* APPIUM *

9/4/19

Appium is a automation tool used to perform mobile testing in a automation approach. The main advantages of appium are,

1. It is a open source tool.
2. It can be used to test mobile applications both in andriod devices as well as ios device
3. We can able to use different types of programming languages to implement the scripts.
   Ex: Java, phython, C#.net. etc.
4. We can able to test different types of applic such as - Native application
   - Web application
   - hybaid applications.

Native applications:-

These are the applications that will come along with the device. These are specific to that device.

Ex: calculator, camera, etc.

Web applications:-

These are the websites that can be opened in the mobile device by using a browser.

Hybrid Applications :-

These are the applications that can be used as websites as well as mobile apps. They can be opened in different types of devices.

Ex: facebook.

Disadvantages of Appium :-

1) Using Appium, we can test the ~~(mobile)~~ applications in the mobile of Andriod type only when the andriod API version is greater than 17.
the old andriod devices can't be tested by appium.

2) In andriod devices web applications can be tested by using chrome browser. And in ios devices we can test by using Safari browser only.

3) Toast messages can't be tested.

Note: A Toast message is a short message displayed to the user only for a shorter period of time, for example 2 seconds. After that time those messages will be removed automatically.

→ Mobile

tools that

→ Appium

Ex:

1. clien
in

2. This
for

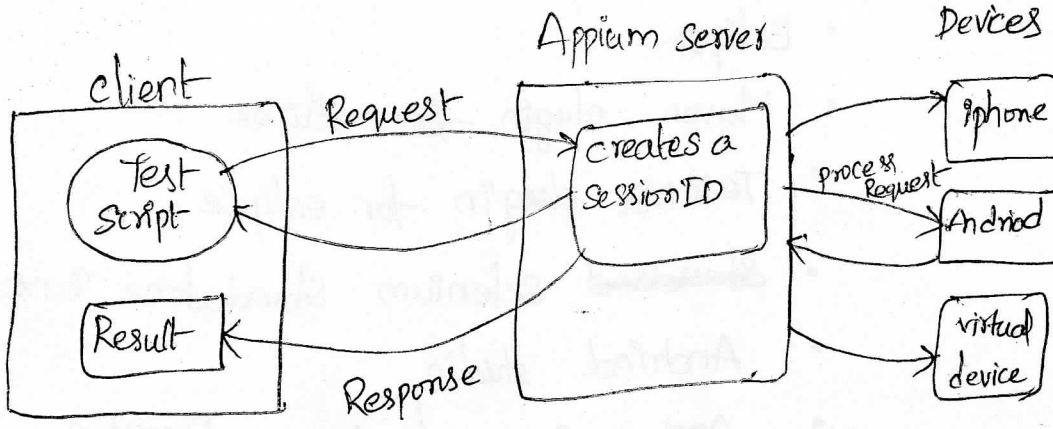
3. After
will

→ Mobile automation tools :-

The following are the different automation tools that can be used for mobile testing.

- Appium
- Selendriod
- ios-driver
- calabash
- Money Talk
- Frank
- KIF (Keep it functional)
- Robotium.

→ Appium Architecture :-



Ex: Eclipse
IntelliJ etc.

1. client will send a request to the appium server in the form of a desired capability.
2. This request uses a Jason wired protocol for communication.
3. After recieving a request, the appium server will create a session ID for that client.

4. This sessionID will be used for the future communication between the client and appium server.
5. The appium server is similar to a WebDriver. It executes the request given by the client on the device.
6. The response will be send back to the client with the help of sessionID.

→ Softwares required :-

The following are the different types of s/ws that are necessary to perform mobile testing using appium.

- Java
- Eclipse
- Maven plugin for eclipse
- TestNG plugin for eclipse
- ~~standard~~ selenium standalone server
- Andriod studio
- Appium server / Appium Desktop
- Appium client library.

→ Downloading & installing Andriod studio :-

Andriod studio is an IDE where we can able to develop an andriod application as well as test that application.

Once we install this andriod studio, internally some tools will be installed automatically.

They a

(ii) AVD

(iii) UI A

It

1. Open
http://
2. click
3. Accept
4. click o
5. After
file.
Ex: a
6 In the
☑ Andriod
7. Follow
process
8. Oper
9. Select
10. Select
11. click

They are, (i) SDK manager – used to install android api software.

(ii) AVD manager – Android Virtual Device (AVD) used to create android virtual device

(iii) UI Automate viewer – used to inspect elements in android devices applications.

It can be installed as follows,

1. Open a browser with the following URL.
   http://developer.android.com/studio.
2. Click on [Download Android Studio] button.
3. Accept the license agreement.
4. Click on [Download Android Studio for windows] button
5. After completion of downloading, open the executable file.
   Ex: android-studio-ide-183.5452501-windows.exe.
6. In the welcome screen displayed select
   ☑ Android virtual Device checkbox.
7. Follow the steps and finish the installation process.
8. Oper the Android studio.
9. Select the install Type as ☑ standard.
10. Select UI Theme as ☑ Dacula / IntelliJ ☑
11. Click on the [finish] button.

→ Types of Devices :-

We can make use of different types of devices to test an application. They are,

1) Real device
2) Emulator / Simulator.

- Real Devices are those actual devices where we use the applications to be tested.
- An andriod type of virtual device is called as Emulator.
- An ios type of virtual device is called as Simulator.

→ Creating a Emulator :-

In order to create an andriod virtual device called emulator, we have to perform the following activities.

1. Creat, New andriod project.
2. Open AVD manager.
3. Create an andriod device

Ex: - Open Andriod studio.
- Select File → New → New project

4. Select Add no Activity option & click Next.

Configure the project as follows,

Name [My Application]

Language [Java]

Minimum API level [API 28 : Andriod 9.0 (Pie)]

↑ API of which

5. Click on [Finish] button.

6. In the andriod studio click on AVD Manager icon () or in Tools menu
   ↳ AVD manager

7. click on [create Virtual Device] button in device mana

8. In the 'choose a device' window select the device typ
   Ex: Nexus S

9. click [Next] button.

10. Select the andriod version and click on [Next].

11. In the configuration window, provide a name to the dev

12. Click on [Finish] button.

13. In the newly created device click on launch (▶) ic
    under Actions.

14. Once the device is created Open
    Settings
    ↳ System
    ↳ About emulated device/phone
    ↳ navigate to the 'Build number'
    ↳ Keep on click on it until developeropti is enabled.
    ↳ click on back button
    ↳ And verify inside the developer option 'USB Debugging' is on/not

→ Creating Environmental Variables :-

In order to run some commands from the command prompt we have to create some set of environmental variables for the android tools

1. Open the environmental variables
2. Under System variables section, click on new button
3. Specify the variable details as follows.

Variable Name: ANDROID_HOME

Variable Value: c:\\ user\ Ramya \AppData\local\Android\sdk

4. click ok button.
5. Under system variable click on path variable,
6. click on edit button.
7. click on New button and enter the following paths

% ANDROID_HOME %\bin

% ANDROID_HOME %\tools

% ANDROID_HOME %\platform_tools\bin

% ANDROID_HOME %\sdk\build_tools\28.0.3

→ Finding the devices connected :-

In order to get the list of devices connected to the system, in the command prompt execute the following command,

> adb devices ↵

→ Downloa

1. Open
2. In the
3. Under

Appium

4. After
the fi
the ins

→ Working

In order
the appi

Inspector
perform
(1) Install
(2) Open
(3) Inspec

Installi

will have
android
(apk - a
mobile
file to

Ex:

→ Downloading and installing Appium server :-

1. Open a browser with URL http://appium.io/
2. In the window displayed click on Download Appium
3. Under the latest version click on the link Appium_windows_1.12.1.exe
4. After completion of downloading the file, open the file and follow the instructions to complete the installation process.

→ Working with real device from the appium server:

Inorder to perform any test on the mobile device the appium server should run in the background

From appium server, we can start inspector session. By using this session we can perform the following activities

(1) Install an application in the mobile device.
(2) Open a browser in the device.
(3) Inspect the element to know their details.

Installing an app:-

In general the implemented softwares will have .exe extension. Where as in android the applications will have .apk extension (apk - android application package). In the mobile devices directly we can run this apk file to start install the application.

Ex:

1. Download some apk file from the website apkpure.
   Ex: inshorts.apk.
2. Start the appium server.
3. click on start Inspector Session icon.
4. Specify the Desired capabilities as follows,

platformName	text	Android
device Name	text	ba23d4f41 (device name of mine)
app	text	c:\inshorts.apk

5. click on Start Session button and observe that app Installation will begin in the device.

Ex2:

To open a browser in the device Specify the desired capabilities for the session as follows.

platformName	text	Android
device Name	text	ba23d4f41
browserName	text	chrome

Ex3: Inspecting elements.

1. Start a session by opening a browser
2. After open the browser in the device, open a web site example ebay.com
3. In the appium session window click on Refresh icon.

of the
5. click
(we get
6. Observe
details

→ Mobile

from se
plugin
server lib

Seleniu
from the
web drive

→ Downloa

1. Op
dowr

2. click o
3. In th
downl
4. Inord
project
mvn repo

4. In the left side pannel we can see the screensho of the mobile.
5. click on the element by highlighting the element (we get yellow colour).
6. Observe that in the right side pannel elements details are displaying.

→ Mobile testing using selenium :-

In order to perform mobile application testing from selenium along with maven plugin, testNG plugin we must have selenium standard alone server library and appium client library.

selenium standalone server libarary can be downloaded from the same place where we downloaded selenium web driver.

→ Downloading Appium client library :-

1. Open the appium offical website (appium.io) download section by using URL
   http:// appium.io / downloads.html
2. click on Related language example java.
3. In the next window displayed, click on download icon (⇩)
4. Inorder to get dependency information for maven project search for ~~appium~~ java client library in the mvnrepository.com

Ex: Testing 'LG' product search in ebay.com site using selenium Webdriver in a real mobile device. (Appium web application testing).

1. Start a new maven project.
2. In the pom.xml file add the dependencies for java-client, commons-lang3 & testng.
3. In the src/test/java folder, create a new package.
4. Inside the package create a java class / testng class.
5. Implement java's main method (or) @Test method as follows,

```
main() throws Exception
{
    DesiredCapabilities dc = new DesiredCapabilities();
    dc.setCapability("deviceName", "baz3d4f41");
    dc.setCapability("platformName", "Android");
    dc.setCapability("browserName", "chrome");
    URL url = new URL("http://0.0.0.0:4723/
                       wd/hub");
    AppiumDriver<MobileElement> driver = new
                AppiumDriver<MobileElement>(url, dc);
    Thread.sleep(10000);
    driver.get("http://ebay.com");
    driver.findElement(By.id("kw")).sendKeys("LG");
    driver.findElement(By.id("ghs-submit")).click();
}
```

we have
appPackage
can be
① Using
② Usi

① Using
- Ins
the
- Th
disp
- For
nam
- clic
- A li
whe

② Using
- In
inform
- open
com
- We

→ Testing a mobile application:-

In order to test a mobile application we have to specify the desired capabilities about appPackage & appActivity. This application information can be retrieved in 2 ways.

① Using APK Info app.

② Using adb shell command.

① Using APKInfo app:-

- Install APKINFO app from the playstore open the app.
- The list of apps installed in the mobile will be displayed
- For every app, App name below to it appPackage name will be displayed.
- click on the App & click on activities.
- A list of activities will be displayed from where we can identify main Activity.

② Using adb shell command:-

- In the mobile device, open the app whose information should be retrieved.
- open the command prompt & type the following command > adb shell ↵
- Write the following shell command.

~~dumpsys window windows | grep -E 'mCurrentFocus'~~ ↵

$ dumpsys window windows | grep -E 'mCurrentFocus' ↵

- Displays appPackage, appActivity.

Appium Desired Capabilities

Desired Capabilities are keys and values encoded in a JSON object, sent by Appium clients to the server when a new automation session is requested. They tell the Appium drivers all kinds of important things about how you want your test to work. Each Appium client builds capabilities in a way specific to the client's language, but at the end of the day, they are sent over to Appium as JSON objects.

Some important capabilities are demonstrated in the following example:

```
{
       "platformName": "Android",
       "platformVersion": "9.0",
       "deviceName": "ba5541",
       "app": "/path/to/myapp.apk"
}
Ex:
```

1. Write a selenium script to install an app “inshorts.apk” in the real device**[App install Test]**

```
Package AppiumDemos;
import org.openqa.selenium.By;
import org.openqa.selenium.remote.DesiredCapabilities;

import io.appium.java_client.AppiumDriver;
import io.appium.java_client.MobileElement;

public class appTest
{
  public static void main(String[] args) throws Exception
  {
      DesiredCapabilities dc = new DesiredCapabilities();

      dc.setCapability("deviceName", "ba23d441");

      dc.setCapability("platformName", "Android");

      dc.setCapability("browserName", "chrome");

      dc.setCapability(MobileCapabilityType.APP, "c:\\selsoft\\inshorts.apk");
      URL url = new URL("http://0.0.0.0:4723/wd/hub");
      AppiumDriver<MobileElement> driver = new AppiumDriver<MobileElement>(url, dc);
    }
}
```

Ex: Write a Selenium script to signup for apsrtc online app.[HybridApp Test]

1. Connect the real device to the system
2. Open apsrtc app
3. Open command prompt
4. Type >adb shell
5. $ dumpsys window windows | grep –E 'mCurrentFocus'
6. Copy the 'appPackage' and 'appActivity' information.

```
package AppiumDemos;

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

import org.openqa.selenium.By;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.testng.annotations.Test;

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

public class apsrtcTest {
  @Test
  public void signup() throws Exception {
        DesiredCapabilities dc = new DesiredCapabilities();

      dc.setCapability("deviceName", "ba23d441");
      dc.setCapability("platformName", "Android");

      dc.setCapability("appPackage", "com.apsrtc.online");
        dc.setCapability("appActivity",
"com.abhibus.app.apsrtc.DashBoardActivity");

        AppiumDriver<MobileElement> driver = new
AndroidDriver<MobileElement>(new URL("http://0.0.0.0:4723/wd/hub"),
dc);
        driver.manage().timeouts().implicitlyWait(15,
TimeUnit.SECONDS);

        driver.findElement(By.xpath(
"//android.widget.TextView[@text='Book Ticket']")).click();
            driver.findElement(By.xpath(
"//android.view.View[@text='Login']")).click();
            driver.findElement(By.xpath(
"//android.view.View[@text='SignUp']")).click();
            driver.findElement(By.xpath(
"//android.widget.EditText[@resource-id='loginName']")).sendKeys("scott98");
```

```
      driver.findElement(By.xpath("//android.widget.EditText[@resour
ce-id='fullName']")).sendKeys("scotttiger");

      driver.findElement(By.xpath("//android.widget.EditText[@resour
ce-id='email']")).sendKeys("scotttiger@tiger.com");

      driver.findElement(By.xpath("//android.widget.EditText[@resour
ce-id='mobileNo']")).sendKeys("9112345678");
            Thread.sleep(10000);
  }
}
```

Ex: Write a Selenium script to test facebook signin process[HybridApp Test]

```
package AppiumDemos;

import java.net.URL;

import org.openqa.selenium.WebElement;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.testng.Assert;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

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

public class facebookTest {
  @Test
  public void login() throws Exception{
      AndroidDriver<WebElement> driver = null;

      String appiumURL = "http://0.0.0.0:4723/wd/hub";

      DesiredCapabilities capabilities = new DesiredCapabilities();

      capabilities.setCapability("deviceName", "ba23d441");

      capabilities.setCapability("browserName", "chrome");

      capabilities.setCapability("platformName", "android");

      driver = new AndroidDriver<WebElement>(new URL(appiumURL),
capabilities);
      String expectedTitle = "Facebook – log in or sign up";
```

```
      driver.get("https://www.facebook.com/");
      String actualTitle = driver.getTitle();
      Assert.assertEquals(actualTitle, expectedTitle);
MobileElement email= (MobileElement) driver.findElementById(
                                    "m_login_email");
      email.sendKeys("selenium.venkatbj@gmail.com");
MobileElement password = (MobileElement) driver.findElementById(
                                    "m_login_password");
      password.sendKeys("password");
      email.clear();
      password.clear();
      driver.findElementById("signup-button").click();
      String actualMessage = driver.findElementById("m-future-page-
header-title").getText();
      Assert.assertEquals(actualMessage, "Join Facebook");
      Thread.sleep(3000);
  }
}
```