Building Automated Android App Tests

@Tom Chavez
tchavez@soasta.com
Sr. Developer Evangelist

Agenda

- Why Automation?
- Test Automation Products
- Getting Started
- Creating Test Scripts from Code
- Creating Test Scripts by Recording
- Running Automated Test Scripts
  - *On local devices, in cloud devices*
- Setting Up Continuous Integration

Why Automation?

- Improved testing efficiency
- Consistent and repeatable testing process
- Improved regression testing coverage
- More tests can be run in less time
- Run 24x7
- Humans free to perform advanced manual tests
- Easy to reproduce defects

Where Are You Now?

- Survey Question 1:
  - *What kind of testing are you doing now?*
    - Doing some automated testing with manual
    - Doing mostly automated testing with manual
    - Doing only automated testing
- Survey Question 2:
  - *How are you automating your tests?*
    - Developers write tests as code
    - QA team records tests from manual scripts
- Survey Question 3:
  - *Are you satisfied with your test automation process?*
    - Yes. Why are you here?

Test Automation Products

- SOASTA TouchTest
- Applause
- Ranorex
- Appium/Selenium and SauceLabs
- Robotium
- Perfecto Mobile
- Device Anywhere
- MobileLabs
- mAutomate
- Experitest (formerly SeeTest)
- SmartBear
- NTT AppKitBox Remote Test Kit
- MonkeyTalk

Things to Consider in Test Automation

- Is your QA team good at manual testing or writing test code in a high-level language (Java, JavaScript, etc.)?
- Do you want one tool for Android and iOS?
- Does your company prefer open source (free) or commercial products?
- Do you want tools that install locally or are SaaS/cloud-based?
- Is your team located together or multi-site?

Which Approach to Choose?

- Record and Playback
  - *Easiest approach to start mobile test automation*
  - *Script parameterization*
- Code-based

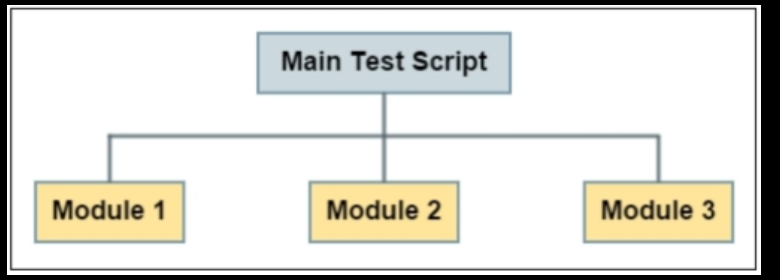
Getting Started

- What you need for each:
- TT vs. Appium
- Mac, TTLite, downloads
- Mac, Ruby, ADK, …

Test Frameworks

- Types of test frameworks
  - *Functional decomposition or test script*
  - *Data driven*
  - *Keyword driven*
  - *Hybrid*

Functional Decomposition / Test Script



- Void fn_gotoEmailAccount() {
  'Open Browser and navigate to email application URL
  WebDriver driver = new InternetExplorerDriver();
  driver.get(" https:// testingemail.sit.com");
  'Page Sync
  driver.manage(). timeouts(). implicitWait(10, TimeUnit.SECONDS);
  'Click on create account
  driver.findElement(By.linkText(" Create Account")). click();
  }

  'Enter the details
  void fn_EnterDetails() {
  driver.findElement(By.name(" First Name")). sendKeys(" TestUserID1");
  driver.findElement(By.name(" Last Name")). sendKeys(" TestUserPwd123");
  }

  void fn_SubmitToCreate() {
  'Submit driver.findElement(By.name(" Next Step")). click();

Test Script

- 'Test Script 1: To validate that application Login is allowed with valid credentials:
- 'Open Email
  fn_gotoTestingEMailAccount();

  'Enter the details
  fn_EnterDetails();

  'Submit
  fn_SubmitToCreate();

Data Driven Framework

- Test input and output values are stared externally
  - *CSV, ODBC, Excel, etc.*
- Script manages reading data files, loggin, navigating through programs
- Can test via parameterization, e.g. for logging in:
  - *Valid username, valid password*
  - *Valid username, wrong password*
  - *Valid username, no password*
  - *Non-valid username, valid password*
  - *Non-valid username, wrong password*
  - *Non-valid username, no password*

Keyword Driven Framework

- Driven by a test script = driver script
- For example for a banking app:

Script ID	Keyword 1	Keyword 2	Keyword 3	Keyworld 4
TC_1	Login	Account_Balance_Check	Fund_Transfer	Logout
TC_2	Login	Fund_Transfer	Account_Balance_Check	Logout
TC_3	Login	Account_Balance_Check	Utility_Bill_Pay	Logout
TC_4	Login	Utility_Bill_Pay	Logout	

Hybrid Framework

- Combination of Keyword and Data-driven frameworks

Script ID	Device ID	Keyword 1	Keyword 2	Keyword 3	Keyworld 4
TC_1	D1	Login	Account_Balance_Check	Fund_Transfer	Logout
TC_2	D2	Login	Fund_Transfer	Account_Balance_Check	Logout
TC_3	D3	Login	Account_Balance_Check	Utility_Bill_Pay	Logout
TC_4	D4	Login	Utility_Bill_Pay	Logout	

ScriptID	Login ID	Password
TC_1	User1	Tester1
TC_2	User2	Tester2
TC_3	User3	Tester3
TC_4	User4	Tester4

Device ID	Device Model	OS_level
D1	Samsung S2	19
D2	Samsung S2	18
D3	Samsung Galaxy Tab	19
D4	Samsung Galaxy Tab	18

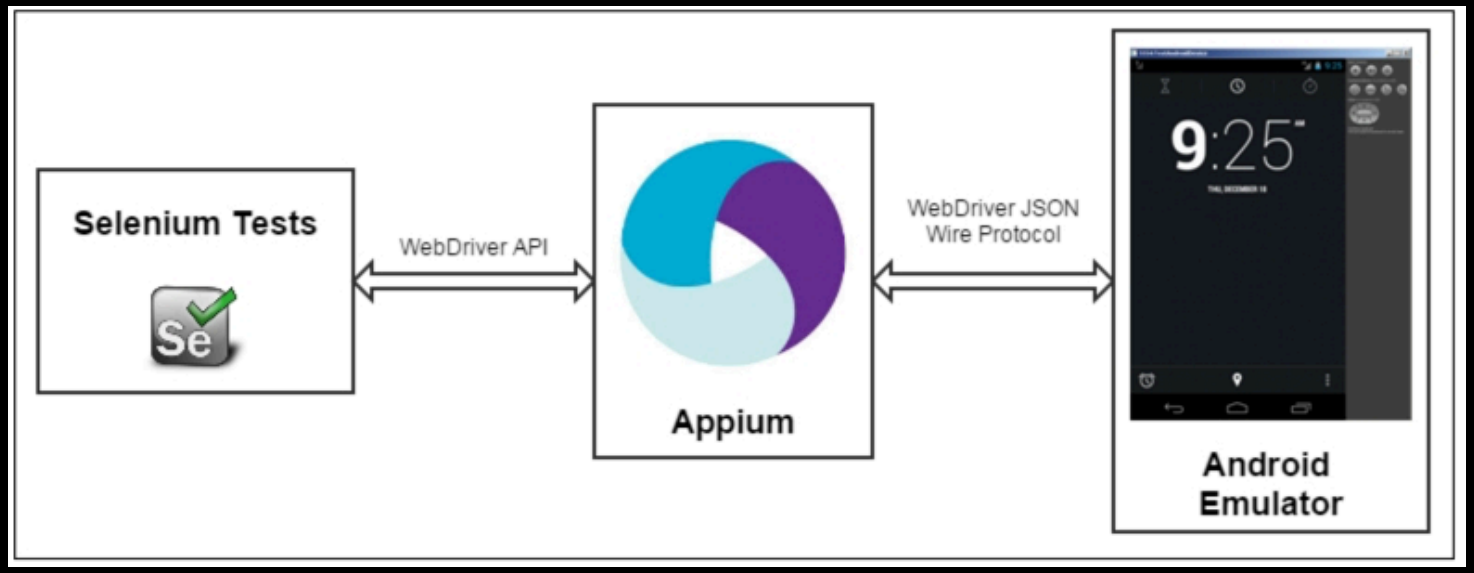
Support Libraries

- General Purpose Routines and Libraries for
  - *File handling*
  - *String handling*
  - *Buffer handling*
  - *Variable handline*
  - *Database access*
  - *Logging utilities*
  - *System and Environment handling*
  - *Application mapping functions*
  - *System messaging or system API enhancements*
- TestNG, Cucumber, JUnit, NUnit

Appium

Download Appium from http://appium.io

- Appium is a HTTP server written in node.js
  - *Relies on UIAutomator library from Android SDK*



Appium Prerequisites

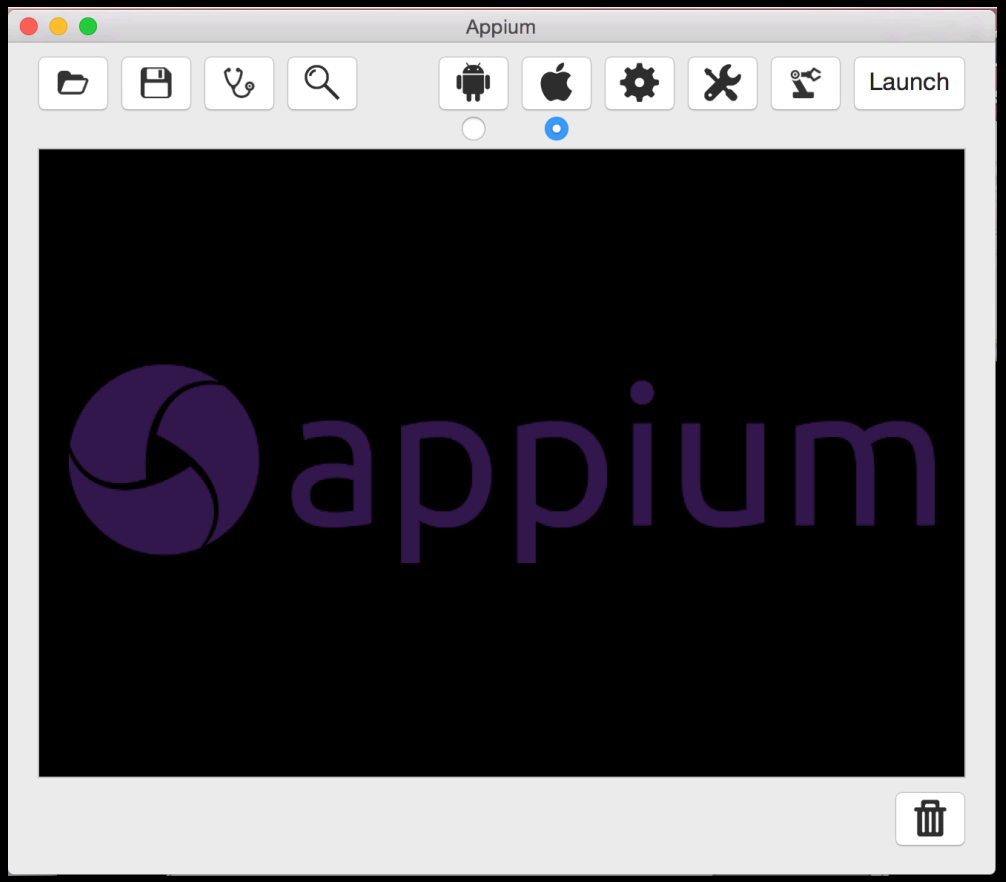
- Prerequisite to use APPIUM
  - *ANDROID SDK [Link]-*
  - *JDK (Java Development Kit) [Link]*
  - *TestNG [Link]*
  - *Eclipse [Link]*
  - *Selenium Server JAR [Link]*
  - *Webdriver Language Binding Library [Link]*
  - *APPIUM For Windows [Link]*
  - *APK App Info On Google Play [Link]*
  - *Node.js (Not Required - Whenever Appium server is installed, it by default comes with "Node.exe" & NPM. It's included in Current version of Appium.)*

Install the Selenium WebDriver

- Start the Emulator
  - *emulator @Nexus_5_API_21_x86 &*
  - *adb devices #to get serialID*
- Download Android server
  - *Get from http:// selenium-release.storage.googleapis.com/ index.html*

Appium App for Mac

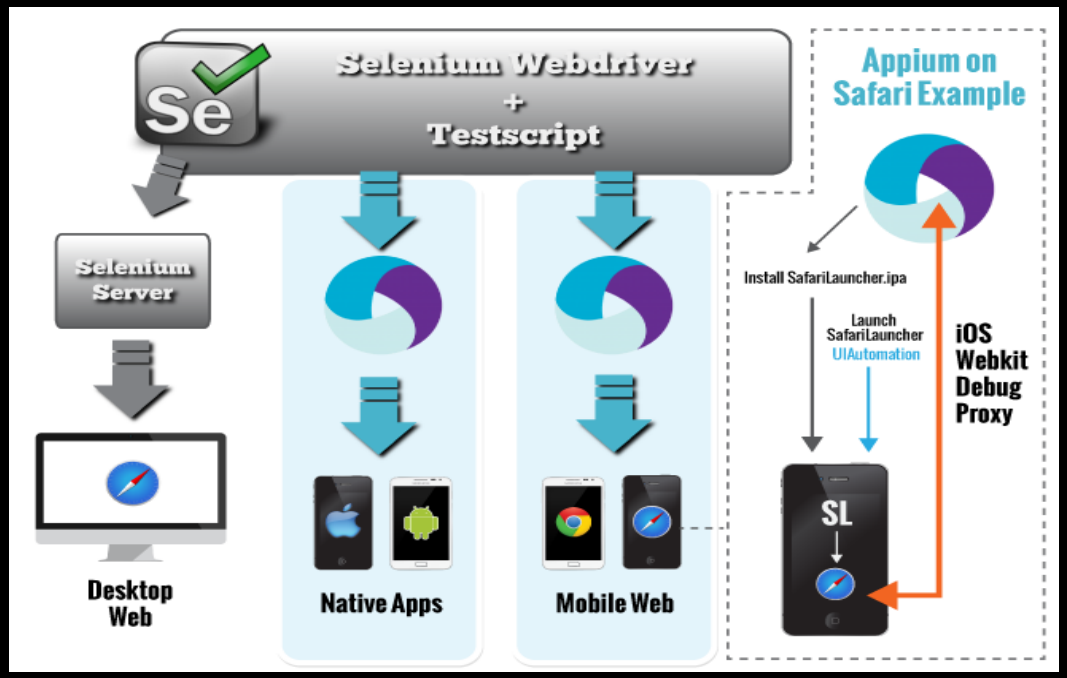
- Appium window

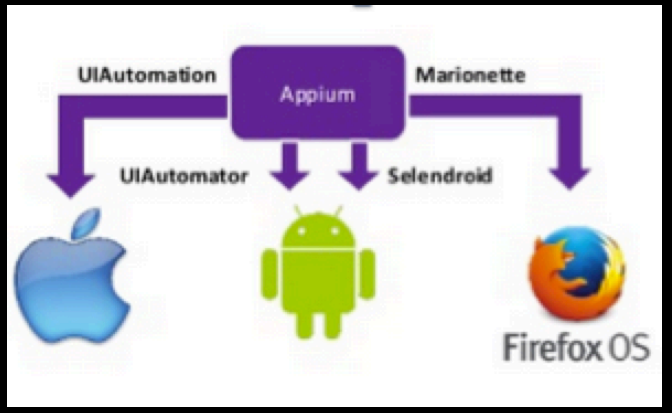


Creating Test Scripts from Code

What is Appium?

- Appium is an HTTP Server that creates and handles Web driver sessions
- Appium Test Spawns a Server and listens for proxied commands
- Appium proxies commands to a UIAutomation / Selendroid





Creating Test Scripts from Recording

Setting Up Your Android Device

Put it into Developer Mode

- Open Settings / About phone
- Tap Build Number 7x
  - *You are now a developer*
- Device Settings:
  - *Developer Options / USB Debugging*
  - *Security – check Unknown Sources*
- Settings / Developer options
  - *Show Touches*
  - *Enable pointer location*

Setting Up the Device

Install TouchTest Agent

SOASTA

Running Automated Test Scripts

Setting Up Your Device Matrix

- How many releases of Android [GooglePlay]
- How many device types – most popular
- Emulators
  - *One per system; cloud hosted e.g. SauceLabs*
  - *Not the same as running on hardware*
- Real devices
  - *Local – how many and which to purchase*
  - *Remote – inside your company*
  - *Remote – 3rd party cloud: NTT AppKit, DeviceAnywhere, SauceLabs, Perfecto Mobile, Mobile Labs, Amazon Remote Device Cloud*

Device Clouds

- On-premise device cloud: your own devices set up for access by anyone at your company
  - *Examples: Mobile Labs, Device Anywhere*
- Public cloud: publicly hosted devices accessible on a pay-as-you-go model
  - *Examples: Device Anywhere, NTT Remote Test Kit, SauceLabs, AWS Device Farm*
- Private cloud: Reserved devices hosted by third-party
  - *Examples: NTT Remote Test Kit*

Setting Up Continuous Integration

Measuring Code Coverage

- Emma