Programmation Android

Debugger ses applications

Jérémy VINET

IE-Concept 2017

Debugger ses applications

- Le concept de débogage
- DDMS
- Utilisation d'un logeur
- Utilisation du débuggeur (ADB)
- Ressources

Le concept débogage

- Le débogage d'une application permet au développeur de tester et analyser les réactions de l'application en fonction de son utilisation.
- Il est couramment admis qu'un développeur va passer au moins 50% de son temps de développement à débugger son application.
- Depuis l'avènement d'internet et le partage massif d'application, il est devenu inconcevable pour un éditeur d'application de ne pas corriger/améliorer son application dans le temps.

Le concept débogage

- **Avec le temps le concept de débogage a évolué afin de correspondre aux nouveaux usages des applications.**
  - Alpha et beta test
  - Sortie de version preview public
  - Envoi de reporting vers un serveur
  - Bug tracker

- **Si le concept a évolué, la finalité reste la même :**
  - Recherche de bug.
  - Amélioration ergonomique.
  - Amélioration technique/stabilité.
  - Test de nouveau modèle.

Le concept débogage

- **Dans le cadre d'une application Android le débogage peut se faire de plusieurs méthodes :**
  - Utilisation d'un logeur pour avoir une traçabilité de l'application.
  - Utilisation du débuggeur intégré (ADB).
  - Retour des utilisations de l'application via les commentaires sur le marché.
  - Test Unitaire et test d'intégration.
  - Android Profiler

- **Dans le cadre de ce cours nous verrons seulement les deux premières méthodes.**
  - Logeur
  - ADB

DDMS

- DDMS : Dalvik Debug Monitor Server.
- C'est une application graphique de profilage d'applications Android.
- DDMS est présente dans le SDK Android.
- Android studio intègre parfaitement DDMS pour permettre le débogage des applications plus facilement.
- Possibilité de le lancer en mode « lignes de commandes ».

DDMS

- **Parmi les outils de DDMS, nous pouvons citer :**
  - La vue « Devices » : elle permet de sélectionner le device (virtuel ou non) a monitorer.
  - La vue « LogCat » : elle présente les logs du device sélectionné dans la vue « Devices ».
  - La vue « Console » : elle présente des informations sur l'application en cours d'exécution pour le device sélectionné.
  - La vue « FileExplorer » : elle permet de naviguer dans le système de fichiers du device sélectionné.
  - La vue « Emulator » : elle permet de contrôler votre émulateur.
    - Par exemple, elle permet de contrôler les données de localisation, …
- **DDMS est un composant d'un outil qui s'appelle Android Device Monitor qui lui-même permet le débogage d'application en profondeur.**

Utilisation d'un logeur

- Qu'est ce qu'un logeur ?
- Le logeur d'Android Logcat

Qu'est ce qu'un logeur ?

- Un logeur est soit une classe soit un service qui a pour but de lister et mémoriser des évènements.
- Il permet de voir le déroulement de l'application et le cas échéant les problèmes lors de l'exécution de celle-ci.
- Il n'a pas pour but d'apporter une fonction en plus à l'utilisateur final.
- L'application doit pouvoir fonctionner sans ressentir de différences que le logeur soit activé ou éteint.

Qu'est ce qu'un logeur ?

- En règle générale le logeur est vu comme un tableau ou chaque ligne représente un évènement ou une action que l'application a exécuté.

- Chaque ligne doit contenir des informations utiles permettant d'analyser l'application.
  - Date et heure précise de l'évènement
  - Un nom humainement compréhensible, par le développeur, de l'évènement
  - Des informations concernant l'évènement (variables, choix, ..)

- Le logeur doit avoir un flux de sortie (console, fichier, mail, …) afin que l'on puisse accéder à ces données.

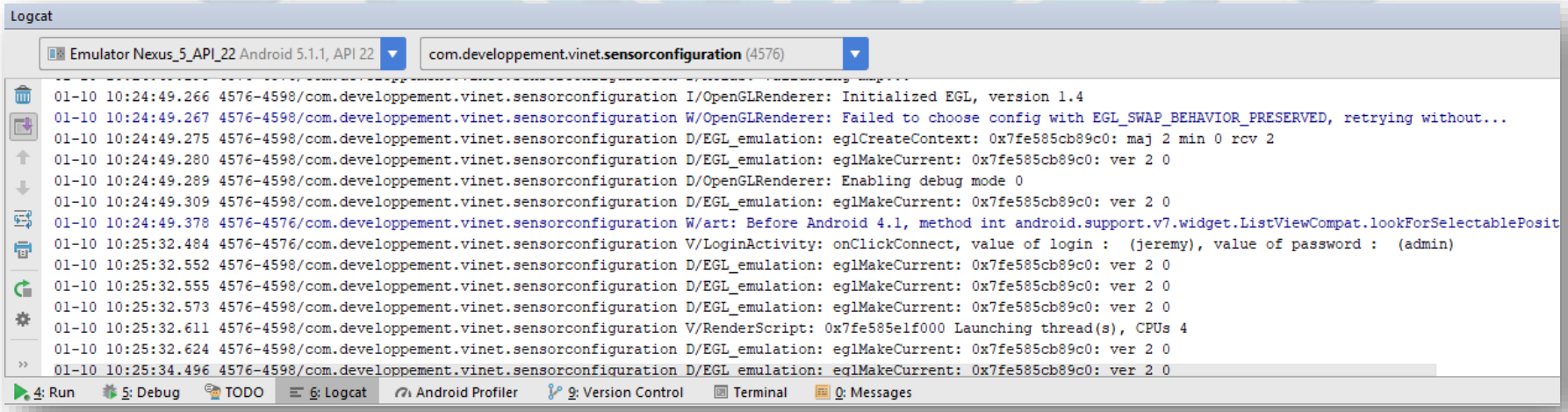
Qu'est ce qu'un logeur ?

- Le logeur étant un outil indispensable pour un développeur d'application, une classe Log a été intégrée dans le SDK Android.
- Cette classe de log permet la remonté des informations de logeur dans une console dédiée (DDMS -> LogCat).
- Elle fonctionne aussi bien avec le simulateur qu'avec un périphérique réel.

Le logeur d'Android Logcat

- Le SDK Android intégre la classe « Log » qui permet la remonté d'informations dans la console « Logcat ».

- La console Logcat affiche toutes les informations provenant du périphérique, pas juste celle de l'application en cours.



Le logeur d'Android Logcat

- Le logcat permet donc d'avoir énormément d'informations sur l'état du périphérique sans même rajouter des informations de log.
- Pour rajouter des informations de log il faut utiliser la classe « Log ».
- Elle n'a pas besoin d'être instanciée, les méthodes permettant d'afficher un message log étant static.

```
Log.i(TAG_DEBUG,  msg: "onCreate");
```

Le logeur d'Android Logcat

- Afin de pouvoir classer les messages par ordre de priorité , il existe plusieurs méthodes à la classe « Log » pour afficher des messages.

- Il est possible, dans le Logcat, de filtrer les messages en fonction du niveau de priorité voulu.

- Liste des commandes les plus courantes :
  - Log.v -> niveau verbose
  - Log.d -> niveau débug
  - Log.i -> niveau information
  - Log.w -> niveau warning
  - Log.e -> niveau error
  - Log.wtf -> niveau fatal (What a Terrible Failure)

Le logeur d'Android Logcat

- Afin d'avoir un bon niveau de traçabilité, lors d'une utilisation d'une commande de log, il est fortement conseillé d'utiliser un TAG.

- Un TAG peut, par exemple, être le nom de la classe dans laquelle s'exécute l'évènement de log.

```
private final String TAG_DEBUG   =   "LoginActivity";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Log.i(TAG_DEBUG,  msg: "onCreate");
```

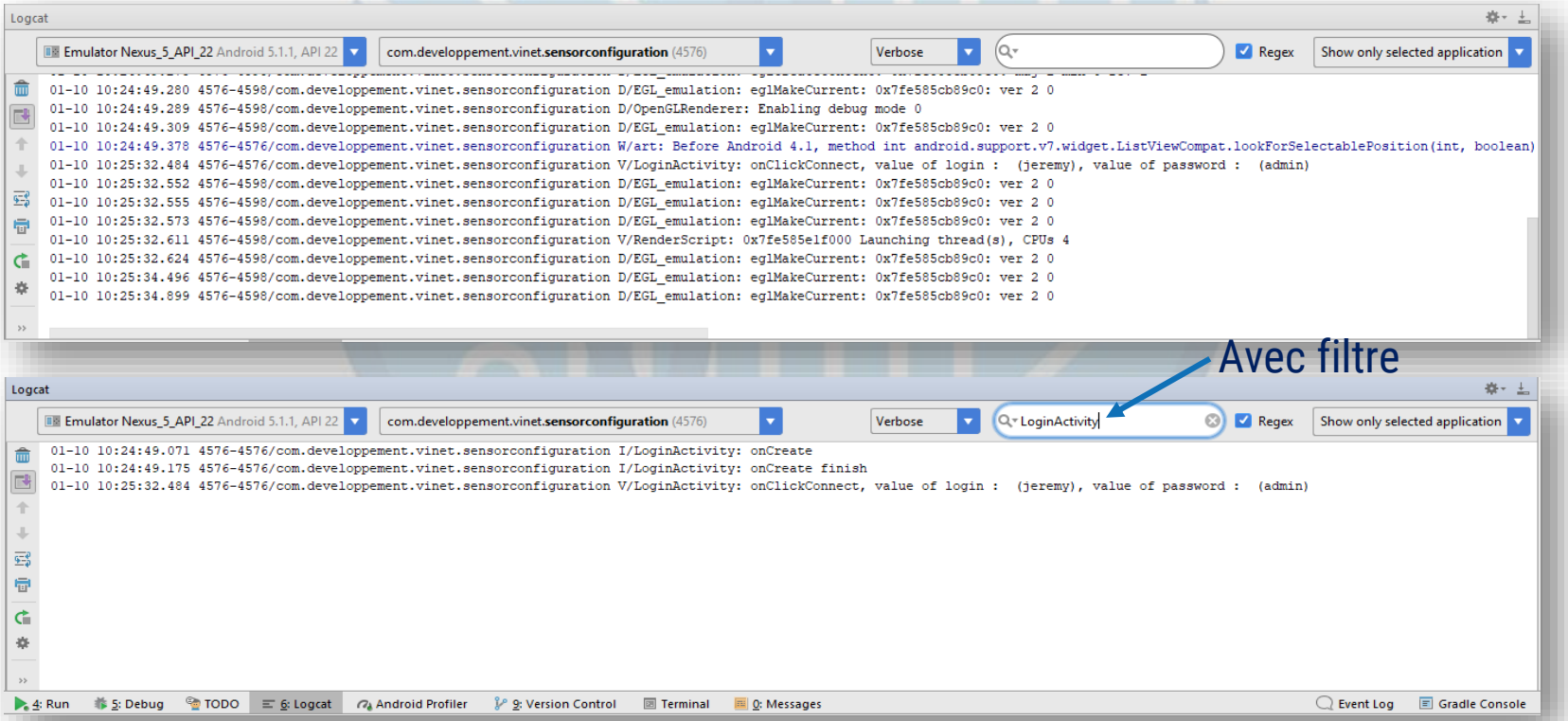
Le logeur d'Android Logcat

- Le logeur prend en paramètre des string, il est donc tout à fait possible d'afficher le contenu d'une variable à condition de l'avoir converti en string juste avant.

```
//--- Load ressources
m_editTextLogin     = (EditText)    findViewById(R.id.editTextLogin);
m_editTextPassword  = (EditText)    findViewById(R.id.editTextPassword);
m_btnConnect        = (Button)      findViewById(R.id.btnConnect);
//---
m_btnConnect.setOnClickListener((v) → {
        Log.v(TAG_DEBUG,  msg: "onClickConnect, value of login : "
                +m_editTextLogin.getText().toString()+" ("+ADMIN_LOGIN+"), value of password : "
                +m_editTextPassword.getText().toString()+" ("+ADMIN_PASSWORD+")");
```

Le logeur d'Android Logcat

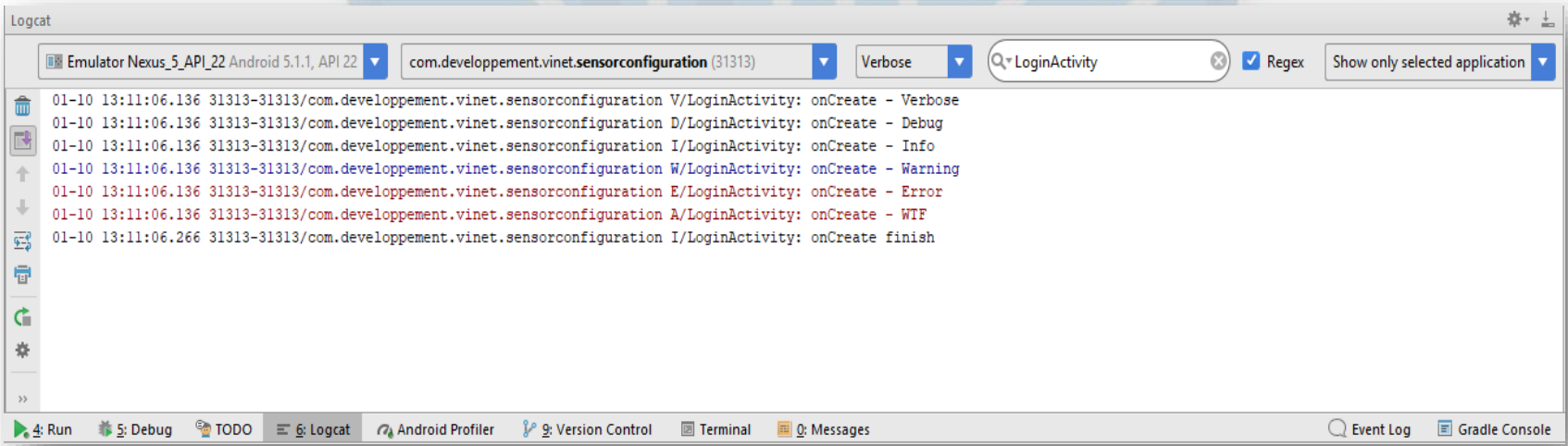
- Une fois dans la console Logcat d'Android studio on peut venir filtrer sur le TAG afin d'avoir seulement les messages comportant ce TAG.



Le logeur d'Android Logcat

- Exemple de rendus en utilisant les 6 niveaux de log possibles

```
Log.v(TAG_DEBUG, msg: "onCreate - Verbose");
Log.d(TAG_DEBUG, msg: "onCreate - Debug");
Log.i(TAG_DEBUG, msg: "onCreate - Info");
Log.w(TAG_DEBUG, msg: "onCreate - Warning");
Log.e(TAG_DEBUG, msg: "onCreate - Error");
Log.wtf(TAG_DEBUG, msg: "onCreate - WTF");
```



Utilisation du déboguer (ADB)

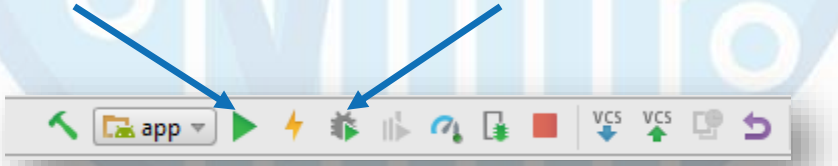
- Si le logeur est un outil très utile, il est parfois insuffisant pour déboguer en profondeur son application.
- Android possède parmi tous ces outils un déboguer du nom de ADB
- ADB : Android Debug bridge
- Il permet, en mode débug, de se connecter à une application afin de pouvoir gérer des points d'arrêt et l'analyse de la mémoire du device.

Utilisation du déboguer (ADB)

- L'intégration de ADB dans Android studio est complète, pas de ligne de commande à lancer.
- Pour débugger une application, il faut la lancer en mode debug.

Lancement de l'application en mode normal

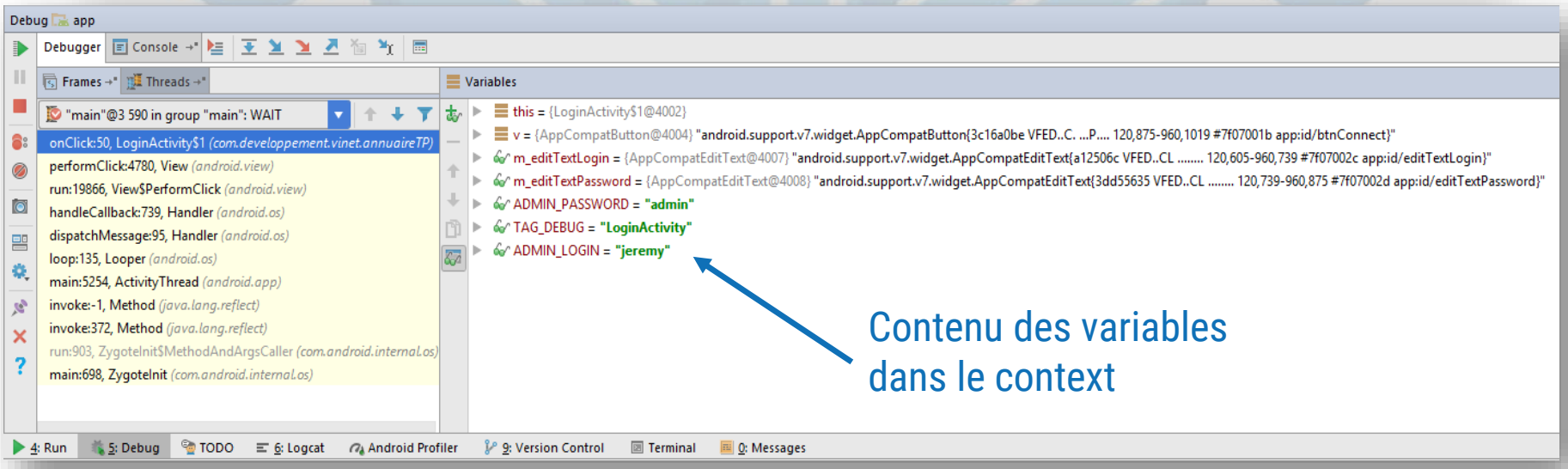
Lancement de l'application en mode debug



Utilisation du déboguer (ADB)

- L'utilisation de point d'arrêt permet de bloquer l'exécution du programme le temps d'analyser le contenu des variables et autres données utiles.

```
49          //--- check login
50          if(m_editTextLogin.getText().toString().equals(ADMIN_LOGIN))
51          {
```



Utilisation du déboguer (ADB)

- Le déboguer permet aussi d'exécuter du code en mode pas à pas.
- Ce qui permet de vérifier la manière dont le code s'exécute et donc comment votre application réagit.
- ADB permet une grande souplesse pour déboguer une application
- L'utilisation d'un logeur et d'ADB permet de répondre à la grande majorité des cas de débogage d'application.

Ressources

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