Debugging

Dr Tadhg O’Sullivan

Outline

- Debugging
- Android Debug Bridge
- Debugging in Code
- Debugging in Emulator
- Debugging in Hardware
- Eclipse Debugging
- Debug & Dalvik Debug Monitor Server (DDMS)
- Other tools

Debugging

- **Debugging** is a methodical process of finding and reducing the number of bugs, or defects, in a computer program or a piece of electronic hardware, thus making it behave as expected.

(Wikipedia ☺)

Why We Need Debugging

- No one is perfect , that includes me, you and every developer in the world.
- You never learn unless you try things.
  - Developing by ‘Trial and error’ is sometimes necessary
  - remember Computer Science we build to learn
- Faulty/complex underlying frameworks
  - App ->Android Libs -> Dalvik -> OS
  - Faulty / Complex / Differnet hardware (CPU (ARM or Intel) native element breaks, etc..

Android Debug Bridge

- Almost all debugging features of Android in Eclipse are also available in command line interface (CLI) via Android Debug Bridge (adb)
  - Eclipse just makes it nicer
- For example, LogCat tab in Eclipse is available via 'adb logcat' command in CLI
- CLI is useful for debugging but in an ideal world is used to integrating 3rd party tools

Debugging in Code

- Android's alternative to Java's 'System.out.println' statements (you can still use **java.util.logging** in Android*)*

```
import android.util.Log;
...
Log.i("MyActivity", "we're here!");
```

- You'll see in LogCat tab:
  - "I/MyActivity we're here!"

android.util.Log

- Other methods:
  - Log.v() - verbose
  - Log.d() - debug
  - Log.i() - information
  - Log.w() - warning
  - Log.e() - error
  - Log.wtf() - 'what a terrible failure' :-) (take action)
- LogCat allows flexible filtering configuration

Debugging in Emulator

- You can easily debug your programs in emulator
- Emulator allows you to change/emulate
  - Android version
  - Screen size/dpi
  - Network speed
  - Location (mockup locations)
  - Phone calls/SMS sending and receiving

Debugging in Hardware

- Nothing beats a real device for look & feel,
- speed, reaction, etc.
- Enable Developer options (might be hidden, 'secret taps' required to open them up!)
  - Normally got to “about phone”
  - Tap on “Build number” until you’re a developer
  - Can be different so check your android version / model on-line to find out

- Available Options:
  - Drawing, Monitoring, Animation, Apps

Debugging in Hardware

- Drawing:
  - Show layout bounds
  - Various GPU settings
  - Disable HW overlays
- Monitoring
  - Show CPU usage
  - Profile GPU rendering
  - Enable traces
- Animation:
  - Scale of animation for window and transition.
  - Animation duration
- Apps:
  - Don’t keep activities
  - BG process limit
  - Show all ANR’s

Eclipse Debugging

- Eclipse has two useful perspectives for debugging:
  - Debug Perspective
  - Dalvik Debug Monitor Server (DDMS) Perspective

- Can be added in Eclipse via 'Window -> Open Perspective -> Other ... -> (Debug|DDMS)

Eclipse: Debug Perspective

- Built into Eclipse already, and enhanced by Android ADT
- Allows:
  - Debug – display android apps and currently running threads
  - Variables – when breakpoints are set, display variable values
  - Breakpoints – display list of breakpoints in your code
  - LogCat – allows to view system messages

Eclipse: DDMS

- All features of DDMS (also available in CLI)
  - Devices – list of devices and AVDs
  - Emulator Control – device functions
  - LogCat – system messages (also in Debug)
  - Threads – running threads within VM
  - Network – Network traffic
  - Heap – heap usage for a VM
  - Allocation Tracker – memory allocations for objects
  - File Explorer – device filesystem

Other Debugging Tools

- Lint – help identify and correct problems with the structural quality of your code
- Hierarchy Viewer – debug & optimize your user interfaces
- Traceview – graphical viewer of execution logs
- Systrace – collect and review code execution data for your app and Android system

Questions

- Please ask in the Student Forum