Android Testing Recipes

Patrick Hammond
@patrickhammond

Presentation Info

Presentation and code is available on GitHub

https://github.com/patrickhammond/Presentation-AndroidTestingRecipes

This is not a complete coverage of testing on Android!

Testing ecosystem and challenges is constantly growing and maturing.

Not at all possible to talk about in one sitting.

Goals

- Gain a common understanding of automated testing concepts
- Demo several (free!) automated testing tools available today
- Call out areas of pain that can be avoided
- Make it easier to improve the testing you are already doing today!
- Give you new areas of your app to test

Agenda

- Start with JUnit and Android
- Work into Android's JUnit extensions
- Demo testing of scenarios using Robotium
- Show how Emma can help make testing fun
- Discuss "hot spots" for app testing
- Discuss other interesting testing tools

Automated Testing Motivation (partial list)

- Excellent way to document API behavior
- Helps prevent and uncover "surprises"
- Documents expectations and knowledge about your product
- Small investments let you focus on more "interesting" parts of your product
- Helps you ship a better and more maintainable software faster!

Testing Vocabulary

- Manual testing - Using human intelligence for verification and for when automated testing is not economical
- Unit testing - Testing individual parts of the application in isolation
- Integration testing - Testing how multiple parts of the application function together
- Functional testing - Testing the functional behavior of an application

Setting up a test project

- Your test project can live anywhere on the file system.
- Convention is to put the project in a "tests" folder inside of the main project.

Unit Testing with JUnit

- xUnit style of testing is very common
- JUnit is the base for many testing tools
- Out of the box support in Android
- Easy to run (but not always fast in Android)

Android JUnit Extensions

- Additional JUnit test cases and assertions
- Provides access to resources or context
- Great for testing app components
- Be careful about entering Context hell...
- Make sure to extend the TestCase2 classes

Testing of scenarios using Robotium

- Sits on top of the Android JUnit extensions
- Great for quickly writing high level application tests.
- Convenience methods for views but you always can access a view by its id if needed

Emma on Android for code coverage

- Gives you visibility into what code in your app was invoked at least once during testing
- Only available from the command line and only on the emulator or a rooted phone
- cd tests; ant all clean emma debug / uninstall install test

Interesting areas to test

Do not be terrified by the next few slides!

Also, the next few slides are not a complete list of things to consider when planning out what you want to test

Interesting areas to test

- Different OS versions
- Different form factors
- Different orientations
- Orientation changes
  - Especially when dialogs are active!
  - Especially during background processing!
  - Especially when you have captured input!

Interesting areas to test

- Devices without capabilities your application might need (ex: phone capabilities don’t exist on tablets)
- Features only available on some versions of Android
  - Especially if you are branching around API availability in your code!
- Support for devices like the Kindle Fire

Interesting areas to test

- Application upgrade and downgrade
  - Incompatible data models across versions
  - Unexpected application state
- No data connection
- Backend service outages
- Unexpected backend service responses
- Backend services respond slowly

Interesting areas to test

- Widgets, custom components, animations
- Alarms and notifications
- Cross application integration
- App behavior with the SD card unmounted
- Environmental issues (ex: can't get a GPS location, low battery)

Interesting areas to test

- Potentially long operations on the main thread (check out android.os.StrictMode)
- Loading lots of data in your app
- Performance impact of DB transactions
- CPU, battery, and memory impact
  - Parsing, images, chatty network apps
- Application interruptions (ex: phone calls)

Interesting areas to test

- Sensors (ex: accelerometer's don't behave the same on all devices)
- Time changes
  - DST
  - Traveling from one time zone to another
- Data sharing across multiple installs of your app

Interesting areas to test

- Device restarts
  - Does your app kick off needed services
  - Is your app data now stale?
- Other apps calling into your app
- Sharing data from your app to others
- Behavior of "standard" APIs
- Behavior on HTC phones :-)

Interesting projects

- **Robolectric**
  http://pivotal.github.com/robolectric/
  Advertises that it helps reduce test-feedback cycles on Android
- **Android Mock**
  http://code.google.com/p/android-mock/
  Lets you perform mocking (EasyMock syntax) on Dalvik

Interesting projects

- **Calabash**
  http://github.com/calabash/calabash-android
  Lets you write BDD Cucumber tests. Sits on top of Robotium.

Other areas of interest

- **monkeyrunney**
  Python tool for starting and interacting with applications

- **The Monkey**
  Generates a set of pseudo-random events in your app for stress testing

Useful Links

- JUnit - http://www.junit.org
- Android Developers - Testing (read this!)
  http://developer.android.com/guide/topics/testing/index.html
- Robotium - http://code.google.com/p/robotium/

Thanks!

Questions?