Journal de débogage Android
« Logcat »

Introduction

On veut suivre l'état d'exécution de l'application afin de s'assurer de son bon fonctionnement.

Android ne dispose pas d'une console d'exécution dans laquelle nous pouvons afficher des traces sur l'état d'avancement. Exemple : utilisation de « System.out.println » et que le résultat s'affiche sur une console.

Android propose à la place un journal des évènements en cours dans votre appareil.

Ce journal fournit donc un moyen pour déboguer l'application.

Ce journal est appelé « Logcat » en raison du nom de la commande associée

https://developer.android.com/tools/logcat

Le « Logcat » n’affiche pas de messages si l’appareil n’est pas connecté à un outil de développement comme « Android Studio ».

« Logcat » fait partie des outils d’ « adb ». Il vous permet de :

- Voir, filtrer, collecter toutes les traces de l’application.
- Voir, filtrer, collecter toutes les traces du système.
- Récupérer toutes les erreurs inattendues générées au moment de l’exécution de l’application.

Accès au « Logcat »

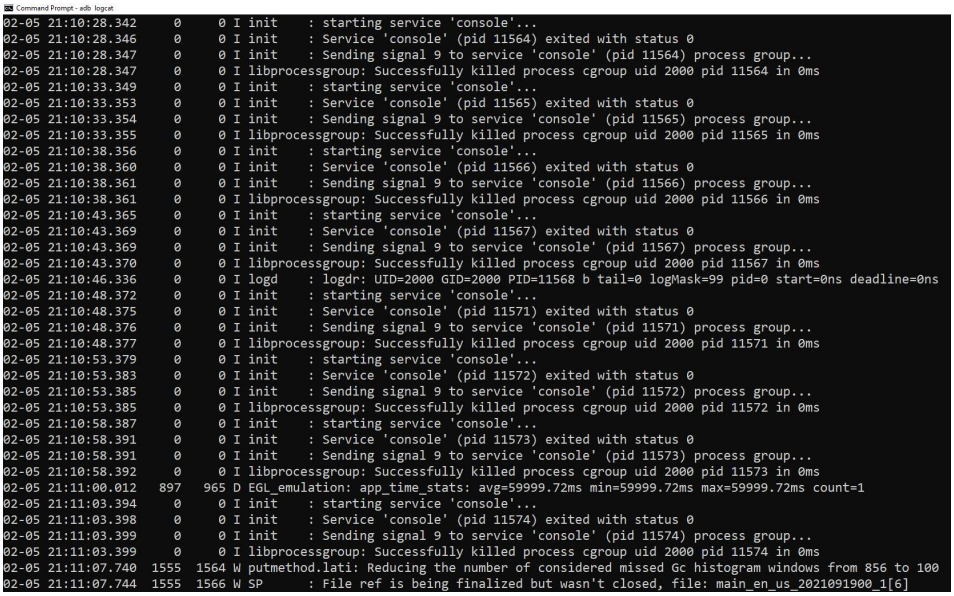
Le « Logcat » prend en charge les téléphones, tablettes, objets connectés, les émulateurs.

Pour un appareil physique, le mode développeur doit-être activé afin d’avoir accès aux traces générées dans le « Logcat ».

Commencer par connecter un appareil Android à votre ordinateur ou bien démarrer un émulateur.

Par la suite, dans une console de votre système (« DOS » ou un « terminal »), il suffit de taper la commande « adb logcat ».

Vous allez obtenir « en mode direct », toutes les traces générées par votre appareil.



La capture est en temps réel.

Utilisation des filtres

« Logcat » génère une très grosse quantité de traces.

La trace recherchée au moment du débogage risque d’être noyée dans cette quantité d’informations.

Les filtres viennent à la rescousse pour permettre de ne garder que les traces « pertinentes », recherchées.

Les messages sont organisés par niveau de priorités. Ils sont regroupés dans 4 catégories et 7 niveaux :

- Catégorie « développer et déboguer » :

« V » : Verbose (bavard)
« D » : Debug (débogage)

- Catégorie « audit » :

« I » : Info (information)

- Catégorie « erreurs » et « suivi de production » :

« W » : Warning (avertissement)
« E » : Error (erreur)
« F » : Fatal (fatal)

- Catégorie « autre » :

« S » : Silent (mode silence, permet de faire un 2e niveau de filtrage)

Exemples

Prendre comme repère le flux global et prendre un exemple en particulier pour lui expliquer les commandes ci-dessous.

1/

```
02-05 21:27:53.731  438  438 E android.hardware.power.stats@1.0-service-mock: Failed to getEnergyData
```

Ce message a une priorité « E » pour erreur et est tagué à l'application « android.hardware.power.stats ».

2/

```
adb logcat *:W
```

On affiche toutes les traces dont le niveau « W » (avertissement).

3/

Le mode silence peut-être utilisé comme suit :

```
adb logcat "ProcessReaper:W" "*:S"
```

On se contente d'afficher toutes les traces du niveau « W » associées à l'application « ProcessReaper » et on masque toutes les autres traces. Ainsi on ne concentre notre attention que sur les avertissements générés par notre application. On évite ainsi que ces avertissements soient noyés dans le volume de traces affichées.

4/

```
adb logcat "ProcessReaper:W" "init:I" "*:S"
```

On suit à la trace le niveau « I » (information) pour l'application « init » et le niveau « W » (Avertissement) pour l'activité « ProcessReaper » et on met en mode silence toutes les autres traces.

Filtres utiles

Sauvegarder tous les logs dans un fichier : `adb logcat -f <nom du fichier en sortie>`

Obtenir toutes les erreurs fatales : `adb logcat "*:E"`

Filtrage supplémentaire avec les commandes Linux :

```
adb logcat | grep -i "ProcessReaper" #obtenir tous les logs relatifs au tag “ProcessReaper”
```

Filtrage par le nom de l’application : `adb logcat "application_or_tag_name:*" "*:S"`

Obtenir tous les changements relatifs à l’état du « GSM » :

```
adb logcat -b events "gsm_service_state_change" "*:S"
```

Obtenir tous les évènements « radio » : `adb logcat -b radio`

Association classe « log » avec « Logcat »

Le langage Java dispose d’une classe native pour le traitement des « log ».

Cette classe a été introduite dans Android depuis l’API 1.

https://developer.android.com/reference/android/util/Log

On déclare une constante dans le programme Java

```
private static final String TAG = "MyActivity";
```

On utilise cette constante par la suite dans la méthode « Log », comme suit :

```
Log.v(TAG, "index=" + i);
```

On ajoute par la suite un filtre dans l’interface « Logcat » d’Android Studio pour filtrer les messages en fonction de ce tag. On peut filtrer aussi sur une console de commandes, à l’aide d’adb.

On dispose ainsi des méthodes suivantes :

- `Log.v(String, String)` (verbose)
- `Log.d(String, String)` (debug)
- `Log.i(String, String)` (information)
- `Log.w(String, String)` (warning)
- `Log.e(String, String)` (error)

Exemple

```
Log.i("MyActivity", "MyClass.getView() — get item number " + position);
```

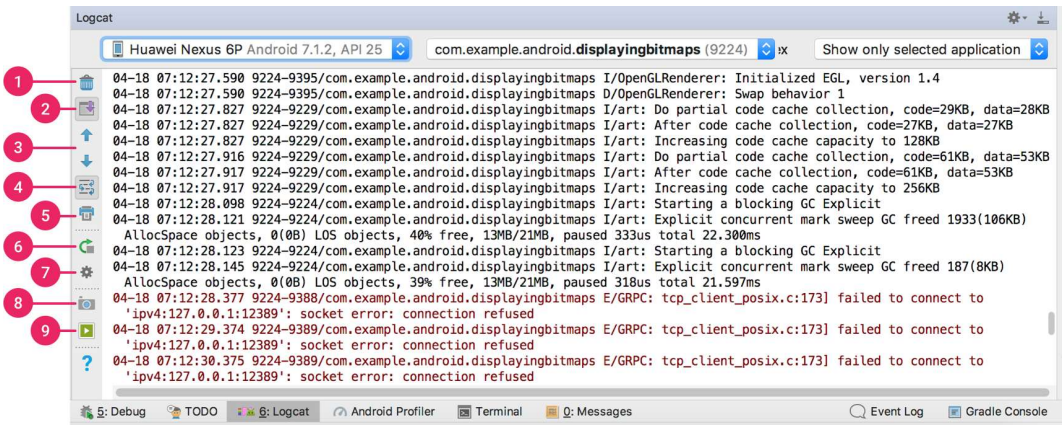
Affichage en sortie

```
I/MyActivity( 1557): MyClass.getView() — get item number 1
```


Démo

Utilisation de « Logcat » dans Android Studio

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



La barre des outils associée au « Logcat » contient ce qui suit :

1. **Clear logcat**  : on efface le log visible.
2. **Scroll to the end**  : on saute jusqu'à la fin du log afin de voir les derniers messages.
3. **Up the stack trace**  and **Down the stack trace**  : pour naviguer dans la pile des traces.
4. **Use soft wraps**  : pour couper la longueur de la ligne affichée et la fixer sur la largeur de la fenêtre.
5. **Print**  : pour imprimer les logs.
6. **Restart**  : effacer le log et redémarrage du "Logcat".
7. **Logcat header**  : pour configurer l'entête des messages affichés.
8. **Screen capture**  : pour prendre une capture d'écran (photo) de l'écran de l'appareil.
9. **Screen record**  : pour enregistrer une vidéo (maximum 3 minutes).

Ouvrer dans Android Studio, le projet « MyPhoneReceiver » du chapitre 02.

Ne pas oublier d'ajouter les permissions nécessaires.

Utiliser l'interface « Logcat » dans Android Studio et filtrer le tag « MY_DEBUG_TAG ».

Maintenant, nous allons faire la même chose avec « adb ».

Pour avoir le même filtrage que dans l'interface, nous avons ces deux options :

```
adb logcat -s "z1MY_DEBUG_TAG" -s  "z2MY_DEBUG_TAG" -s  "z3MY_DEBUG_TAG"
```

```
adb logcat "z1MY_DEBUG_TAG:W" "z2MY_DEBUG_TAG:W" "z3MY_DEBUG_TAG:W" "*:S"
```

Références

https://developer.android.com/tools/logcat

https://developer.android.com/reference/android/util/Log

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