Introduction to Selenium + Appium

Testing an existing web site

Step 1: Set Up Prerequisites

Before you begin, ensure you have the following installed:

- Java Development Kit (JDK): Download and install the latest version of the JDK. Set up the JAVA_HOME and update the system Path variable to include the JDK's bin directory.
- Android Studio: Install Android Studio to get the Android SDK and set up virtual devices (emulators) if needed. Set the ANDROID_HOME environment variable to your SDK location, and update the Path variable to include the platform-tools directory.
- Node.js and Appium: Install Node.js from its official website. Then, install Appium using npm with npm install -g appium. Optionally, download the Appium Desktop client for a GUI interface.
- Appium Java Client: Add the Appium Java client library to your project if you're using Java. For Maven projects, include it in your pom.xml.

Step 2: Start the Appium Server

- Via Command Line: Open a terminal and run appium. This starts the Appium server on the default port (4723).
- Via Appium Desktop: Launch Appium Desktop and start the server through the GUI.

Step 3: Configure DesiredCapabilities for Web Testing

You'll need to specify DesiredCapabilities to tell Appium which device and browser to use for testing. Here's an example for testing with Chrome on an Android device:

```
DesiredCapabilities caps = new DesiredCapabilities();
caps.setCapability("platformName", "Android");
caps.setCapability("deviceName", "Android Device"); // Use "Android Emulator" if testing on an emulator
caps.setCapability("browserName", "Chrome");
caps.setCapability("automationName", "UiAutomator2"); // Recommended for Android
```

If testing on a real device, replace "Android Device" with your device name or udid. The deviceName is not strictly enforced by Appium for web testing, but it's good to specify it for clarity.

Step 4: Create a Test Script

Here's a simple Java test script using Selenium WebDriver to open a website:

```
import io.appium.java_client.AppiumDriver;
import io.appium.java_client.MobileElement;
import io.appium.java_client.android.AndroidDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import java.net.URL;

public class MobileWebTest {
  public static void main(String[] args) {
    try {
      DesiredCapabilities capabilities = new DesiredCapabilities();
```

```
            capabilities.setCapability("platformName", "Android");
            capabilities.setCapability("deviceName", "Android Device");
            capabilities.setCapability("browserName", "Chrome");
            capabilities.setCapability("automationName", "UiAutomator2");

            AppiumDriver<MobileElement> driver = new AndroidDriver<>(new
URL("http://0.0.0.0:4723/wd/hub"), capabilities);

            // Navigate to a web page
            driver.get("https://www.example.com");

            // Add your test steps here

            // Quit the driver
            driver.quit();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Step 5: Run Your Test
Compile and run your test script through your IDE or command line. The script should initiate a session with the Appium server, which in turn communicates with the specified Android device to launch Chrome and navigate to the specified URL.

Additional Steps for Real Devices
If testing on a real device, ensure:

- The device is connected via USB and detected by your system (adb devices should list it).
- USB debugging is enabled on the device (in Developer Options).
- Chrome is updated to the latest version on the device, as Appium will interact with it for web testing.

Testing a mobile application

Step 1: Install Prerequisites
Before you start with Appium, ensure you have the following installed:

- Java Development Kit (JDK): Java is commonly used for writing test scripts.
- Node.js and npm (Node Package Manager): Appium is a Node.js application.
- Android Studio: This provides you with the Android SDK tools necessary for Appium.
- Appium: The core of what we'll be using for automation.

Step 2: Install Appium
You can install Appium either via npm (recommended for command-line usage) or as a desktop application (which provides a GUI and inspector tools).

- npm installation:

```
npm install -g appium
```

- Appium Desktop: Download and install from the Appium website.

Step 3: Set Up Android Environment
Install Android Studio: Download from the official site and follow the installation instructions.

Configure Environment Variables:

- ANDROID_HOME: Set this to your Android SDK location. This is typically under C:\Users\<Your User Name>\AppData\Local\Android\Sdk on Windows or ~/Library/Android/sdk on macOS.
- Update your PATH: Include paths to the platform-tools and tools folder within the SDK.

Step 4: Create an Android Virtual Device (AVD)

- Open Android Studio and go to the AVD Manager.
- Create a new virtual device, selecting a device definition and a system image (e.g., Pie, API Level 28).
- Finish and create the AVD.

Step 5: Install Appium Doctor
Appium Doctor checks your system for configuration issues. Install it via npm:

```
npm install -g appium-doctor
```

Run it to verify your setup:

```
appium-doctor --android
```

Step 6: Writing Your First Test
You'll need a testing framework. For Java, you can use TestNG or JUnit. Here's an example using Java with TestNG:

Create a Maven Project: In an IDE like IntelliJ IDEA or Eclipse, create a new Maven project.

Add Dependencies: Your pom.xml file should include dependencies for Selenium, Appium, and the testing framework.

```
<dependencies>
  <dependency>
    <groupId>io.appium</groupId>
    <artifactId>java-client</artifactId>
    <version>7.3.0</version>
  </dependency>
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
  </dependency>
  <!-- Include dependencies for your testing framework here -->
</dependencies>
```

Write Your Test:

- Instantiate an AndroidDriver object.
- Set desired capabilities for the Appium server to know what app and device to run the test on.
- Use the driver to interact with elements in your app.

Here's a simple example to open an app:

```
import io.appium.java_client.AppiumDriver;
import io.appium.java_client.MobileElement;
import io.appium.java_client.android.AndroidDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import java.net.URL;

public class FirstTest {
  public static void main(String[] args) {
    DesiredCapabilities caps = new DesiredCapabilities();
    caps.setCapability("deviceName", "Your AVD Name");
    caps.setCapability("platformName", "Android");
    caps.setCapability("appPackage", "Your App Package");
    caps.setCapability("appActivity", "Your App Activity");

    AppiumDriver<MobileElement> driver;
    try {
      driver = new AndroidDriver<>(new URL("http://127.0.0.1:4723/wd/hub"), caps);
      // Your test code here
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Step 7: Start Appium Server and Run Your Test

Start the Appium Server:

- If using the command line, run appium.
- If using Appium Desktop, start the server through the GUI.
- Run Your Test: Execute your test script from your IDE or the command line, depending on your setup.