptutorial.com/es/appium/topic/10016/pruebas-paralelas-en-appium

Creditos

S. No	Capítulos	Contributors
1	Empezando con el appium	BenJi, Community, Domestus, mrtuovinen, Priya, Richa Shrivastava
2	Cliente Java	Domestus
3	Pruebas paralelas en Appium	Richa Shrivastava