Testing Android Apps Going from Zero to Hero

Manfred Moser

simpligility technologies inc.
http://www.simpligility.com
@simpligility

simpligility technologies inc

Wanted:

Your Feedback!

simpligility technologies inc

Agenda

Overview about testing
and continuous integration
for Android app development

Why testing?

What can we test?

How can we do it?

Contents

- Testing Overview
- Plain Java testing
- Android SDK testing
- Robotium
- Testing with mocking frameworks
- Robolectric and Robospecs
- Borachio
- Continuous Integration with Hudson

You and me

- You all together know more about this than me.
- Probably some committers and users here.
- If I get something wrong, correct me.
- If you have a question, interrupt me.
- Let's learn LOTS together.
- And have fun doing it.

Why (automated) testing?

Find a problem early and you

- Can fix it quickly
- Save money on QA testing
- Do not get negative customer feedback
- Deal with feature requests instead of bugs
- Avoid production problems
- Can refactor (and change) without breaking old stuff

What are we testing?

Plain java code

Android dependent code

Configuration

User interface

Look and feel

Side note: Apache Maven

Some examples will use Maven as the build tool

See presentation on Monday morning

Good library reuse and dependency use – makes testing easier out of the box

Good tool support

But its all possible without Maven too...

simpligility technologies inc

My first tests

- Pure Java test – can use all JVM based tooling
- Great performance
- Great IDE integration
- Lots of books, documentation,... available
- JUnit in example
- extends TestCase
- Assert

  Morselib example

simpligility technologies inc

JVM testing tools

There is a large choice available:

- Unit testing - JUnit, TestNG
- Mocking - EasyMock, Mockito
- Coverage - Cobertura, Emma
- and many, many more

Large choice of languages like Java, Scala, Groovy, Clojure, Jython, ...

JVM vs Dalvik VM

- Run junit on JVM with android.jar
  - java.lang.RuntimeException("Stub!")
  - Other challenges like final methods, static methods, non-public constructors...
- Some classes in java.*, in commons* and others are patched/different
- "Android" is more than just Java code running on Dalvik – full stack

JVM vs Dalvik/Android stack

JVM based:

- Faster
- More tools available
- More mature tooling

Dalvik based:

- Necessary for integration tests
- Reproduce actual behaviour
- Full stack testing (beyond VM, to native..)

Android SDK Test Tools

- Unit testing
- “Instrumentation” testing
- Monkey
- MonkeyRunner

Instrumentation Testing

- Run on device/emulator
- Uses JUnit 3 and additions
- Various classes to support testing
- Includes some mocking support
- Now pretty good documentation
- Coverage via Emma
- Test app interacts with instrumented app

Instrumentation Testing Setup

- AndroidManifest
  - InstrumentationTestRunner
  - targetPackage
- TestSuite to tie it together
- override setUp(), tearDown()
- @SmallTest, @MediumTest, @LargeTest
- Naming convention test* method name
- ApiDemos, Spinner and MorseFlash examples

Application Testing

- extends ApplicationTestCase<T>
- testPreconditions()

Activity Testing

- extends ActivityInstrumentationTestCase2<T>
- pass T.class to super constructor
- override setUp(), tearDown()
- public void test* methods

Others

- ServiceTestCase
- ActivityUnitTestCase
- ProviderTestCase
- MoreAsserts
- ViewAsserts
- TouchUtils
- android.test.mock

See http://developer.android.com/reference/android/test/package-summary.html

Android-10 Sample - ApiDemos

- android update project -p . --target android-10
- cd tests
- android update test-project -p . -m ..
- cd ..
- ant debug
- cd tests
- ant debug
- ant installt
- ant test

simpligility technologies inc

Android 10 Sample - Spinner

- cd Spinner
- android update project -p . --target android-10
- cd SpinnerTest
- android update test-project -p . -m ../Spinner
- ant debug
- ant installt
- ant test

Android Maven Samples

- MorseFlash, apidemo-android-10, spinner-android-10
- mvn clean install
- ant: error: more than one device and emulator
  - Show multiple devices
  - Failure on tablet

Instrumentation Testing

Pros:

- Test actual Android stack
- Now decent tooling

Cons:

- Slow to deploy
- Slow to run
- Potentially requires devices attached
- Very limited mocking available

Monkey

- User interface exerciser
- Stress testing tool sending UI events
- Monitors application under test
- Detect ANR's and thrown exceptions
- Command line invocation

Demo with HelloFlashLight

Hint: There is a specific presentation on Wednesday

MonekyRunner

- Control device/emulator from outside
- Emulate user behaviour
- Take screenshots
- Jython scripts running on host
- Extensible plugin architecture
- Interaction works via adb
- Unstable?!, not much used?

MonkeyRunner

- MonkeyRunner
- MonkeyDevice
- MonkeyImage
- Write your own plugin in Java
- Write script in Jython

Demo from Android in Practice Appendix

Test tooling

- Eclipse integration
- Ant integration
- And all other tools that are part of SDK
- Other IDE's and tools use SDK tools like adb,..

Robotium

Robotium

Like Selenium for Android

Extends SDK instrumentation testing
- same pros and cons

Add dependency to pom.xml
Or add jar to libs in test project

And start coding tests

Robotium

- ActivityInstrumentationTestCase2
- Solo class

- RobotiumAndroidCalculator example
- Robotium Sample

Robotium

Pros:

- Simplifies SDK instrumentation testing
- All SDK tool based testing can be done in same test project

Cons:

- Limited use for non-ui testing
- Same problems as SDK instrumentation testing

Robolectric

Runs in JVM

Shadows Android SDK classes

No device/emulator necessary for run

High performance

Robolectric – Under The Cover

- Robolectric uses method rewriting
- To access Shadow* methods whenever and Android class method is called
- Result from shadow is passed back to your code

Robolectric

- @RunWith(RobolectricTestRunner.class)
- Can use Junit 4!
  - @Test, any method name
- http testing
  - Robolectric.addPendingHttpResponse(requestMatcher, response) ..
- ShadowAlertDialog.getLatestDialog()
- Robolectric.getShadowApplication() .getNextStartedActivity()

Robolectric

Pros:

- Fast
- Great turnaround during development
- Powerful API

Cons:

- Incomplete Shadows
- Limited since not running on device
- Can not distinguish between API levels

Robospecs

- .equals(Robolectric with Specs2)
- Specs2
  - executable software specification DSL
- Write specifications with Scala
- Due to Scala usage, test code is readable like a natural language
- Build with sbt or Maven

Robospecs example

Testing with mocks

- Mock?
  - wrapper around class
  - returns defined output rather than actually calling wrapped method
  - can remove dependency on environment
    - database
    - network
    - performance
  - In Java – Interface, Implementation and MockImpl
    - Without mocking frameworks test code can be more than the tested code..

Testing with Mocks

- Dalvik
  - SDK mocking very limited
  - AndroidMock – on Dalvik Vm
  - Borachio
- JVM
  - andject example
  - Roboguice – AstroBoy example – with Robolectric and EasyMock

Borachio

- Scala based mocking frameworks that works on Android
- Tests run on device/emulator
- Written in Scala but can test Java and via JNI also native code
- Borachio Warehouse example

- sbt for build

Other tools

- Android Mock
  - Runs on Dalvik
  - EasyMock on Android
- Android Junit Report
  - Allows download of junit report off emulator
- Vogar/Caliper
  - Google sponsored (test) code execution and (micro) benchmarking tool
  - Targets JVM, Harmony or Dalvik

What is Continuous Integration?

- Run build and tests for each check in
- Setup for various development branches
- Run release build as one click action
- Create website with project details as well as analysis of build history
- Provide user interface for non development user e.g. to publish release

Why Continuous Integration

- Avoid “works on my machine” problems - Reproducibility
- Free up developer machine/time – I don't have time to run all tests before each commit
- No IDE dependency (less setup problems)
- Rapid feedback in team
- Improved communication

Continuous Integration Servers

- Hudson/Jenkins
- Cruise Control
- Bamboo
- TeamCity
- and lots more

Commonalities for Setup

- Need to get tool chain on master/slaves
- Get a command line focused build working
- Integrate with SCM
- Look for specific plugins that might help
- Work with deployment platform like web browser, emulator, actual hardware

Example Eclipse Hudson

- Easy to install
- Large community
- Android plugin
- Commercial offering as hosted
- Open source

Eclipse Hudson and Android

- JDK
- Android
  - SDK (for Java and normal API type Android app)
  - And potentially NDK (for C support, e.g. games)
  - And platform versions (1.6, 2.1, 2.2, 3.0...)
- Apache Ant or Apache Maven for command line build
- Install manually, via scripts, via puppet or VM snapshots..

Building an Android App

- Default build
  - Within Eclipse/Android Development Toolkit ADT
  - Optionally command line Apache Ant
- Better with Apache Maven
  - Dependency management
  - Work with repo for release management
  - Advanced features for testing
  - and lots more
- Possible Gradle, SBT, bash, scons...

Why do we need emulators/devices?

- No need for normal plain build
- But build should have testing!
  - Instrumentation testing runs on emulator/device

Android Emulator Plugin

- Create, start and stop emulator(s) for each run
- Capture logcat from emulator/device

Android Maven Plugin

- Start and stop emulators
- Automatically deploy to all attached devices
- Run tests on all attached devices
- Produces junit reports compatible xml on build machine

CI for Other Platforms

- As long as you can run a build on the command line..
- Characteristics of setup for other platform will be similar
  - Java ME
  - iOS
  - Windows Mobile 7
  - Phone Gap and other cross platform tools
- More or less painful depending on operating system requirements, tool chain needs, ...

Options for Install

- On demand on development machine
- Local networked server
- Virtual machine in cloud
- Commercial offering

Installation of CI

- Headless install of Android SDK and build tools
- Install
- Configure
- Watch it run and be notified

Beyond testing

Static analysis
Test code coverage
Trending

Look at Sonar

Automated merging of feature branches

Site build for documentation and more

What lies ahead?

- Automated UI tests using screenshot diffs
- Install and upgrade tests
- Performance tests
- Stability tests (low memory, memory usage...)
- ... towards
- Deployment automation to many markets
- Continuous deployment

Resources Testing

- Android SDK Test Tools
  http://developer.android.com/
- Testing Guide
  http://developer.android.com/guide/topics/testing/index.html
- HelloAndroidTest
  http://developer.android.com/resources/tutorials/testing/helloandroid_test.html
- ActivityTest
  http://developer.android.com/resources/tutorials/testing/activity_test.html
- MonkeyRunner
  http://developer.android.com/guide/developing/tools/monkeyrunner_concepts.html
- Monkey
  http://developer.android.com/guide/developing/tools/monkey.html

- JUnit
  http://www.junit.org/
- TestNG
  http://www.testng.org/

Resources Testing

- Robotium
  http://code.google.com/p/robotium/
- Robotium Samples
  https://github.com/jayway/robotium-samples
- Robotium Calculator example in Maven build
  https://github.com/mosabua/RobotiumAndroidCalculator

- Robolectric
  http://pivotal.github.com/robolectric/
- Robolectric Sample
  https://github.com/pivotal/RobolectricSample
- RoboSpecs
  https://github.com/jbrechtel/robospecs
- Specs2
  http://etorreborre.github.com/specs2/

Resources Testing

- Calculon
  https://github.com/kaeppler/calculon
- Android Mock
  http://code.google.com/p/android-mock/
- Android Junit Report
  https://github.com/jsankey/android-junit-report
- EasyMock
  http://easymock.org/
- Borachio
  http://borachio.com/
- Boarchio Warehouse example
  https://github.com/jaley/borachio-warehouse
- andject
  https://github.com/ko5tik/andject
- Android Maven Plugin
  http://code.google.com/p/maven-android-plugin/
- Vogar
  http://code.google.com/p/vogar
- Caliper
  http://code.google.com/p/caliper

Presentations/Videos/Examples

- Robolectric presentation fro Joe Moore (Robolectric committer)
  http://www.slideshare.net/joemoore1/droidconuk-tdd-android-with-robolectric
- Video about Guice and Borachio used for testing
  http://skillsmatter.com/podcast/os-mobile-server/mocking-and-testing/js-1930
- Source code for Android In Practice book examples
  http://code.google.com/p/android-in-practice/
- And many many more

Resources Continuous Integration

Hudson
http://hudson-ci.org/
http://eclipse.org/hudson

Jenkins
http://jenkins-ci.org/

CruiseControl
http://cruisecontrol.sourceforge.net/

AtlassianBamboo
http://www.atlassian.com/software/bamboo/

JetBrains TeamCity
http://www.jetbrains.com/teamcity/

and many more

Hudson/Jenkins Resources

- The Hudson Book
  http://www.eclipse.org/hudson/hudsonbook
- Jenkins: The Definitive Guide
  http://www.wakaleo.com/books/jenkins-the-definitive-guide
- Android Emulator Plugin
  https://wiki.jenkins-ci.org/display/JENKINS/Android+Emulator+Plugin

Summary

Testing makes things easier
But there is not the ONE solution

Find problems before your customers find it

Implement new features confidently without breaking existing functionality

Use continuous integration to improve communication in your team and software quality

You Want More...

Android Maven Presentation
Monday 10am

Fireside chats
about tablet development and markets
Monday evening

Free signed copy
Maven: The Complete Reference
Tuesday, 3pm

The End

Thank you for your attention.

Contact me for consulting
and open source hacking...

@simpligility

manfred@simpiligility.com

http://simpligility.com