Logging in Android - LogCat

Log Meldung sind allgemein Programmausgaben, die über den (reibungslosen) Ablauf bzw. über Probleme informieren und hauptsächtlich für den Entwickler oder für die Problemsuche gedacht sind.

Wie können wir Log-Meldungen ausgeben?

1. Direkte Programmausgabe mit `System.out.println` bzw. `System.err.println`
2. Ausgabe mithilfe eines Logging Frameworks

Ausgabe mittels `System.xxx.println`

Diese Variante verführt vor allem unerfahrene Entwickler aufgrund seiner Einfachheit. Das Print-Statement ist einer der ersten Befehle, den der geneigte Java Entwickler erlernt. Doch ist dieser Befehl aus mehreren Gründen für die Ausgabe einer Log-Meldung **denkbar ungünstig**:

- Jede Ausgabe hat die gleiche Priorität. Ich kann nur zwischen den Streams `out` --> direkt auf die Konsole und `err` --> Fehler-Stream unterscheiden.
  Ein Fehler kann jedoch verschiedene Abstufungen zugeordnet sein (kritisch für die weiteren Programmablauf, leichter Fehler, reine Information für den Entwickler, etc.)
- Es ist nicht sicher, dass die Meldungen ausgegeben werden, da der Stream ja direkt auf die Konsole zeigt. Ist keine Konsole vorhanden, sind die Meldungen idR nicht sichtbar.
- Die Log-Meldungen werden während der Laufzeit des Programms ausgegeben und sind nach Programmende weg. (*Theoretisch könnte man natürlich den Stream umleiten, aber dies muss direkt beim Programmaufruf erfolgen.*) Will man jedoch im Nachhinein eine Fehleranalyse durchführen, benötigt man auch die historischen Log-Files.

Ausgabe mittels Logging Framework

Die Verwendung eines Logging-Frameworks löst diese Probleme. IdR kann das Framework konfiguriert werden:

- Welche Logmeldungen sollen angezeigt werden?
- Wohin erfolgt die Ausgabe? Konsole, Datei, etc.
- Wie oft werden Log-Files gelöscht (Log-Rotation)

Der Entwickler kann die Aufgaben der Verwaltung der Logs direkt an das Framework delegieren und muss sich nur noch um folgende Aspekte kümmern:

- *Sprechende* Logmeldungen erstellen
- Für jede Logmeldung, die ausgeben wird, den Level überlegen.

Das LogCat-Framework in Android

In Android können wir mit statischen Methoden der Klasse `Log` Log-Meldungen erzeugen:

Methode	Bedeutung	__
Log.v	verbose	reine Information
Log.d	debug	Information zur Fehlersuche / Debugging im Programm
Log.i	inform	Allgemeine Information, die auch zur Programmlaufzeit ausgegeben werden soll
Log.w	warning	Warnung, die jedoch nicht kritisch ist für die weitere Programmausführung
Log.e	error	Tatsächlicher Fehler. Die weitere Programmausführung ist nicht mehr sinnvoll möglich
Log.wtf	XXX	Fehler, der eigentlich nie auftreten dürfte

Dem Aufruf der Log-Methoden müssen zwei Parameter übergeben werden:

- ein *TAG*, der der Zuordnung der Log-Meldung dient. Dies ist in der Regel der Name der Klasse
- eine Logmeldung (message)

Beispiel:

TAG definieren wir als String Konstante in der Klasse

```
public final static String TAG = MyActivity.class.getSimpleName();
```

Nun können wir *TAG* für die Ausgabe der Logmeldungen verwenden:

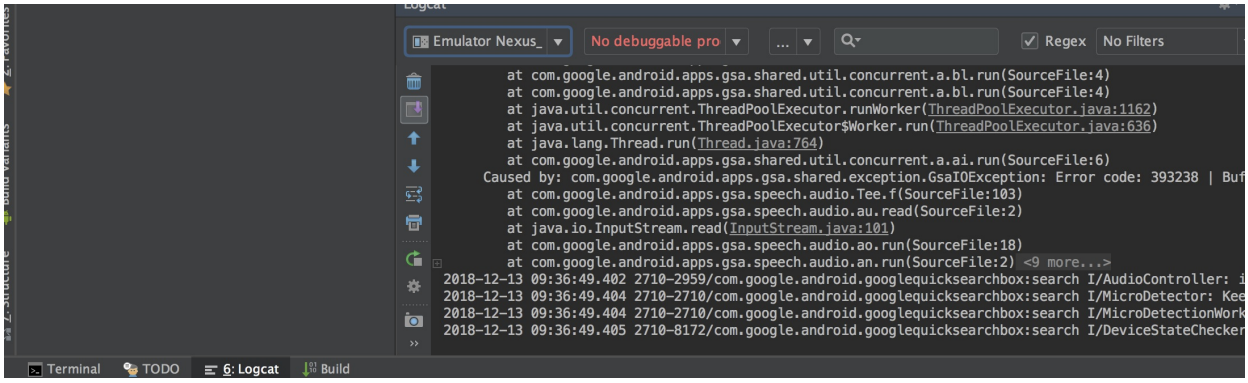
```
Log.d(TAG, "Asset File not found!");
```

In Android Studio kann die Eingabe durch Verwendung des Kürzels logd vereinfacht werden.

Android Monitor

Wenn wir die Android App auf dem Emulator laufen lassen, so treten die Fehlermeldungen nicht auf dem Entwicklungsrechner, sondern direkt auf dem Android-Emulator bzw. dem angeschlossenen Device auf.

Daher müssen die Log-Meldungen umgeleitet werden. Dies ist in Android Studio mit dem Device-Monitor gelöst. Standardmäßig sehen wir die Logmeldungen im Fenster *LogCat*:



Hier kann ich das Gerät bzw. den Emulator auswählen (sofern mehrere Geräte gleichzeitig aktiv sind) und entsprechende Filter setzen, um einen leichteren Überblick über die Log-Meldungen zu erhalten.

Aus diesem Grund ist die Verwendung des Klassennamen als Log-TAG sinnvoll. Denn so kann ich gezielt nach Meldungen suchen, die genau aus der betroffenen Klasse stammen.

Links

https://developer.android.com/studio/debug/am-logcat