

Android for .NET Developers Series

Getting Started

Android Toolset Fundamentals

Jim Wilson

jimw@jwhh.com

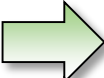


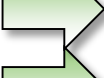

@hedgehogjim

<http://facebook.com/hedgehogjim>



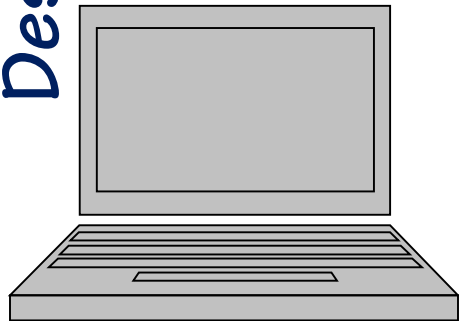
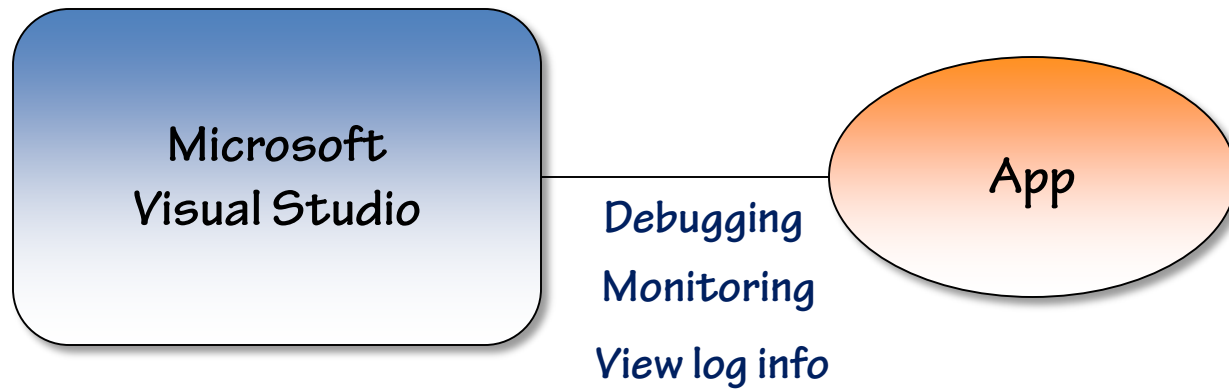
pluralsight
hardcore developer training

Outline

-  Tool architecture
-  Overview of key Android tools
-  Android Debug Bridge
-  Eclipse
-  Logcat

.NET tools architecture overview

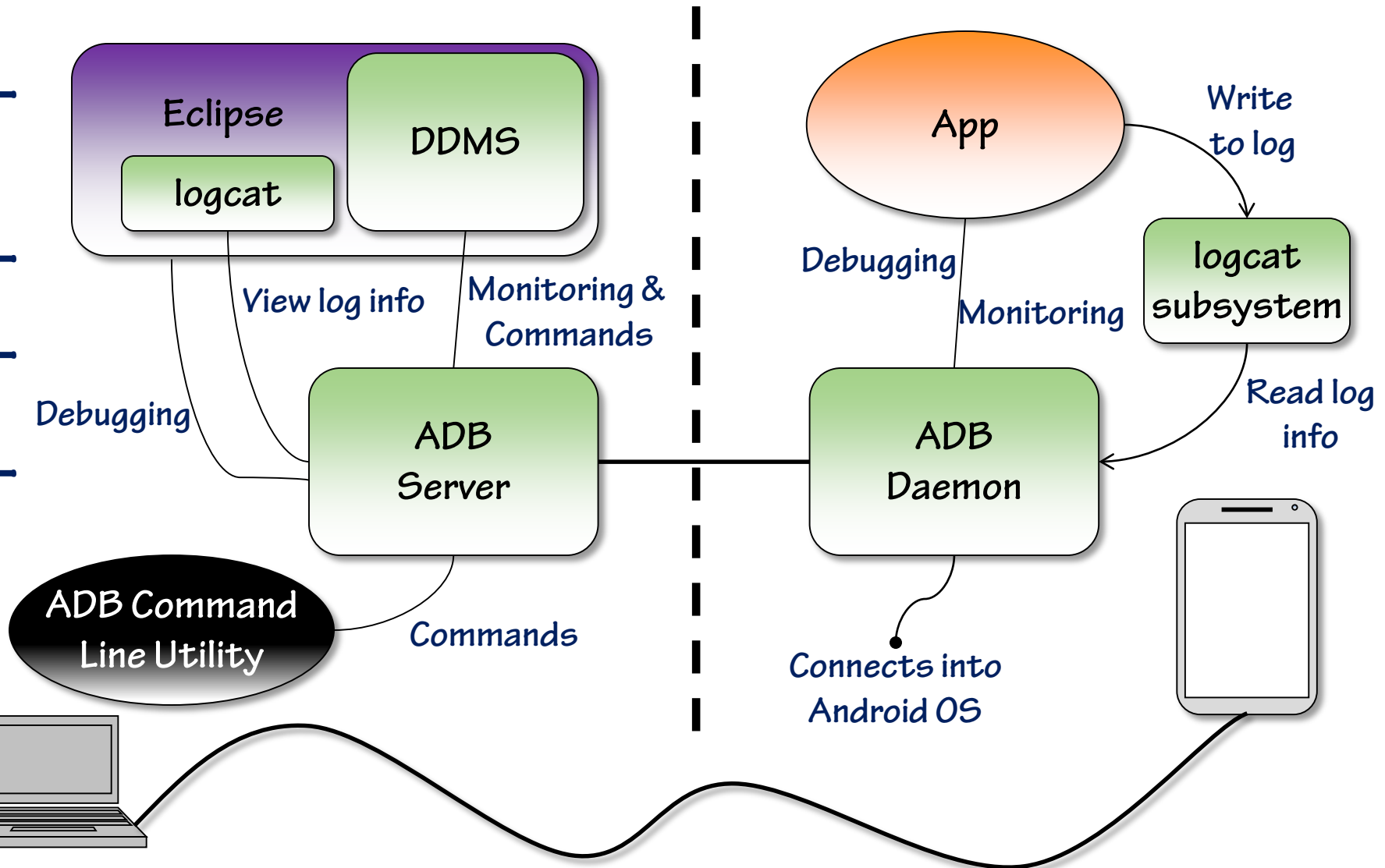
Desktop/Laptop Computer



Android tools architecture overview

Desktop/Laptop Computer

Device or AVD



Key Android debugging tools

 **Majority of Android development handled with 4 tools**

 Android Debug Bridge (ADB)

 Eclipse

 Logcat

 Dalvik Debug Monitor Server (DDMS)

 Many more tools available

- Complete list at <http://bit.ly/11Wd2PC>

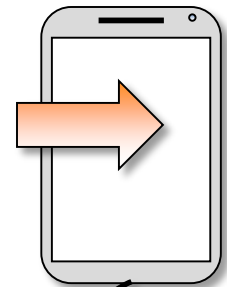
Android Debug Bridge (ADB)

- ➡ **Android Debug Bridge is the most important development tool**
- ➡ Makes interaction between your desktop and device/AVD possible
 - ❑ Daemon running on the device
 - ❑ Server process running on your desktop
 - ❑ If ADB isn't working, no aspect of debugging works
- ➡ Provides a command-line interface through the adb utility
 - ❑ Located in `<install-folder>\sdk\platform-tools`
 - ❑ Useful to add to your "path" environment variable
 - ❑ Some features provide control over ADB processes & device/AVD connections
 - ❑ Some features provide ability to interact with device/AVD

ADB Command
Line Utility

ADB
Server

ADB
Daemon



adb utility process control commands

- ➔ **adb utility manages ADB processes and device/AVD connections**
- ➔ One of the most important commands shows connected devices/AVDs
 - ❑ Run adb passing the **devices** command
 - ❑ Shows each device/AVD along with ADB assigned serial number
- ➔ Controlling adb server lifetime
 - ❑ Use **kill-server** command to signal current ADB server instance to be shutdown
 - ❑ If works correctly, provides no feedback
 - ❑ ADB server will normally restart automatically when needed
 - ❑ Can assure that an instance is running with **start-server** command

Real
Device

```
> adb devices
List of devices attached
00000X999999X123    device
emulator-5554      device
```

```
> adb kill-server
```

Emulator

5554:AVD_for_Galaxy_Nexus_by_Google2

adb utility device commands

➡ **adb utility can perform actions on device/AVD**

➡ Provides a rich set of features and capabilities

- ❑ Copy files between desktop & device/AVD
- ❑ Can install/uninstall app on device/AVD
- ❑ Much more – complete list at <http://bit.ly/11WGTHF>

➡ Can open a interactive Linux shell on device

- ❑ Use **shell** command with no arguments

➡ Can issue a command-line that runs within a Linux shell on the device

- ❑ Use **shell** command followed by Linux command-line
- ❑ Returns immediately

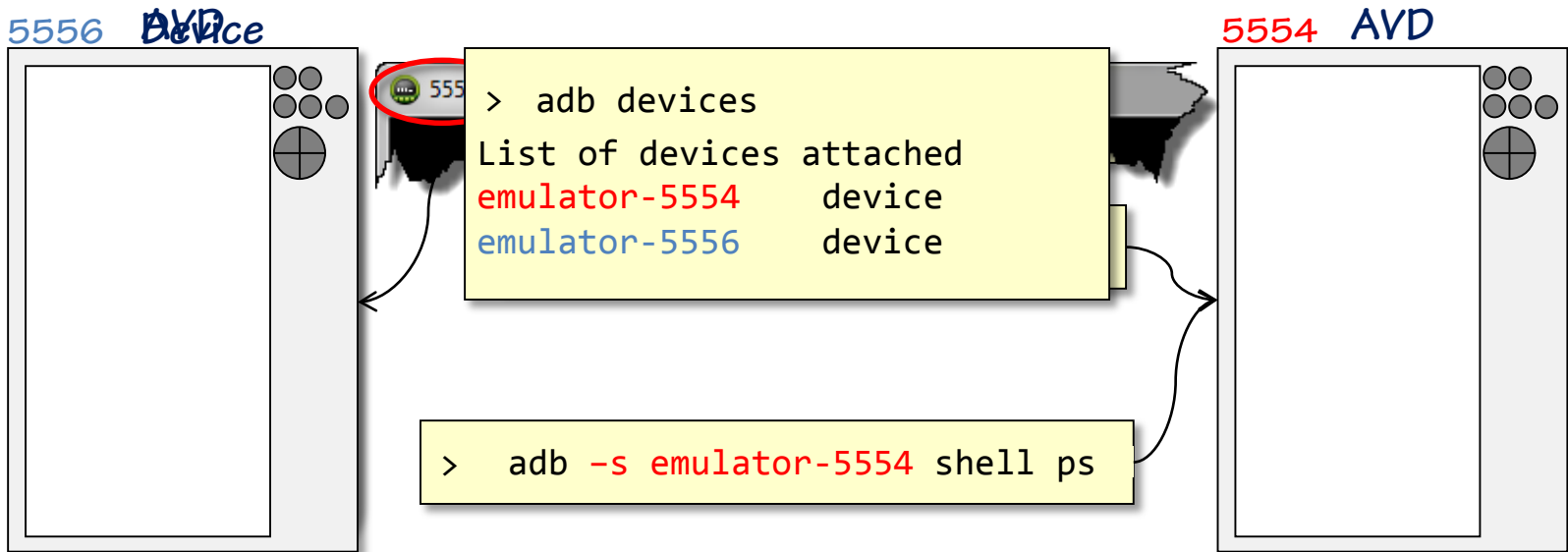
```
> adb shell
shell@android:/ $ cd ~
cd ~
shell@android:/data $ ps
ps
... ..
... ..
shell@android:/data $ exit
>
```

Running in a
Linux shell on
device/AVD

```
> adb shell ps
... ..
... ..
>
```

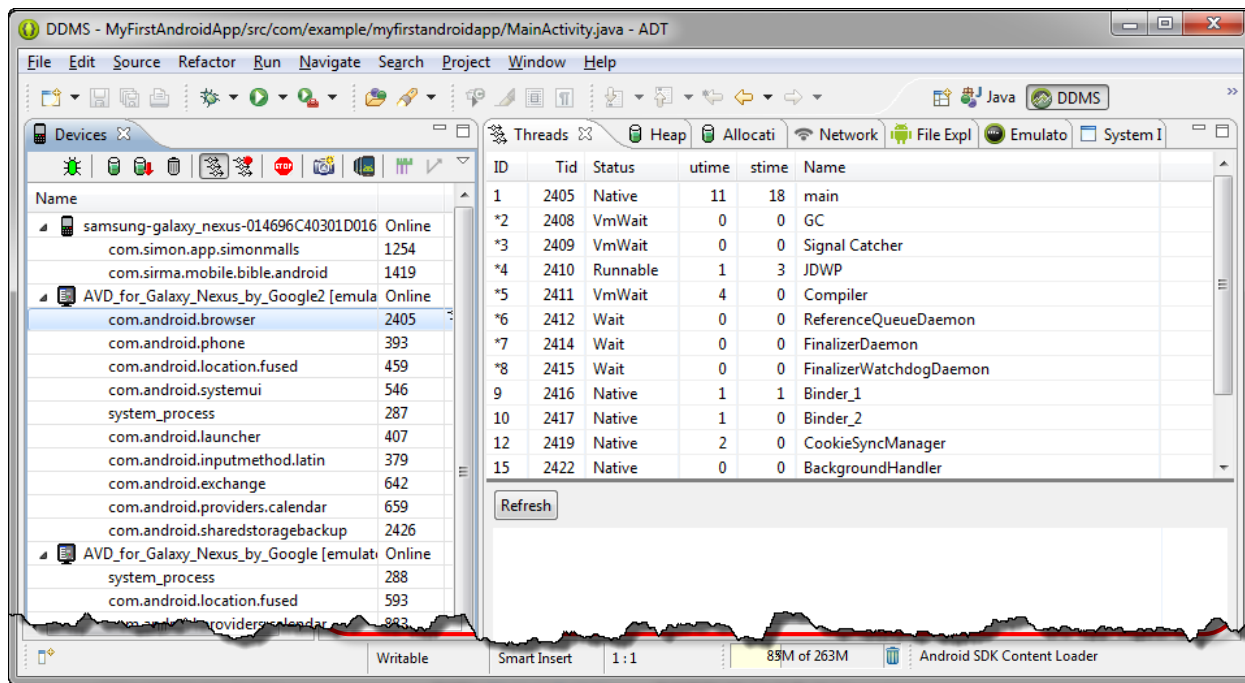
Identifying the adb utility target

- ➔ **Multiple connected devices/AVDs require special handling**
- ➔ When multiple connections, you must specify the adb command target
 - ❑ When only a device or only an AVD, target could be inferred
- ➔ Provides short-hand for common scenarios
 - ❑ To target the only connected device, include **-d** prior to the command
 - ❑ To target the only open AVD, include **-e** prior to the command
- ➔ Multiple connections of the same type, require explicit target
 - ❑ Use **-s** followed by the ADB assigned serial number



Eclipse

- ➔ **Eclipse is the hub of your development work**
- ➔ Serves as the editor, source code debugger, project manager
 - ➔ It is truly the Integrated Development Environment
 - ➔ Serves as the host user interface for other Android tools
 - ❑ Logcat is available as a console-style View
 - ❑ DDMS is available as Perspective



Logcat

➡ **Logcat is the Android logging system**

➡ Serves as a message repository

- ❑ Records information about system events
- ❑ Apps can write messages

➡ Messages written to logcat using android.util.Log class static methods

➡ Can be easily read and filtered

- ❑ Available via logcat view within Eclipse
- ❑ Accessible through adb utility using **logcat** command



```
Log.e("MyApp", "Something's wrong");
```



Logcat structure

➡ Logcat information is stored as a consistent structure

➡ Level

- Indicates importance/severity

➡ Time stamp


➡ Process and thread id

➡ Name of source Application

➡ Tag

- Application defined label
- Intended to identify system, component or method

➡ Text



Level	Time	PID	TID	Application	Tag	Text
D	05-02 18:48:51.671	642	737	com.android.exchange	ExchangeService	Received deviceId from Email app: null
D	05-02 18:48:51.671	642	737	com.android.exchange	ExchangeService	!!! deviceId unknown; stopping self and retrying
D	05-02 18:48:56.748	642	642	com.android.exchange	ExchangeService	!!! EAS ExchangeService, onStartCommand, startingUp = f
W	05-02 18:48:56.780	287	476	system_process	ActivityManager	Unable to start service Intent { act=com.android.email.
D	05-02 18:48:56.780	642	657	com.android.exchange	ExchangeService	!!! Email application not found; stopping self
W	05-02 18:48:56.808	287	427	system_process	ActivityManager	Unable to start service Intent { act=com.android.email.
E	05-02 18:48:56.808	642	642	com.android.exchange	ActivityThread	Service com.android.exchange.ExchangeService has leaked

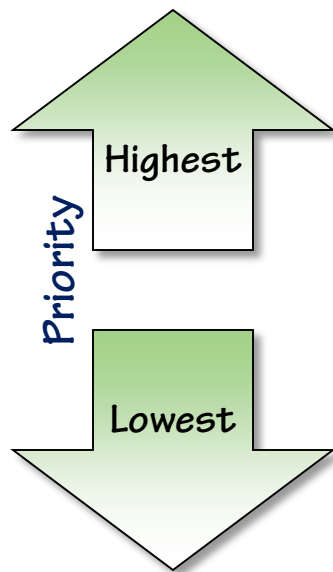
Logcat levels

➡ Levels are important to both creating and viewing logcat entries

➡ Indicate the relative importance of message

➡ Managed as a hierarchy

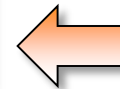
- Setting view filter to a level includes the level and all higher levels



Level	Label	Log method
Silent	FILTER ONLY	
Assert	A	Log.wtf
Error	E	Log.e
Warning	W	Log.w
Info	I	Log.i
Debug	D	Log.d
Verbose	V	Log.v

No messages
Can be written
at this level

What a
Terrible
Failure



Logcat buffers

Logcat segments messages into buffers

Overwhelming majority of messages go into the “main” buffer

- Standard Log class method calls write here
- All messages viewed from Eclipse are read from the “main” buffer

Two special purpose buffers available

- events: Shows all system events
- radio: Shows activity of system radios (cellular, etc.)

Only accessible through adb command-line utility

- Use **logcat** command with **-b** option followed by the buffer name

Summary

Android Debug Bridge is essential to Android development

- ❑ If not functioning correctly, nothing else will function well
- ❑ Can reset ADB server with adb ***kill-server*** command
- ❑ Can get list of connected devices with adb ***devices*** command
- ❑ If multiple connected devices/AVDs must provide include -d, -e, or -s

Eclipse is the hub of our development

- ❑ Logcat information available in a console-style view
- ❑ DDMS available through a perspective

Logcat is the Android logging system

- ❑ Levels indicate the importance/severity of the message
- ❑ Tag indicates the system, component, or method generating the message
- ❑ Logcat can be easily viewed and filtered
- ❑ Can view low-level information using the radio and events buffers