Robotium

Prepared by:

- David Miguel
- Hema Lakshmi
- Karam Alhadithi
- Elham Shahrour
- Nesreen Al-Malkawi
- Abdullilah Alhaj Younes

bit.ly/robotium-testing

Slides, tutorials, source code, useful links...

Index:

What is Android?

- Android is an open source and Linux-based Operating System for mobile devices.
- Android application run on different devices powered by Android.
- The latest Android version 6 Marshmallow.

Android Features

Beautiful UI	Multi-tasking
Connectivity	Multi-touch
Storage	Multi language
Media support	Resizable widgets
Web browser	Android Beam
Wi-Fi Direct	Messaging
GCM (Google Cloud Messaging)	

Android Features Cont.

- Android powers hundreds of millions of mobile devices.
- largest installed.
- growing fast.

You can start your Android application development on either of the following operating systems:

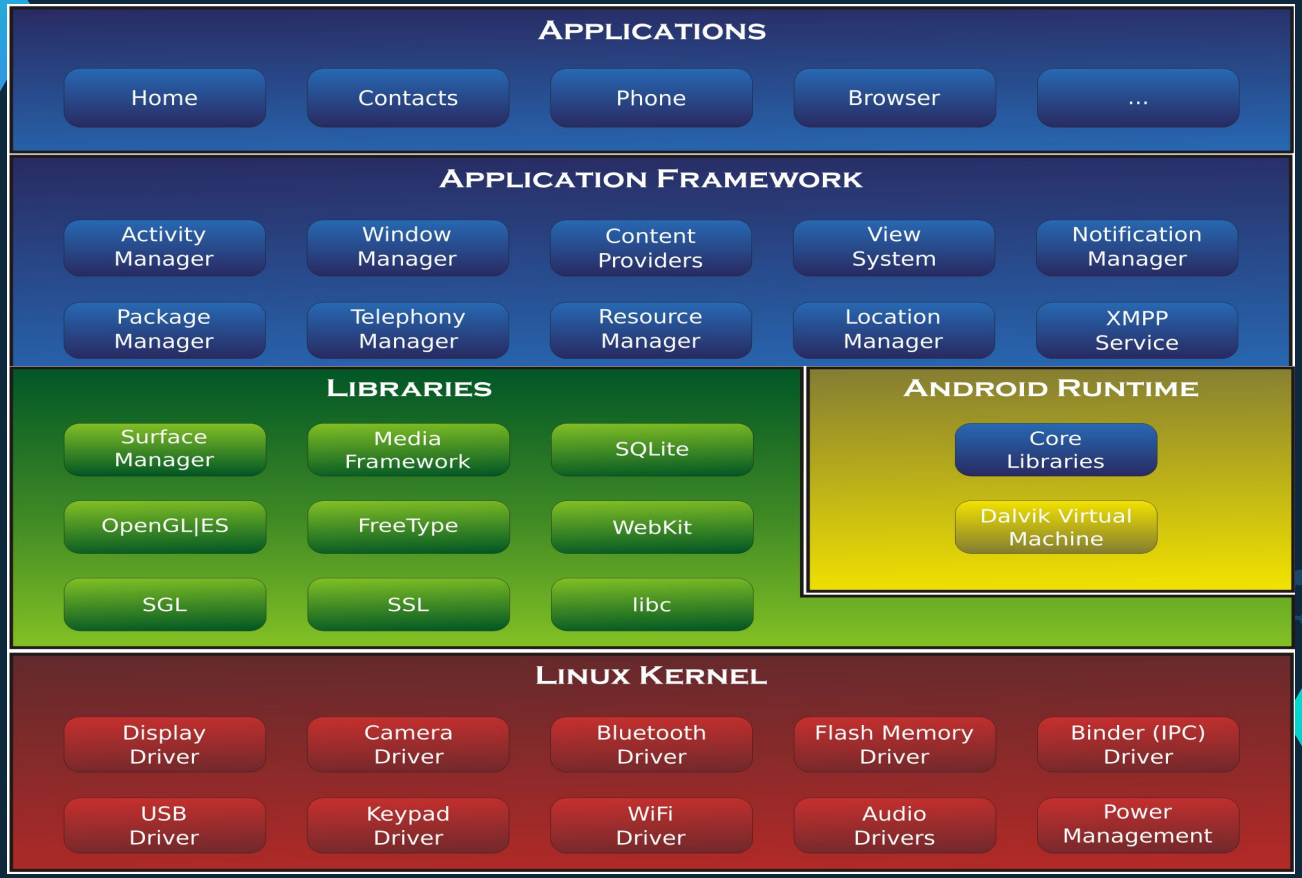
- Microsoft Windows XP or later version.
- Mac OS X 10.5.8 or later version with Intel chip.
- Linux including GNU C Library 2.7 or later.

Android Features Cont.

Second point is that all the required tools to develop Android applications are freely available:

- Java JDK7 (Java Development Kit).
- Android studio.

Architecture:



Architecture Cont.

- <u>Linux kernel:</u>

This provides basic system functionality:

➔ process management.
➔ memory management.
➔ device management.
➔ networking .

Architecture Cont.

- <u>Libraries:</u>

Libraries including open-source:

- ➔ Web browser engine WebKit.
- ➔ SQLite database.
- ➔ Libraries to play and record audio and video,
- ➔ SSL libraries responsible for Internet security etc.

Architecture Cont.

- <u>ART and Dalvik:</u>
  - ➔ Provides a key component called Dalvik which is a kind of Java Virtual Machine specially for Android and this VM is used for Android versions before Android 5.
  - ➔
  - ➔ Now (ART) is the managed runtime.
  - ➔ ART as the runtime executes the Dalvik.

Architecture Cont.

- <u>Application Framework:</u>

Provides higher-level services to applications in the form of Java classes. Application developers can use this services.

- <u>Applications:</u>

You will find all the Android application at the top layer to be installed on this layer only.

Application Components

- There are following four main components that can be used within an Android application:

  - ➔ Activities
  - ➔ Services
  - ➔ Broadcast Receivers
  - ➔ Content Provides

Application Components Cont.

- Activities:

An activity represents a single screen with a user interface. If an application has more than one activity, then one of them should be applied first.

Application Components Cont.

- <u>Services:</u>

A service is a component that runs in the background to perform long-running operations. like playing music in the background while the user is in a different application.

Application Components Cont.

- <u>Broadcast Receivers:</u>

respond to broadcast messages from other applications or from the system.

Application Components Cont.

- <u>Content Providers:</u>

A content provider component supplies data from one application to others on request.

Additional Components

- Fragments
- Views
- Layouts
- Intents
- Resources
- Manifest

Additional Components Cont.

- <u>Fragments:</u>

A Fragment represents a part of user interface in an Activity. You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.

Additional Components Cont.

- <u>Views:</u>

Each view in a user interface represents a rectangular area of the display.

Additional Components Cont.

➢ <u>Layouts:</u>

A layout defines the visual structure for a user interface, such as the UI for an activity or app widget. You can declare a layout in two ways:

- Declare UI elements in XML.
- Instantiate layout elements at runtime.

Additional Components Cont.

- <u>Intents:</u>

An intent allows you to start an activity in another app by describing a simple action you'd like to perform in an Intent object.

Additional Components Cont.

➢ <u>Resources:</u>

- An Android app is composed of more than just code, it requires resources that are separate from the source code.

- For every resource that you include in your Android project, the SDK build tools define a unique integer ID, which you can use to reference the resource.

Additional Components Cont.

- <u>Manifest:</u>

Before the Android system can start an app component, the system must know that the component exists by reading the app's AndroidManifest.xml file (the "manifest" file).

Android Testing Tools

❖ Testing is very important because it helps you:

- Improve the quality of your apps.
- Ensure better user satisfaction.
- Reduce overall development time spent on fixing defects.

❖ Android Testing Support Library:

This library allow you to quickly build and run test code for your apps, including JUnit 4 and functional user interface (UI) tests.

Android Testing Tools

- The Android Testing Support Library includes the following test automation tools:

  - AndroidJUnitRunner: JUnit 4
  - UI Automator: UI testing framework.

Android Testing Tools

- **Monkey:**

This tool runs on your emulator or device and generates pseudo-random streams of user events. You can use the Monkey tool to stress-test applications that you are developing, in a random yet repeatable manner.

- **Monkeyrunner:**

This testing system provides an API for writing programs that control an Android device or emulator from outside of Android code.

Android Testing Tools

Black Box Testing impossible!!

- Requires deep knowledge of widgets
  - Widget Ids
  - Widget Properties
  - What has focus
  - Order of widgets
  - Etc.

Android Testing Tools

Black Box Testing impossible!!

- Often requires deep knowledge of Android internals
  - Especially around menus, dialogs, etc.
- Makes for brittle unit tests
  - As the UI changes, the test often must change dramatically.
- Poor instrumentation
  - Instrumentation is a feature in which specific monitoring of the interactions between an application and the system are possible.
  - Use of runOnUIThread to execute UI work that isn't covered by TouchUtils or TestCase class.

Robotium

What is Robotium?

- Robotium: Is an open-source test framework for writing automatic Gray-Box testing cases for Android applications.
- Robotium framework is released under Apache License 2.0.
- Its founder and main developer is Renas Reda.

Robotium Cont.

- Test Case That Could Be Written:
  - Function test scenarios.
  - System test scenarios.
  - Acceptance test scenarios.

- Robotium and Android Application Testing:
  - Source code and the APK file.
  - Only the APK file.

Robotium Cont.

- Robotium is similar to Selenium, but for Android.
- Tests can be executed on an Android Virtual Device (AVD) or Real device.

What is Needed?

Android Studio:

- Install JDK.
- Install Android Studio.
- Add *Android API level.*

Eclipse:

- Install JDK.
- Install Eclipse.
- Install SDK.
- Install ADT.

Why Robotium?

- Easy to use for anyone with Android Studio & Eclipse.
- Records user actions on emulators and actual devices.
- Test Android apps, both native and hybrid.
- Supports binary APKs and apps with source code.
- One scripts for all Android versions.

Why Robotium? Cont.

- Requires minimal knowledge of the application under test.
- The framework handles multiple Android activities automatically.
- Minimal time needed to write solid test cases.
- Fast test case execution.

Why Robotium? Cont.

- Integrates smoothly with Maven, Gradle or Ant to run tests as part of continuous integration.
- Automatic timing and delays.
- No modification to Android platform.
- Takes screenshots of test execution.

Types Of Tests

Types Of Tests

White Box Testing:

- Testing internal structure and design of software.
- Visibility to code and write test cases.
- Tests code for accuracy and correctness.

Black Box Testing:

- Software internal structure is not known to tester.
- Based on requirements.
- It can be hard to find the cause of the failure.

Types Of Tests Cont.

Unit Testing:

- Test the small unit possible.
- Individually and independently for proper operation.

Functional Testing

- Verify that a software application performs and functions correctly according to design specifications.
- Checks the core application functions, text input, menu functions and installation and setup on localized machines.

Types Of Tests Cont.

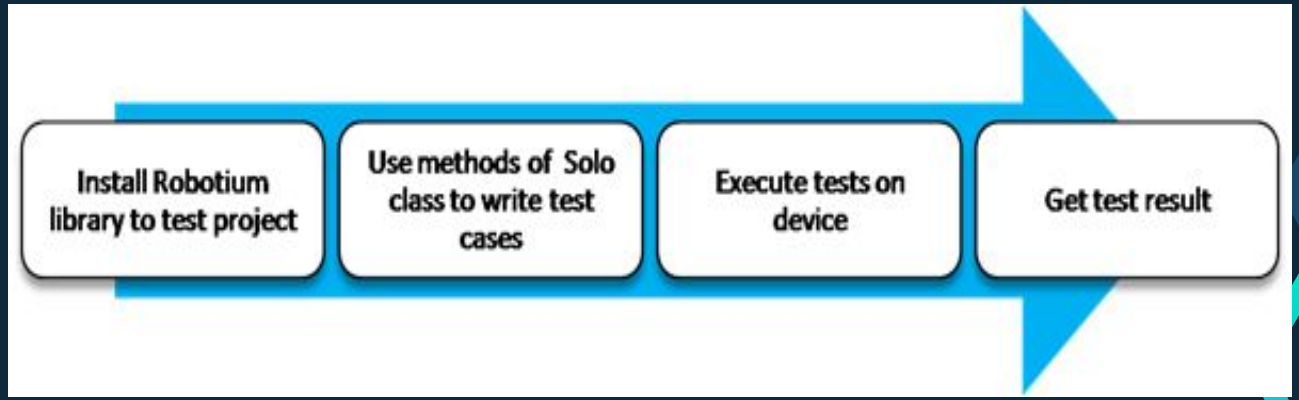
System Testing:

- Tests Software as a whole to make sure it meets specified requirements.
- Subset of black box testing.
- Functionalities of the system are tested from an end-to-end perspective.

Acceptance Testing:

- It is done by the user or customer although other stakeholders.
- Test conducted to determine if the requirements of a specification or contract are met.

Android Application Testing Procedure Using Robotium:



Design Test Specification

- Define target to be tested.
- Plan the test types should be conducted.
- Design test cases for maximum coverage but minimize number of test cases.

Create Test Project

- Click on Android.
- To Add dependency ,Go to build.gradle(Module app) under gradle. scripts
- Add robotium lib inside dependencies block
- Open the tab “Build Variants” (on the bottom left side of Android Studio), and configure “Test Artifact” to “Android Instrumentation Test”
- Expand app -> java folders
- Right-click on that package. Select “New -> Java Class” from the context menu.

Link: Detail tutorial

Test Suite and Test Case

- '*Validation suite*' is a collection of individual test cases that will be run in a test sequence until some stopping criteria are satisfied.

- Test case is a set of conditions under which a tester will determine whether an application, or software program has passed or failed .

Create Test Suite

- Based on test specification can choose various Testing framework.

- Standard Android testing framework and add Robotium library file to a libs directory in the project folder in case want to test with Robotium framework.

Develop Test Scripts

Steps followed to develop the test script:

- ★ Define launcher activity class name.
- ★ Define setup() and teardown() methods.
- ★ Create instance of Solo class in setup().
- ★ Define test methods.

Adding Test Cases

In the same package with TestSuite, we create TestCase classes

- To test certain activity, create a test case
- Tester can obtain testing activity through getActivity() method
- You can freely create test for a testing activity by create method with name "test + original Method Name"
- In test method, tester can use Android JUnit function to compare the actual value and expected value.

Sample Test suite and test case:

```
import junit.framework.Test;
import junit.framework.TestSuite;
public class AllTests {
    public static Test suite() {
        TestSuite suite = new TestSuite(AllTests.class.getName());
        suite.addTest(TestSuite.createTest(myTest.class, "test_1"));
        suite.addTest(TestSuite.createTest(myTest.class, "test_2"));
        suite.addTest(TestSuite.createTest(myTest.class, "test_3"));
        return suite;
        } }
```

Robotium SOLO API

Solo provides methods to call the Android user interface:

- clickOnText(text)-Search for text in the current user interface and clicks on it.
- enterText()-Enters a text.
- searchText(text)-Searches for a text in the current user interface, return true if found.
- SearchButton(text)-Searches for a button with the text in the current user interface.
- ClickOnSearch()-Allows to click on part of the screen.
- ClickOnButton(text)-Clicks on a button with the "text" text.
- waitForText(text)-Waits for a text on the screen, default timeout 5 seconds.

Robotium SOLO API Cont.

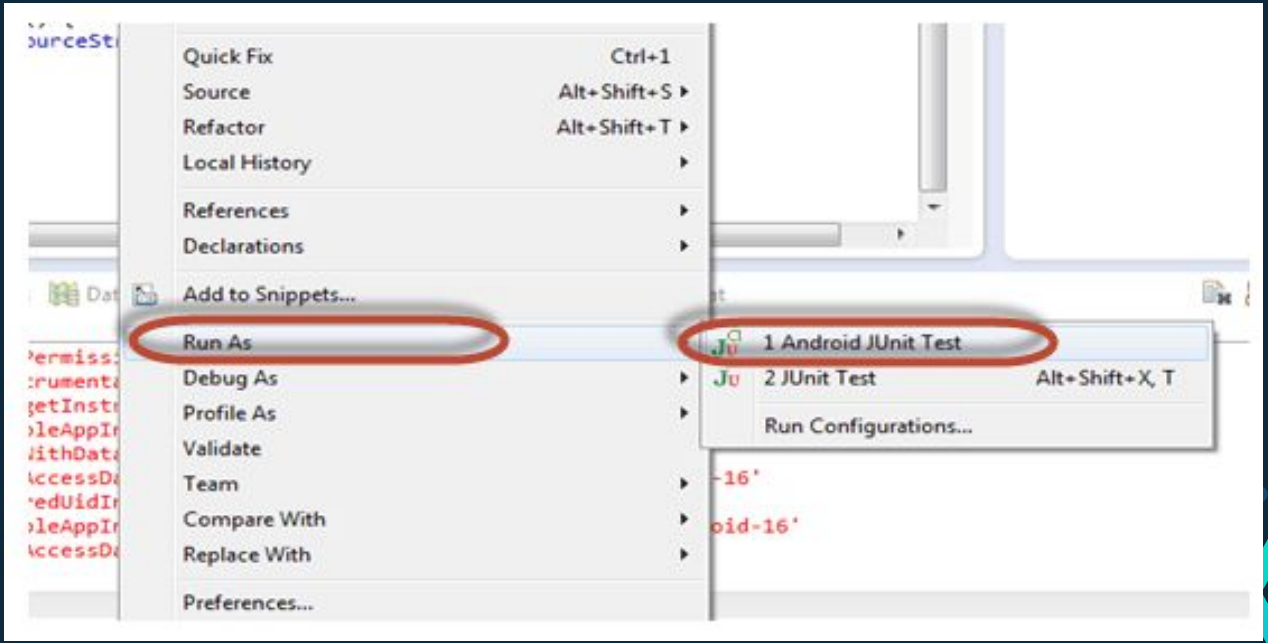
- assertCurrentActivity(text, Activity.class)-Ensure that the current activity equals the second parameter.
- getView(int id)-Searches for the view with the specified ID in the current activity.
- goBack()-Press the back button.
- setDatePicker()-Sets the date in a DatePicker.
- clickInList(x)- Click on item number x in a ListView
- takeScreenshot()-Saves a screenshot on the device in the /sdcard/Robotium-Screenshots/ folder. Requires the android.permission. WRITE_EXTERNAL_STORAGE permission in the AndroidManifest.xml of the application under test.
- isCheckBoxChecked()-Checks if the checkbox is checked.
- waitForActivity(SecondActivity.class, 2000)-Waits for the specified activity for 2 seconds

Run Test Case

After finished writing test program, run the test

1. Right-click on the test project or test case file.
2. Select Run as Android JUnit test.
3. Select Android Emulator from the select device screen.

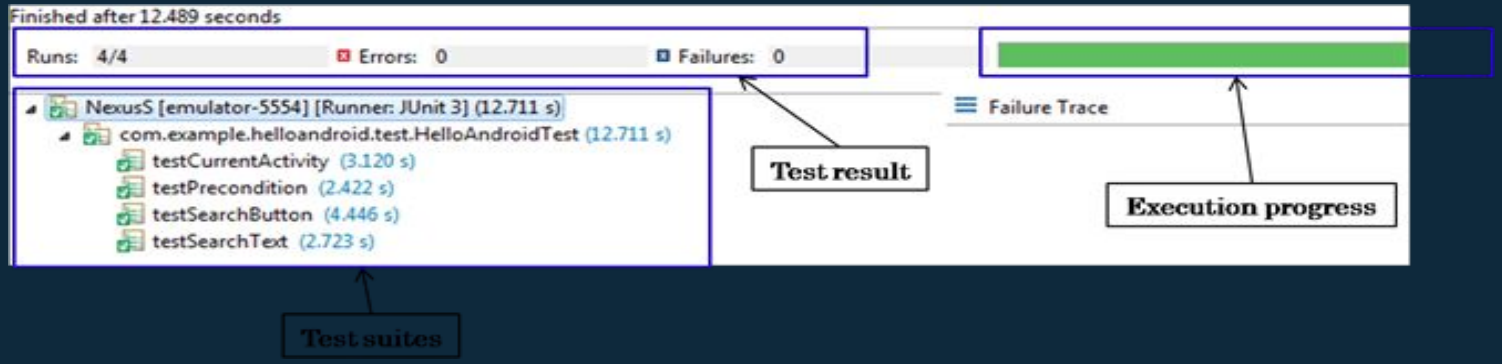
Figure: Run Test Case

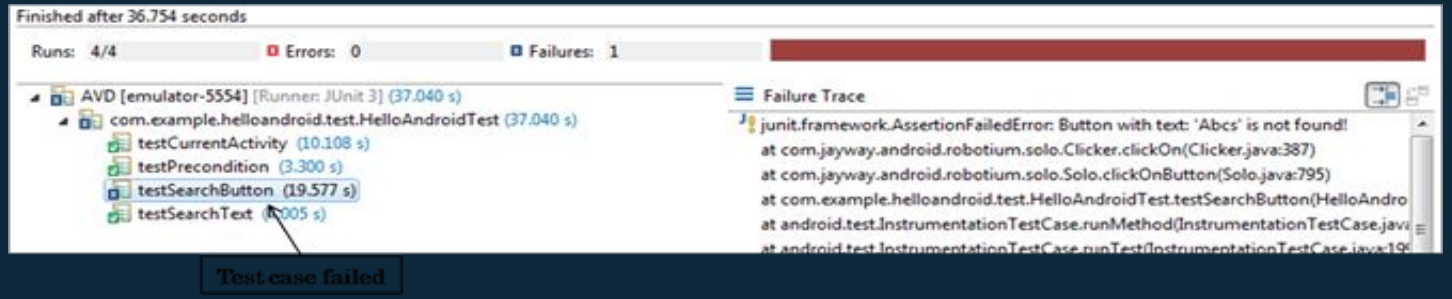


Get Test Results

Junit tab will open and shows the Status of test run

- Display list of tests to run
- Progress of test run, if all test pass, status will appear in Green color, on failures it displays Red color.
- It contains various other options to Stop test run, Rerun test, Show failures etc..





Test result output in all test cases passed and failed

Robotium Limitations

- No support for flash & web applications (web app are supported).
- Test execution on one device at a time.
- Inability of handling different Applications in one test.
- No support for cross-platform (iOS, Windows etc.) testing.
- Can’t interact with Status Bar Notifications.

Complementary Tools

Robotium Recorder

- Paid license Plug-in.
- Available for Android Studio and Eclipse.
- Records and creates professional android test cases.
- Full support for native and hybrid Android apps.
- Robotium Recorder came after Robotium framework in 4 years.

Robotium Testdroid Recorder characteristic

- Tool capture / replay for Android.
- Registered user behavior in the application.
- Tightly integrated with Eclipse.
- Installed as an extension.
- Simplifies automating application testing.
- Creates automatic script code.
- Uses syntax Robotium.
- Allows you to run tests in the cloud.

Remote Control In Robotium

- Helps to test the connection.
- Helps test cases to be executed.
- Software Automation Framework Support (SAFS).

Remote Control In Robotium Cont.

Let’s use it!

Remote Control In Robotium Cont.

Test Application:

- Traditional on-device Robotium.
- Remote Control app.

Remote Control In Robotium Cont.

Implementation:

- On Device.
- Remote Controller.

Alternative Tools

- Espresso
- Robolectric
- Selendroid
- Appium
- Ranorex
- Monkey Talk

Espresso

what is Espresso?

- Is a test automation framework for Android applications developed by Google. It enables you to do automatic black-box UI tests.

- Is a library that ships with the Android SDK to make testing Android user interfaces simpler.

The Main Components Of Espresso

- View Matchers.
- View Actions.
- View Assertions.

Robolectric

What is Robolectric?

- Robolectric is a test framework that mocks a part of the Android framework and allows the running of test cases directly on the **Java Virtual Machine** (**JVM**) with the help of the JUnit for framework.

The Differences Between Robotium and Robolectric

“Robolectric” Vs “Robotium”

Robolectric	Robotium	Features
Robolectric does not need any emulator / device to execute tests. This is why it is much faster than Robotium.	Robotium needs either an emulator or a device to execute tests.	Emulator/Device
It can be configured easily on the build server.	It needs an emulator or a device on the build server to run test cases; otherwise, the test project can't be added to the build process.	Build server
It helps to speed up the test driven development cycle more than Robotium .	It is used to test on an actual Android device and the API edges that are not simulated by Robolectric.	Test-driven development
It uses JUnit 4 non instrumentation testing.	It uses JUnit 3 instrumentation testing.	Instrumentation

Selendroid

What is Selendroid?

- Selendroid is a test automation framework for multi type of mobile application: native and hybrid android app and mobile web.

Why do we need Selendroid?

- No modification of app under test.
- can interact with multiple devices or simulators.
- support gesture.
- support hot plugging of hardware devices.
- support multiple android API.
- built in inspector tool to develop the test case.

Selendroid Components

- Web Driver Client.
- Selendroid-Server.
- Android Driver-App.
- Selendroid-Standalone.

Appium

What is Appium?

- Appium is an open-source tool for automating native, mobile web, and hybrid applications on iOS and Android platforms

Appium Concepts:

- Client/Server Architecture.
- Session.
- Desired Capabilities.
- Appium Server.
- Appium Clients.

Why Appium?

- You don’t have to recompile your app or modify it in any way .
- You can write tests with your favorite dev tools using any WebDriver-compatible language .
- You can use any testing framework.

Appium characteristics

- An open-source tool for creating automated testing of native and hybrid applications.
- Joint and intuitive API for multiple platforms (iOS, Android).
- No need to modify the application being tested.
- GUI for managing the server Appium.

Ranorex

□□What is Ranorex ?

□□- Ranorex is a GUI test automationframework for testing of desktop, web-based and mobile applications. Ranorex is provided by Ranorex GmbH, a software development company for innovative software test automation solutions.

Main Features of Ranorex

- GUI Object Recognition .
- Object-based Capture/Replay functionality

  .
- Test Automation Library for .NET.
- Test Development Environment.
- Flexible Test Automation Interface.

Monkey Talk

What is Monkey Talk?

- Monkeytalk: is a cross-platform testing tool that records and plays back highly readable and maintainable test scripts for native iOS and Android apps, as well as mobile web and hybrid apps. □

Monkeytalk Benefits

- Simulators or Real Devices - no Jailbreaking required
- Robust cross-platform Recording/Playback that actually works!
- Full touch and gesture support
- Integrated Environment to create, run, and edit your tests
- Use the keyword-driven Monkey Talk language, or powerful JavaScript or Java APIs

Monkeytalk Benefit Cont.

- Run your tests interactively or from continuous integration environments
- Validate controls, images, text, or any property of any object
- Data-drive your tests from a spreadsheet
- HTML, XML, xUnit reporting for tests and suites
- Records high-level interactions instead of low-level event streams

Conclusion

- Mobile apps are becoming as complex as desktop programs: **testing is needed.**
- Default Android testing tools are not enough: **no black-box testing.**
- We have to use other testing tools .
- Robotium is a good option to **run gray-box tests** in emulator or physical device.
- There are other tools like Robolectric that **doesn't need emulator or physical device.**

Thanks!.. :)

Any questions?