Globant

appium

Introduction to APPIUM

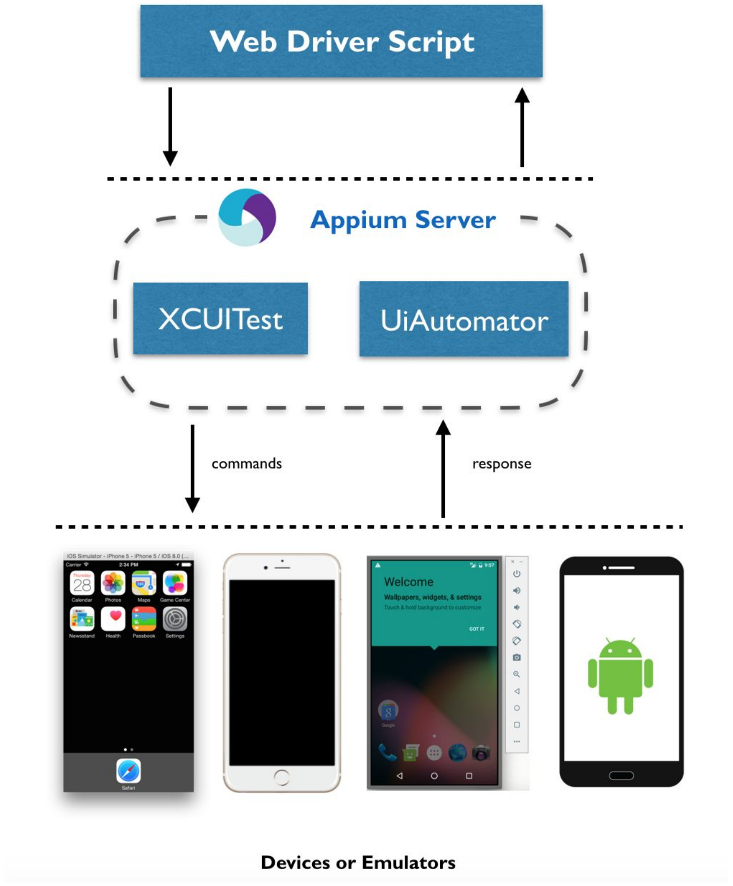
Appium is an open source tool to automate native apps, iOS hybrid apps, Android and Windows based platforms as well.

Native apps are those apps which are written using iOS, Android and Windows SDKs. Web applications are those that are accessed through a browser (Appium supports Safari on iOS and Chrome or the preloaded browser on Android).

Mobile Automation
APPIUM

Introduction to APPIUM

General Architecture



Appium philosophy

Appium Challenges

Appium was designed to solve needs of automating applications based on the following principles:

1. You should not recompile your app or modify it in any way to try to automate it.

2. You shouldn't restrict yourself to a specific language or framework for writing and running your tests.

Appium Challenges

3. A mobile automation framework shouldn't reinvent the wheel.

4. A mobile automation framework should be open source, in spirit, and also in practical

Appium Solutions

Appium Solutions

- ***You should not recompile your app or modify it in any way to try to automate it.***

Solution: Using the automation frameworks provided by the providers. In this sense, it is not required to compile in the development code or through third parties. This means that the same app that is generated is tested. The frameworks supported by appium are:

Globant

Appium Solutions

- iOS 9.3 and above: Apple's XCUITest
- iOS 9.3 and lower: Apple's UIAutomation
- Android 4.2+: Google's UiAutomator / UiAutomator2
- Android 2.3+: Google's Instrumentation.
- Windows: Microsoft's WinAppDriver

Appium Solutions

- ***You shouldn't restrict yourself to a specific language or framework for writing and running your tests.***

Solution: Wrapping providers' frameworks through an API: “WebDriver”. WebDriver also known as "Selenium WebDriver" specifies a client-server protocol known as JSON Wire Protocol.

Given this client-server architecture, a client written in any language can be used to send the appropriate HTTP requests to the server. There are clients written in the most popular languages (Java, Javascript, Ruby, Python etc).

Appium Solutions

- ***A mobile automation framework shouldn't reinvent the wheel.***

Solution: WebDriver has become the default standard for web automation. Why do something totally different for mobile? Instead, the protocol has been extended with useful API methods for mobile automation.

Appium Solutions

- ***A mobile automation framework should be open source, in spirit and also in practical.***

Solution: Appium is open source.

Appium Concepts

Client/Server Architecture

Appium is at its heart a web server that exposes a REST API.

It receives connections from a client, listens for commands, executes these commands on a mobile device, and responds with an HTTP response that represents the result of executing the command.

With a client/server architecture we can write our test code in any language that has an http client API, but it is easier to use one of the Appium client libraries.

Desired Capabilities

The “desired capabilities” are a set of keys and values that are sent to the Appium server to tell the server what type of automation session we are interested in starting.

There are also various capabilities that can modify server behavior during automation.

For example, we could set the **PlatformName** capability on iOS to tell Appium that we want an iOS session, rather than an Android session.

Or we could set the **safariAllowPopups** capability to true to ensure that, during a Safari automation session, we can use JavaScript to open new windows.

Desired Capabilities

Capability	Description	Values
automationName	Which automation engine to use	Appium (default) or Selendroid or UiAutomator2 or Espresso for Android or XCUITest for iOS or YouiEngine for application built with You.i Engine
platformName	Which mobile OS platform to use	iOS, Android, or FirefoxOS

Desired Capabilities

Capability	Description	Values
platformVersion	Mobile OS version	e.g., 7.1, 4.4
deviceName	The kind of mobile device or emulator to use	iPhone 7, Android Emulator, Galaxy S4, etc....
app	The absolute local path *or* remote http URL to a .ipa file (IOS), .app folder (IOS Simulator), .apk file (Android) or .apks file (Android App Bundle),	/abs/path/to/my.apk or http://myapp.com/app.ipa

Android Capabilities

Capability	Description	Values
appActivity	Activity name for the Android activity you want to launch from your package.	MainActivity, .Settings
appPackage	Java package of the Android app you want to run. By default this capability is received from the package manifest (@package attribute value)	com.android.settings

iOS Capabilities

Capability	Description	Values
udid	Unique device identifier of the connected physical device	e.g. 1ae203187fc012g

Globant

Session

Automation is always done in a session context.

Clients log in to a server specifically for each library, but they all end up sending a POST session request to the server, with a JSON object specifying the "desired capabilities".

At this point, the server will start the automation session and respond with a session ID that is used to send more commands.

THANKS FOR
YOUR ATTENTION!