



IN01

Programmation Android

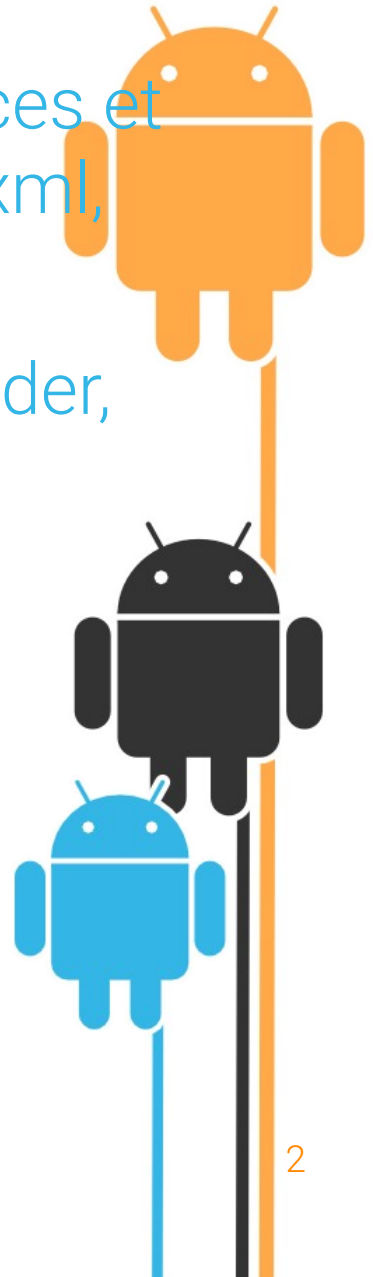
02 – La plateforme Android

Yann Caron
Et Jean-Marc Farinone

ENSG
Géomatique

Sommaire - Séance 02

- Anatomie d'un projet Android : Manifest, Resources et qualifications, R.java, Internationalisation string.xml, Assets, Libraries
- Les composants : Activity, Service, Content Provider, Broadcast Receiver
- Cycle de vie d'une application et persistance
- Un premier projet
- Exécution et adb
- Logcat



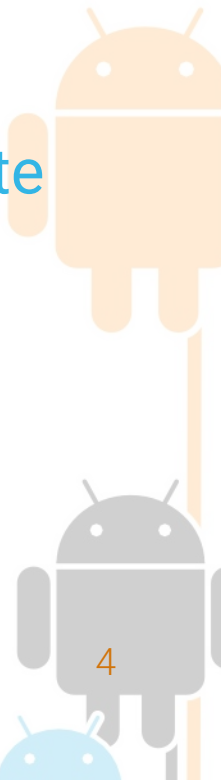
IN01 – Séance 02

Anatomie d'un projet



Workflow

- Les phases sont :
 - ➔ Installation de l'environnement (setup)
 - ➔ Développement (sources et ressources)
 - ➔ Débogage et tests (DDMS et JUnit) : Test Driven Development ?
 - ➔ Publication sur le Google Play (nécessite une clé cryptographique propre au fournisseur (nous), et un compte Google Developer)
- Plus de détails :
<http://developer.android.com/tools/index.html>

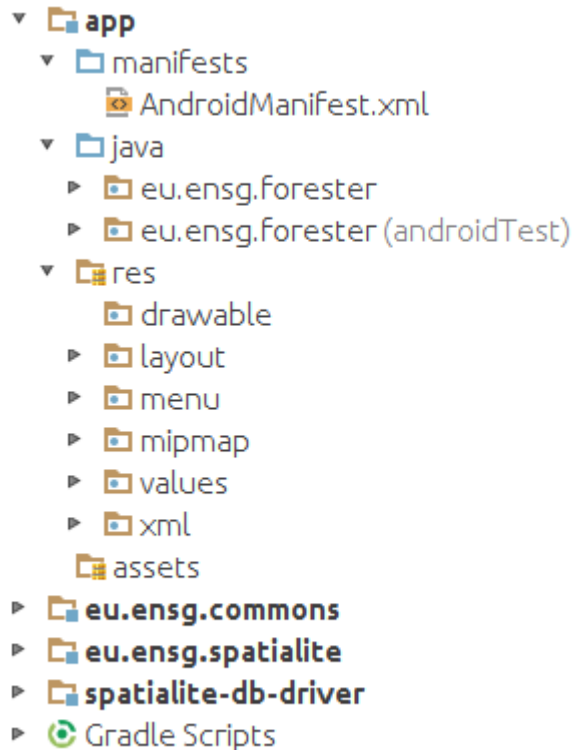


Concepts de base

- Les programmes sont écrits en Java (ce n'est pas un hasard ! C'est un des langages les plus utilisés)
- ADK compile l'ensemble du projet (code Java, données, fichier de ressources, fichier XML) dans un package APK (une application)
- La JVM = stack based virtual machine
- La Dalvik = register based virtual machine
- Le Dexer convertit les JAR d'un bytecode à l'autre (et ça marche plutôt bien, 80 % du code Algoid est compatible PC / Android)



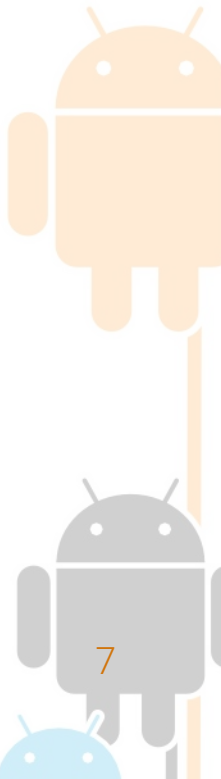
Anatomie d'un projet



- java : les sources Java
 - ➔ Packages et .java
- res : les ressources du projet
- assets : fichiers ressources additionnels et non compilés
- build : le résultat de la compilation
- AndroidManifest.xml

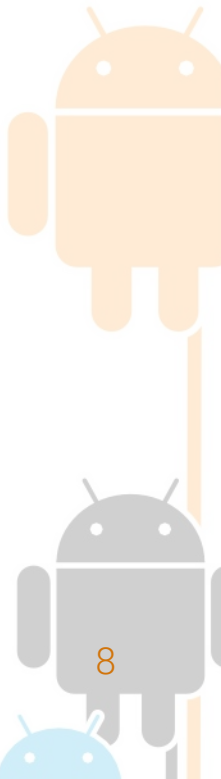
Manifest

- Fichier obligatoire (point d'entrée) ne doit pas être renommé ni déplacé
- Définit le nom de l'application, son icône, son package (identifiant unique sur le Play Store)
- Définit les composants (activities, services, broadcast receivers et content provider)
- Précise quels composants peuvent accueillir l'application



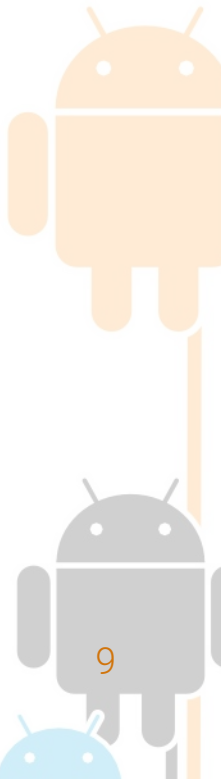
Manifest

- Déclare les permissions de l'application en interaction avec le matériel ou d'autres applications
- Déclare les permissions des autres applications avec les composants de l'application
- Des informations pour l'instrumentation (kesako ?)



Manifest

- Définit les niveaux de compatibilité (notamment l'API minimum pour que l'application s'exécute)
 - ➔ Exclut d'anciens appareils qui ne seraient pas compatibles
- Liste les “libraries” (Android) nécessaires
- Définit les intents



AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="fr.cnam.in01.db4o"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="19" />

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="fr.cnam.in01.db4o.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

API level

Permissions

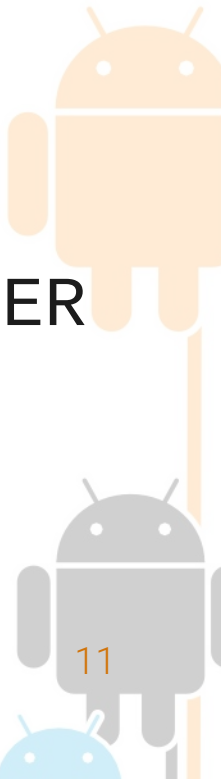
Activities

Main Activity !



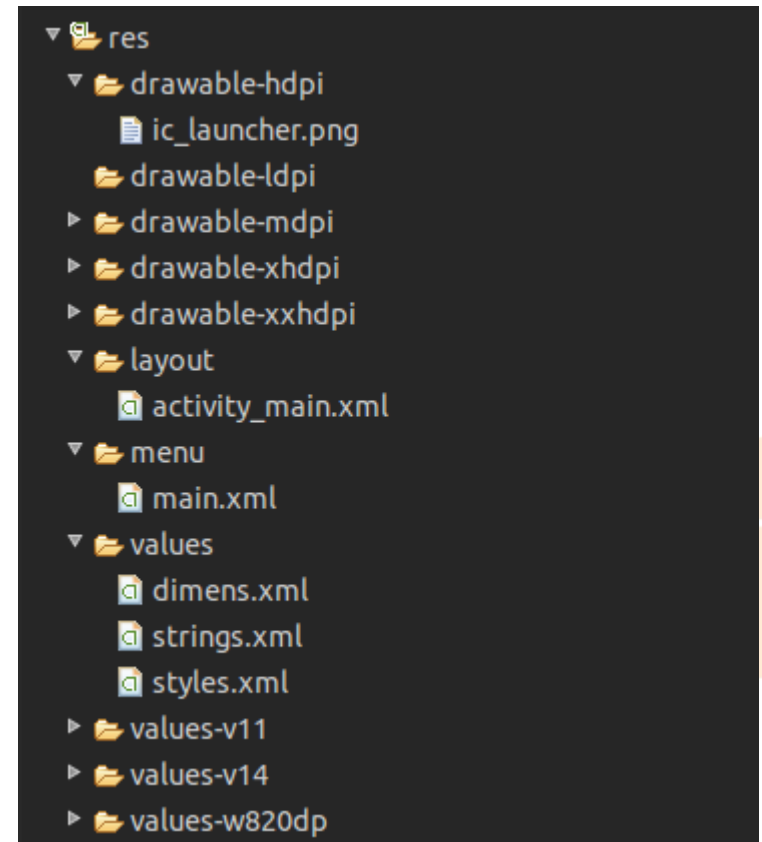
Manifest

- Toutes les activities du projet doivent être référencées dans le Manifest
- Sinon, elles ne fonctionnent pas ! Exception !
- L'activity Main (le point d'entrée du programme) est déclarée grâce à un intent-filter
 - Action : `android.intent.action.MAIN`
 - Catégorie : `android.intent.category.LAUNCHER`
- Attention !! Un seul point d'entrée par projet



Ressources

- drawable : images ou XML descriptifs
- layout : les vues (MVC basé sur une description XML)
- menu
- values : constantes nécessaires à l'application : strings, dimensions, styles, couleurs



Organisation

- Organisation des ressources selon le matériel (ldpi, v14, etc.)
- Ordre de priorité des **qualificateurs**
- Les paramètres par défaut sont ceux dans le répertoire sans modificateur

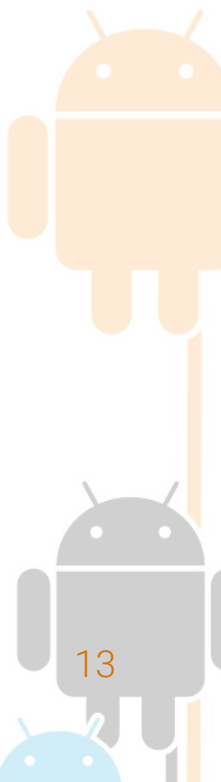
→ ex. :

- layout-land

Disposition de l'écran lorsque l'appareil est en mode paysage

- layout

Défaut (portrait)



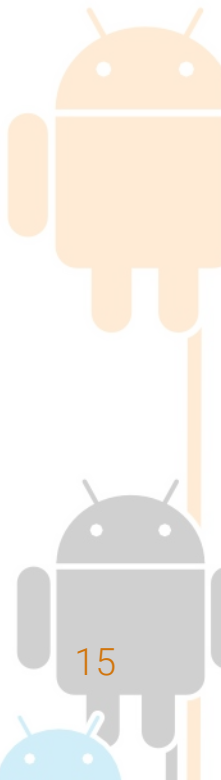
Qualificateurs

Priorité 2	Priorité 3	Priorité 6	Priorité 8	Priorité 14
Langue	Taille de la diagonale	Orientation	Densité	Version de l'API
fr	small	port	ldpi 120 dpi	v3
en	normal	land	mdpi 160 dpi	v5
en-US	large		hdpi 240 dpi	v7
en-UK	xlarge		xhdpi 320 dpi (\geq API 8)	v8
etc			nodpi non redimensionné	etc

- ex. : des images pour de petits écrans en portrait de petite densité
 - ➔ **drawable-small-port-ldpi**

Arborescence type

- drawable-ldpi
- drawable-mdpi
- drawable-hdpi
- drawable-xhdpi
- drawable-xxhdpi
- layout-land
- layout
- menu-fr
- menu
- values-fr
 - string.xml
- values-v11
 - style.xml
- values-w820dp
 - dims.xml
- values
 - string.xml
 - style.xml
 - dims.xml



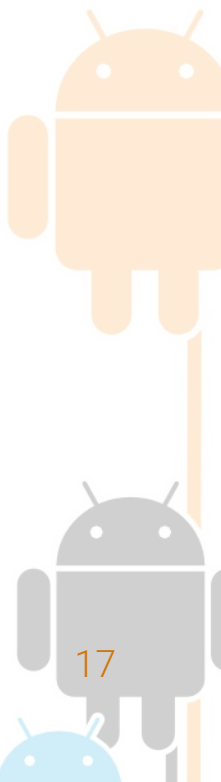
R.java

```
public final class R {  
    public static final class attr {}  
  
    public static final class dimen {  
        public static final int padding_large = 0x7f040002;  
        public static final int padding_medium = 0x7f040001;  
        public static final int padding_small = 0x7f040000;  
    }  
  
    public static final class drawable {  
        public static final int ic_launcher = 0x7f020001;  
    }  
  
    public static final class id {  
        public static final int menu_settings = 0x7f080000;  
    }  
  
    public static final class layout {  
        public static final int activity_main = 0x7f030000;  
    }  
  
    public static final class menu {  
        public static final int activity_main = 0x7f070000;  
    }  
  
    public static final class string {  
        public static final int app_name = 0x7f050000;  
        public static final int hello_world = 0x7f050001;  
    }  
  
    public static final class style {  
        public static final int AppTheme = 0x7f060000;  
    }  
}
```

```
▼ res  
  ▼ drawable-hdpi  
    ic_launcher.png  
  ▼ drawable-ldpi  
  ▼ drawable-mdpi  
    ic_launcher.png  
  ▼ drawable-xhdpi  
    ic_launcher.png  
  ▼ drawable-xxhdpi  
    ic_launcher.png  
  ▼ layout  
    activity_main.xml  
  ▼ menu  
    main.xml  
  ▼ values  
    dims.xml  
    strings.xml  
    styles.xml  
  ► values-v11  
  ► values-v14  
  ► values-w820dp
```

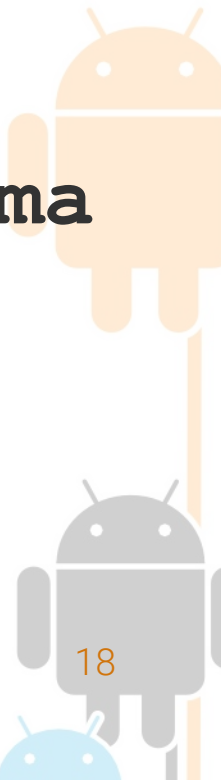

R.java

- Généré automatiquement lors de la phase de compilation du projet
- Référence toutes les ressources du projet et les rend utilisables depuis Java
- Association identifiant (**`R.string.app_name`**) / index (**`0x7f050000`**)
 - ➔ Optimise : entiers plus efficaces que des strings (hashage)
 - ➔ Autocomplétion : accessible depuis les sources Java



Les vues

- Les ressources se trouvent dans le répertoire `res/layout`
- Disponible depuis la class `R.java` depuis la classe interne `R.layout.*`
- Utilisable dans le projet avec `setContentView(R.layout.activity_main)`



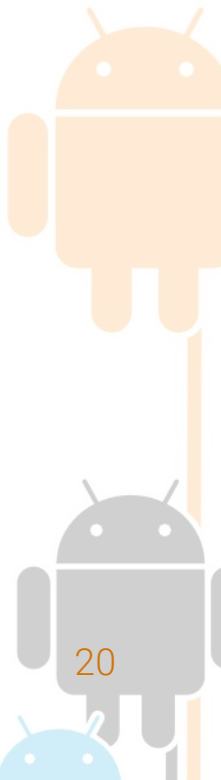
Les composants et les id

- Les composants graphiques présents dans les layouts sont identifiés par un id
- Ils sont créés dans le fichier XML de layout comme suit : **`android:id="@+id/myTextView"`**
- **`@+id`** signifie "créer l'id"
- Ils sont accessibles depuis les sources Java avec l'instruction **`findViewById(R.id.myTextView)`**
- **ex. :** `myTextView = (TextView) findViewById(R.id.myTextView);`



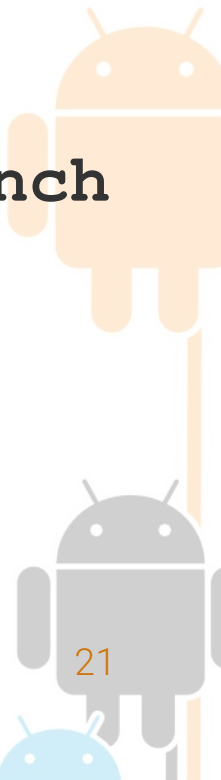
Internationalisation : String.xml

- Les ressources se trouvent dans le fichier `res/values/strings.xml`
- Répertoires différents selon la langue (`values`, `values-fr`, `values-en`, `values-fr-rCA`)
- Classe interne `R.string.*`
- Utilisable dans le projet avec `getString(R.string.hello_world)`
- Utilisable depuis les fichiers XML `android:text="@string/hello_world"`



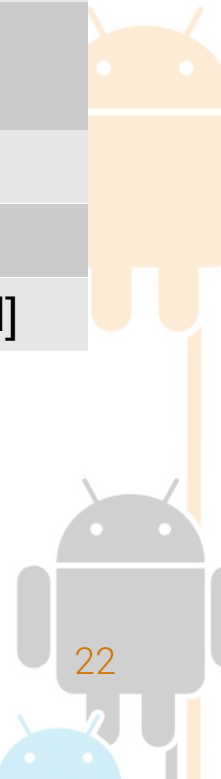
Multirésolutions : les images

- Les ressources se trouvent dans le répertoire `res/drawable` et dérivés
- Répertoires différents selon la résolution (`drawable-ldpi`, `drawable-mdpi`, `drawable-hdpi`)
- Classe interne `R.drawable.*`
- Utilisable dans le projet avec `getResources().getDrawable(R.drawable.ic_launcher)`
- Utilisable depuis les fichiers XML `android:icon="@drawable/ic_launcher"`



Résumé

	Source res/values/*	[id]	Java getResources().*	XML
bool	<bool>	R.bool.*	getBool([id])	@bool/[id]
integer	<integer>	R.integer.*	getInteger([id])	@integer/[id]
array	<integer-array><item> <string-array><item>	R.array.*	getIntArray([id]) getStringArray([])	@array/[id]
string	<string>	R.string.*	getString([id])	@string/[id]
types array	<array><item>	R.array.*	obtainTypedArray([id])	@array/[id]
color	<color>	R.color.*	getColor([id])	@color/[id]
style	<style><item>			@style/[id]
dimen	<dimen>	R.dimen.*		@dimen/[id]



Résumé

	Source	[id]	Java	XML
id		R.id.*	findViewById([id])	@+id/[id]
IHM	res/layout/*	R.layout.*	setContentView([id])	
dessins images	res/drawable/*	R.drawable .*	getResources(). getDrawable([id])	@drawable/ [id]
anim	res/anim/*	R.anim.*		@anim/[id]
menu	res/menu/*	R.menu.*	getMenuInflater().inflat e([id])	@menu/[id]
style	res/values/* <	R.style.*		@style/[id]

- Liste exhaustive :

<http://developer.android.com/guide/topics/resources/available-resources.html>

Exemple d'utilisation de R.java

```
public class MainActivity extends ActionBarActivity {  
  
    TextView myTextView = null;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        // le layout et les vues  
        setContentView(R.layout.activity_main);  
        myTextView = (TextView) findViewById(R.id.myTextView);  
        String text = getString(R.string.hello_world);  
  
        // les images  
        Drawable d = getResources().getDrawable(R.drawable.ic_launcher);  
        ImageView image = (ImageView) findViewById(R.id.image);  
  
        // autres  
        getResources().getColor(R.color.abc_search_url_text_holo);  
        getResources().getDimension(R.dimen.dialog_fixed_height_major);  
    }  
  
    @Override  
    public boolean onCreateOptionsMenu(Menu menu) {  
        getMenuInflater().inflate(R.menu.main, menu);  
        return true;  
    }  
  
    // etc.
```

The diagram illustrates the usage of R resources in the provided Java code. It features seven labels in orange boxes: 'Layout', 'View', 'String', 'Image', 'Color', 'Dimension', and 'Menu'. Arrows point from these labels to the corresponding R resource identifiers in the code: 'R.layout.activity_main' (Layout), 'R.id.myTextView' (View), 'R.string.hello_world' (String), 'R.drawable.ic_launcher' (Image), 'R.color.abc_search_url_text_holo' (Color), 'R.dimen.dialog_fixed_height_major' (Dimension), and 'R.menu.main' (Menu).

Context

- Les méthodes `setContentView`, `findViewById`, `getString`, `getResources` et `getMenuInflater` sont des méthodes héritées de la classe `Context`, dont hérite `Activity` à son tour.

Activity (view source)

extends `ContextThemeWrapper`

implements `ComponentCallbacks2` `KeyEvent.Callback` `LayoutInflater.Factory2` `View.OnCreateContextMenuListener` `Window.Callback`

`java.lang.Object`

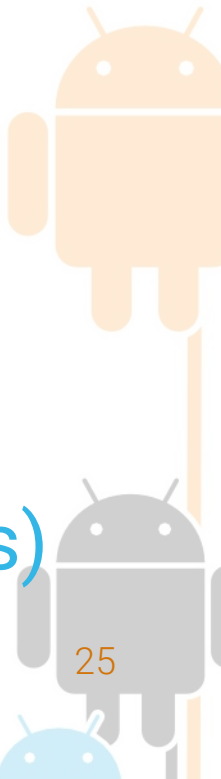
↳ `android.content.Context`

↳ `android.content.ContextWrapper`

↳ `android.view.ContextThemeWrapper`

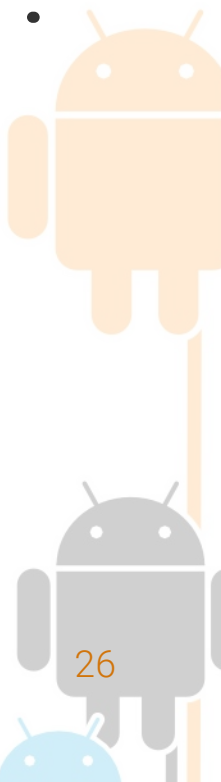
↳ `android.app.Activity`

- Dans d'autres classes, le contexte est passé en paramètre au constructeur (par exemple les vues)



Assets

- Similaire aux ressources, mais sans identifiant généré automatiquement
- répertoire `/assets`
- Accessible depuis la méthode `getAsset()` : `AssetManager` depuis la classe `Context`
- Plusieurs méthodes : `list (String filter)`, `open (String fileName)`, `openXmlRessource (String filename)`



Un exemple

```
String line; StringBuilder sb = new StringBuilder(); BufferedReader input = null;

try {
    input = new BufferedReader(new InputStreamReader(getAssets().open("hello.txt")));
    while ((line = input.readLine()) != null) {
        sb.append(line);
    }
} catch (IOException e) {
    if (input != null) {
        throw new RuntimeException(e);
    }
} finally {
    try {
        if (input != null) { input.close(); }
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

Nom du fichier asset

Attention aux ressources !

Bibliothèques (“Libraries”)

- Pour ajouter une bibliothèque (un JAR) au projet, rien de plus simple
- Il suffit de l'ajouter dans le répertoire **[project]/libs**
- Eclipse se charge de créer la référence dans le projet
- Le Dexer se chargera de convertir le bytecode JVM en bytecode Dalvik



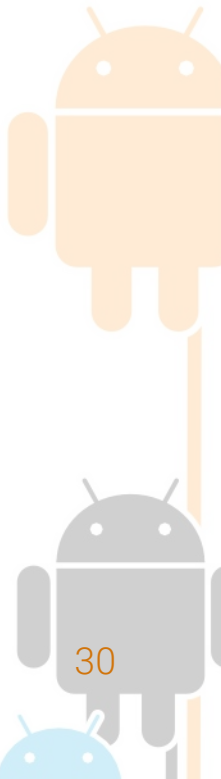
IN01 – Séance 02

Les composants



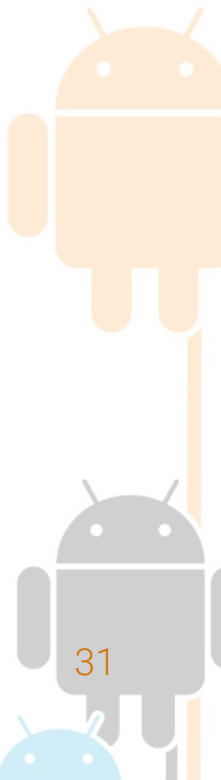
Les composants

- Une app peut être composée d'un ou plusieurs composants
 - ➔ Des activities
 - ➔ Des services
 - ➔ Content provider
 - ➔ Broadcast receiver



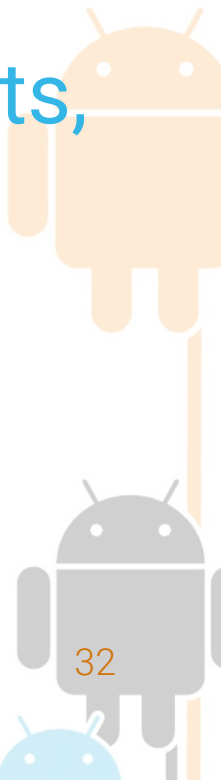
Activity et Service

- Une activity
 - ➔ Une application peut présenter une ou plusieurs Activities
 - ➔ Une “page” avec un HMI
 - ➔ Pris en charge avec le bouton back
 - ➔ Hérite de la classe `android.app.Activity`
- Un service et un composant en tâche de fond
 - ➔ Pas de HMI
 - ➔ Hérite de la classe `android.app.Service`



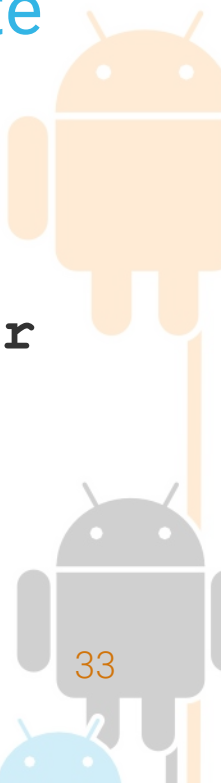
Content provider

- Fournisseur de contenu
- Donne accès à un ensemble de données structurées
- Un certain nombre de providers sont à disposition : Clock, Calendar, CallLog, Contacts, Documents, Media, Settings, Telephony
- Possibilité de créer son propre provider



Broadcast Receiver

- Abonne l'application à un évènement applicatif ou système
- Ex. : **ACTION_BOOT_COMPLETED** émet un évènement lorsque l'appareil est démarré
- Une méthode `onReceive` est appelée sur une classe dite “receiver”
- Processus asynchrone
- Hérite de la classe `android.content.BroadcastReceiver`
- Possibilité de créer son propre “emiter”



Résumé

- Activité == logique associée aux interfaces utilisateur
- Service == opération s'exécutant en arrière-plan
- Content Provider == fournisseur de contenu ! c'est-à-dire gestionnaire des données partageables et/ou persistantes
- Broadcast receiver == gestionnaire des messages système ou applicatifs
- Intent == message indiquant une intention à faire une certaine action



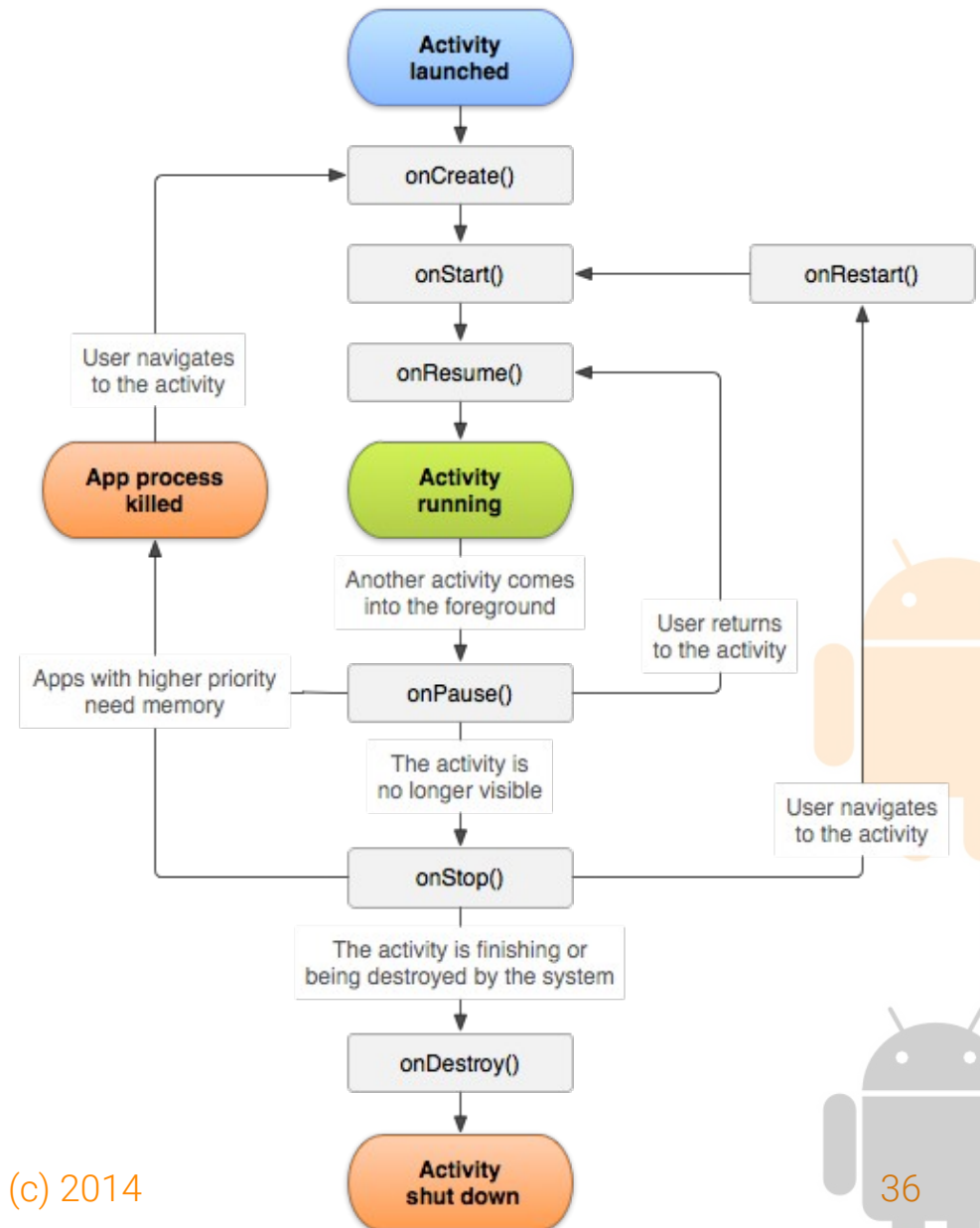
IN01 – Séance 02

Cycle de vie d'une application
et persistance



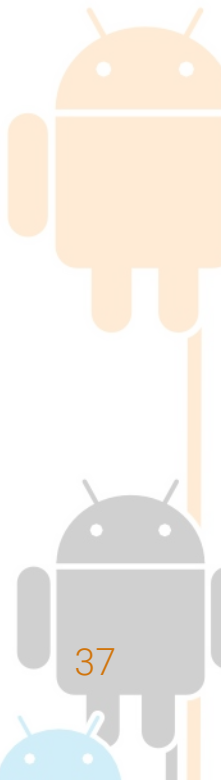
Cycle de vie

- Un certain nombre d'évènements avant et après le démarrage d'une application
- L'application est détruite **onDestroy** et recrée **onCreate** à chaque rotation de l'écran



Persistence

- Pour éviter la perte de données (durant la rotation, notamment), il faut les faire persister.
- Plusieurs options :
 - ➔ Utiliser un outil tout prêt, le Bundle
 - ➔ Faire persister en base de données (cf cours 04)
 - ➔ Sérialiser des objets dans le système de fichier (dans le cas de structures complexes)



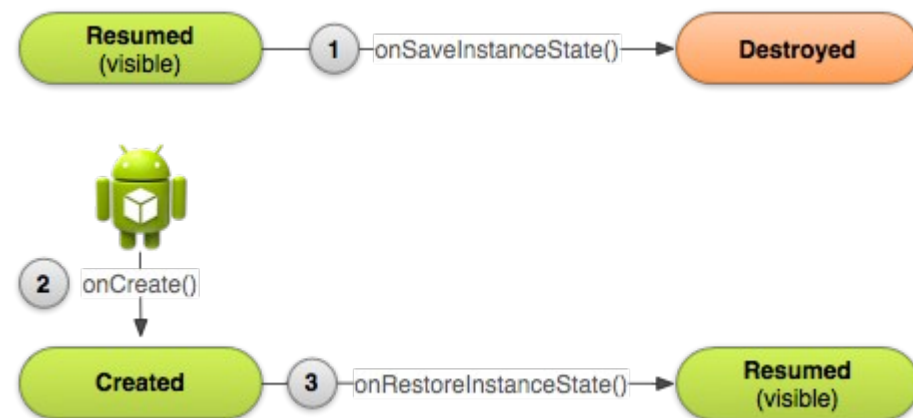
Bundle

- Bundle est un tableau associatif Clé/Valeur typé
- Des méthodes pour stocker les données : `putBoolean (String key, boolean value)`, `putInteger`, `putParcelable`, `putSerializable`
- Des méthodes pour récupérer les données : `getBoolean (String key)`, `getInteger`, `getParcelable`, `getSerializable`
- `Parcelable` décrit comment stocker un objet complexe dans un `Parcel`



Évènements

- Deux méthodes sont disponibles dans l'objet Activity pour réaliser la persistance.
- **onSaveInstanceState** et **onRestoreInstanceState**
- Il suffit de les surcharger dans notre Activity spécialisée



Un exemple

```
static final String STATE_SCORE = "playerScore";  
private int currentScore = 10;
```

@Override

```
protected void onSaveInstanceState (Bundle outState) {  
    outState.putInt(STATE_SCORE, currentScore);  
    super.onSaveInstanceState(outState);  
}
```

Sauve l'état

Appel de la méthode surchargée
Stocke les états au travers la hiérarchie de classes

@Override

```
protected void onRestoreInstanceState(Bundle savedInstanceState) {  
    super.onRestoreInstanceState(savedInstanceState);  
    currentScore = savedInstanceState.getInt(STATE_SCORE);  
}
```

Comme précédemment

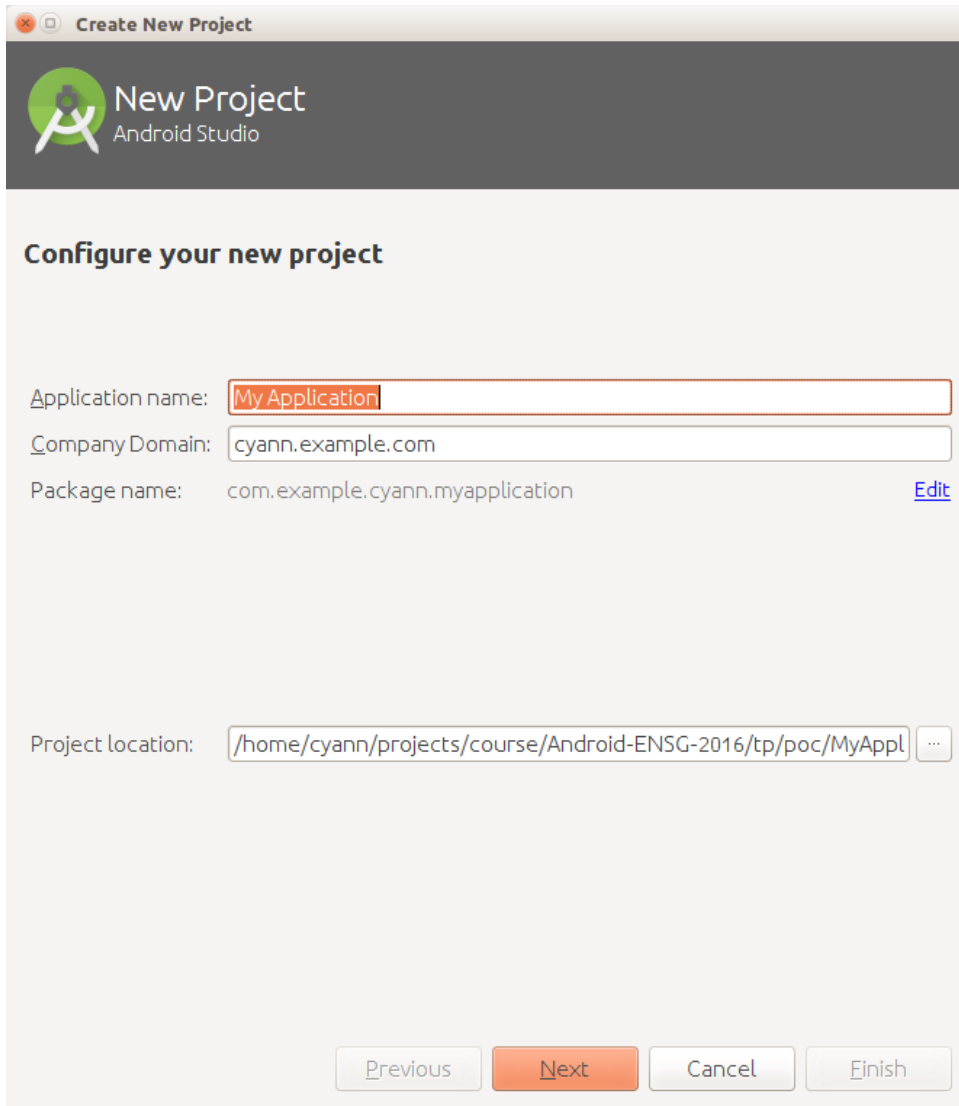
Récupère l'état

IN01 – Séance 02

Un premier projet



Création



Create New Project

New Project
Android Studio

Configure your new project

Application name:

Company Domain:

Package name: [Edit](#)

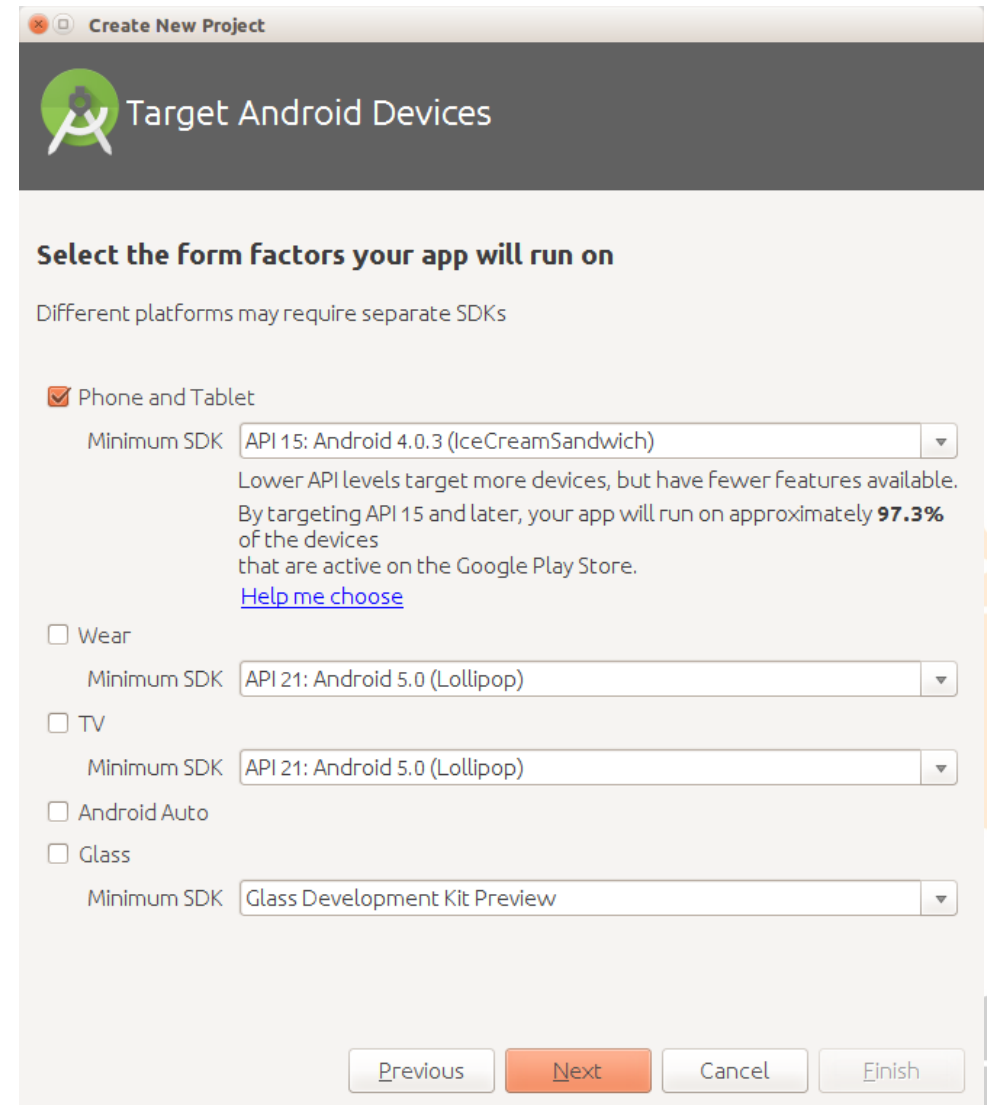
Project location: ...

Previous Next Cancel Finish

- Donner un nom (celui qui s'affiche sous l'icône)
- Un nom de projet
- Un chemin de package eu.ensg.*

Quelques options

- Choisir la version de l'API



The screenshot shows the 'Create New Project' dialog box in Android Studio. The title bar says 'Create New Project'. Below the title bar is a dark header with the Android logo and the text 'Target Android Devices'. The main content area is titled 'Select the form factors your app will run on' with a subtitle 'Different platforms may require separate SDKs'. There are five form factors listed with checkboxes: 'Phone and Tablet' (checked), 'Wear', 'TV', 'Android Auto', and 'Glass'. Each form factor has a 'Minimum SDK' dropdown menu. For 'Phone and Tablet', the dropdown shows 'API 15: Android 4.0.3 (IceCreamSandwich)'. Below this dropdown, there is a text block explaining that lower API levels target more devices but have fewer features, and that targeting API 15 and later covers approximately 97.3% of devices on the Google Play Store. A link 'Help me choose' is provided. The other form factors have their respective SDK versions: 'Wear' (API 21: Android 5.0 (Lollipop)), 'TV' (API 21: Android 5.0 (Lollipop)), 'Android Auto' (API 21: Android 5.0 (Lollipop)), and 'Glass' (Glass Development Kit Preview). At the bottom of the dialog are four buttons: 'Previous', 'Next' (highlighted in orange), 'Cancel', and 'Finish'.

Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

☒ Phone and Tablet

Minimum SDK: API 15: Android 4.0.3 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available. By targeting API 15 and later, your app will run on approximately **97.3%** of the devices that are active on the Google Play Store. [Help me choose](#)

☐ Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

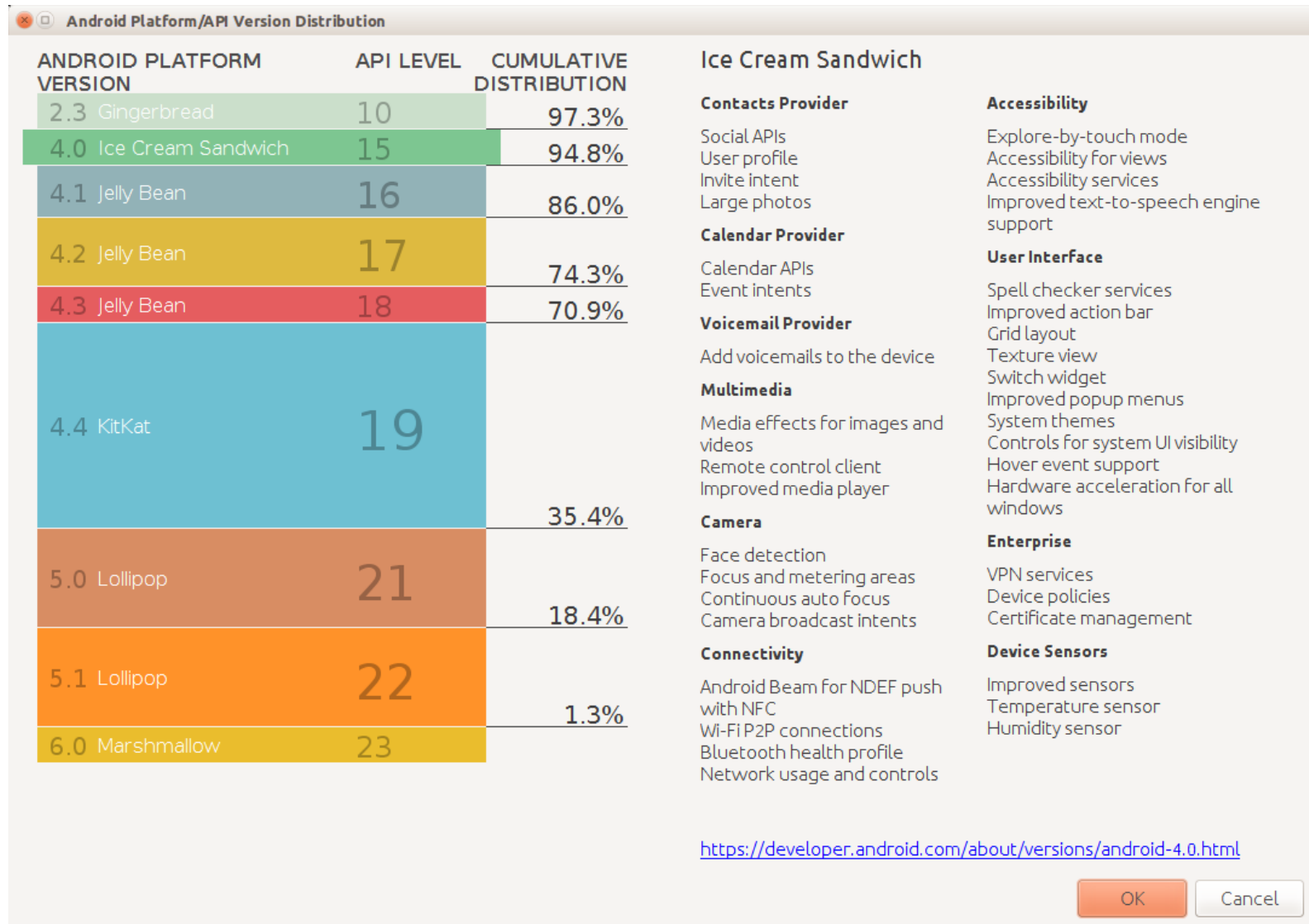
☐ Android Auto

☐ Glass

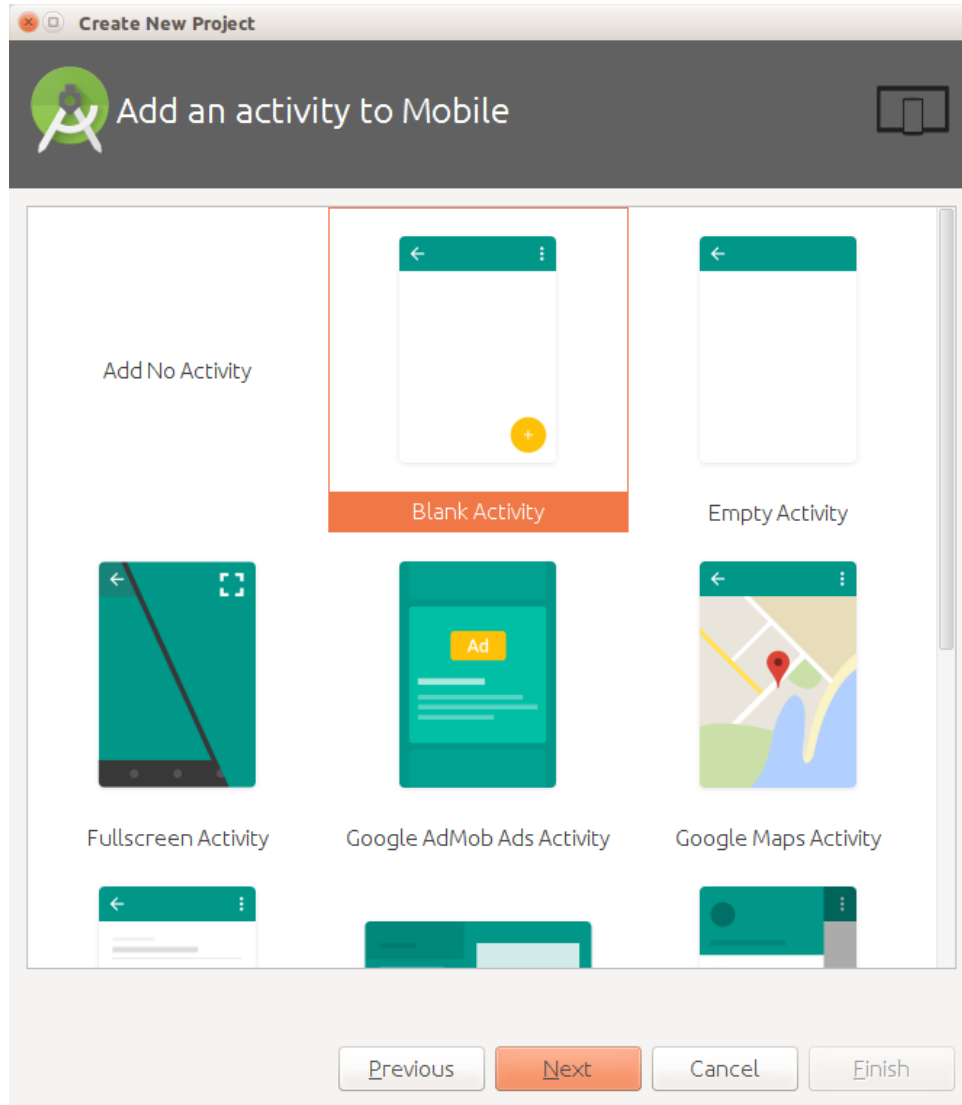
Minimum SDK: Glass Development Kit Preview

Previous Next Cancel Finish

Min API

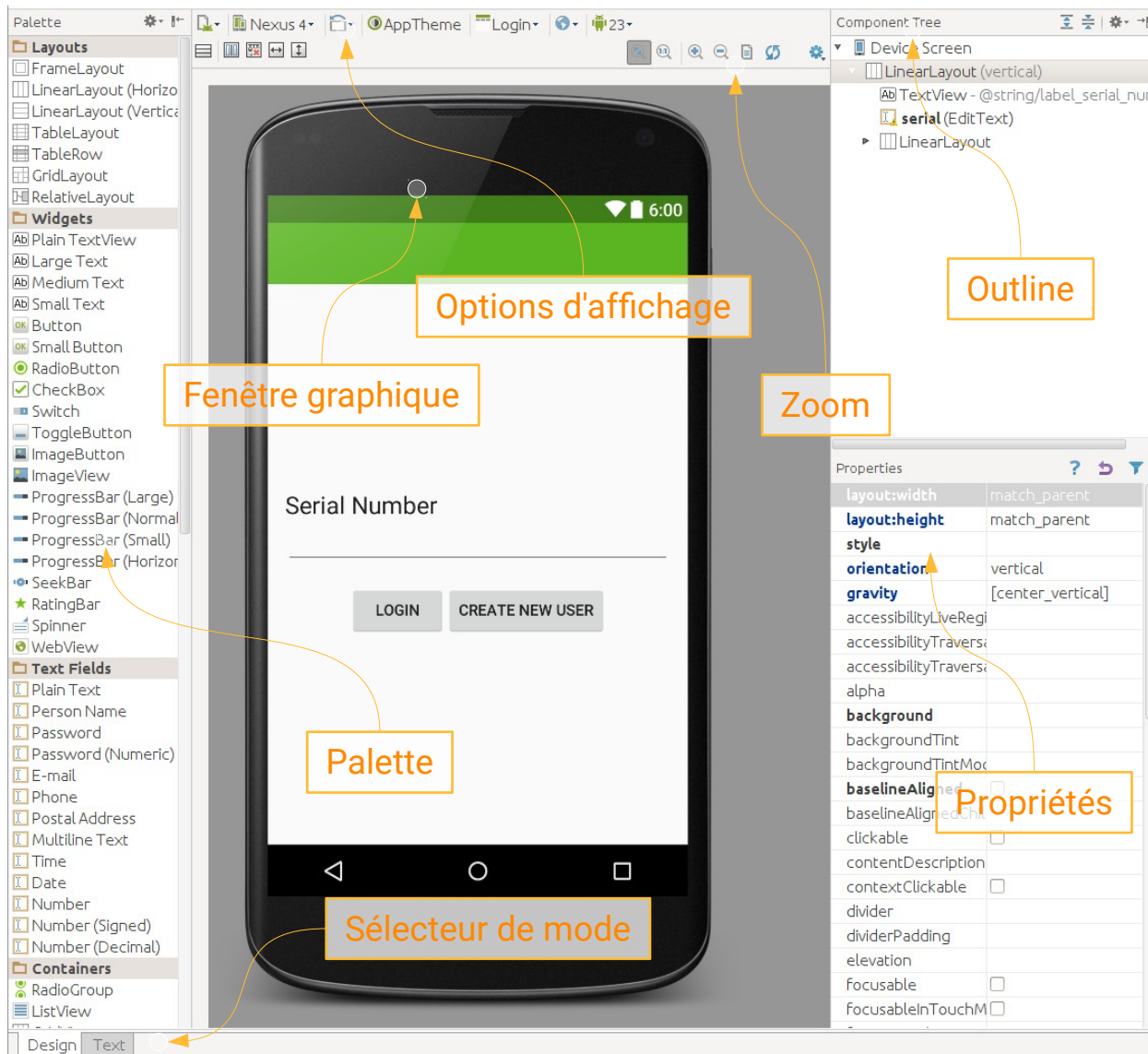


L'activity



- Le type d'activity de départ
- Conseil : `EmptyActivity`

WYSIWYG



■ Préférer le mode XML

- compréhension
- copier/coller
- + efficace à l'usage
- code maîtrisé (+ propre)

Manifest

- Toute la configuration du projet
- Changer l'Icône

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="eu.ensg.forester" >

    <!--
        The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
        Google Maps Android API v2, but you must specify either coarse or fine
        location permissions for the 'MyLocation' functionality.
    -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Forester"
        android:supportRtl="true"
        android:theme="@style/AppTheme" >
        <activity android:name=".LoginActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".CreateUserActivity" />
    <!--
        The API key for Google Maps-based APIs is defined as a string resource.
        (See the file "res/values/google_maps_api.xml").
        Note that the API key is linked to the encryption key used to sign the APK.
        You need a different API key for each encryption key, including the release key that i
        sign the APK for publishing.
        You can define the keys for the debug and release targets in src/debug/ and src/releas
    -->
    <meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="@string/google_maps_key_home" />

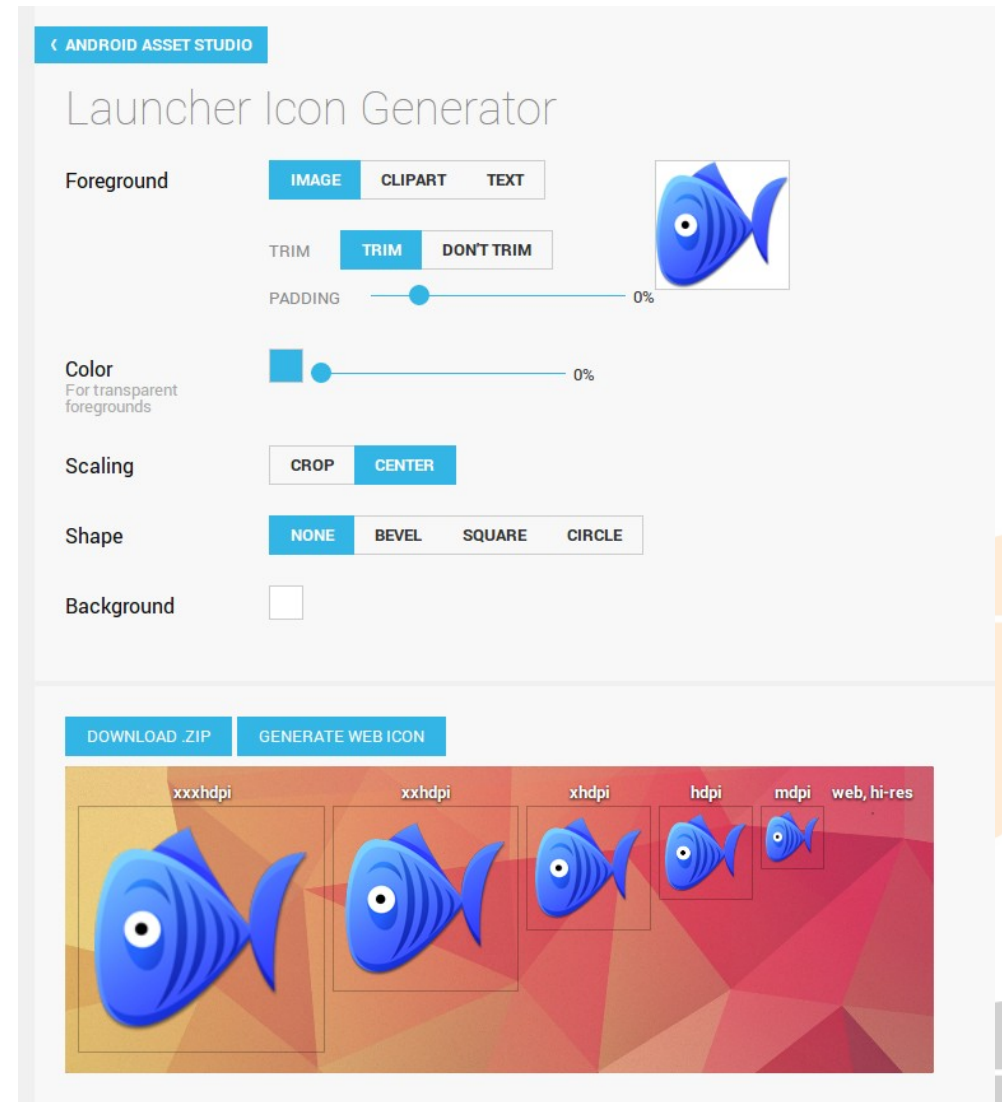
    <activity
        android:name=".MapsActivity"
        android:label="Map" />

    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@string/google_play_services_version" />
    </application>
</manifest>
```

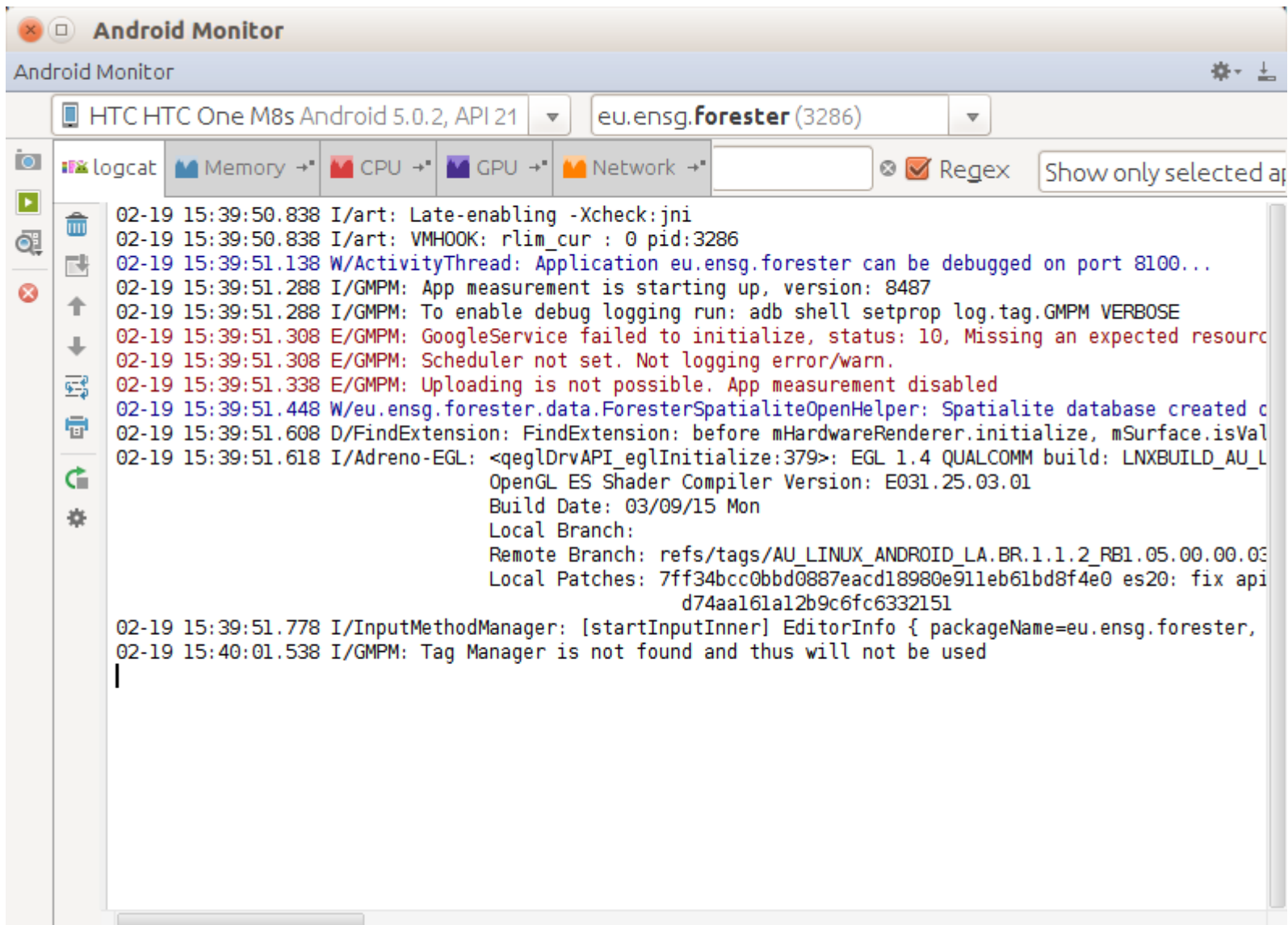


Création d'icône

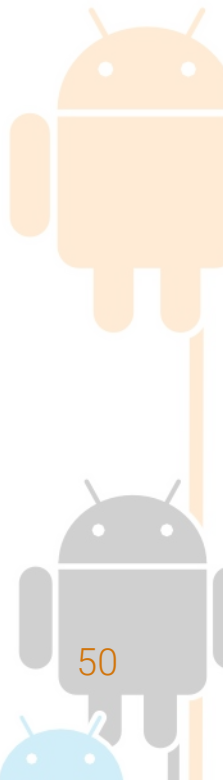
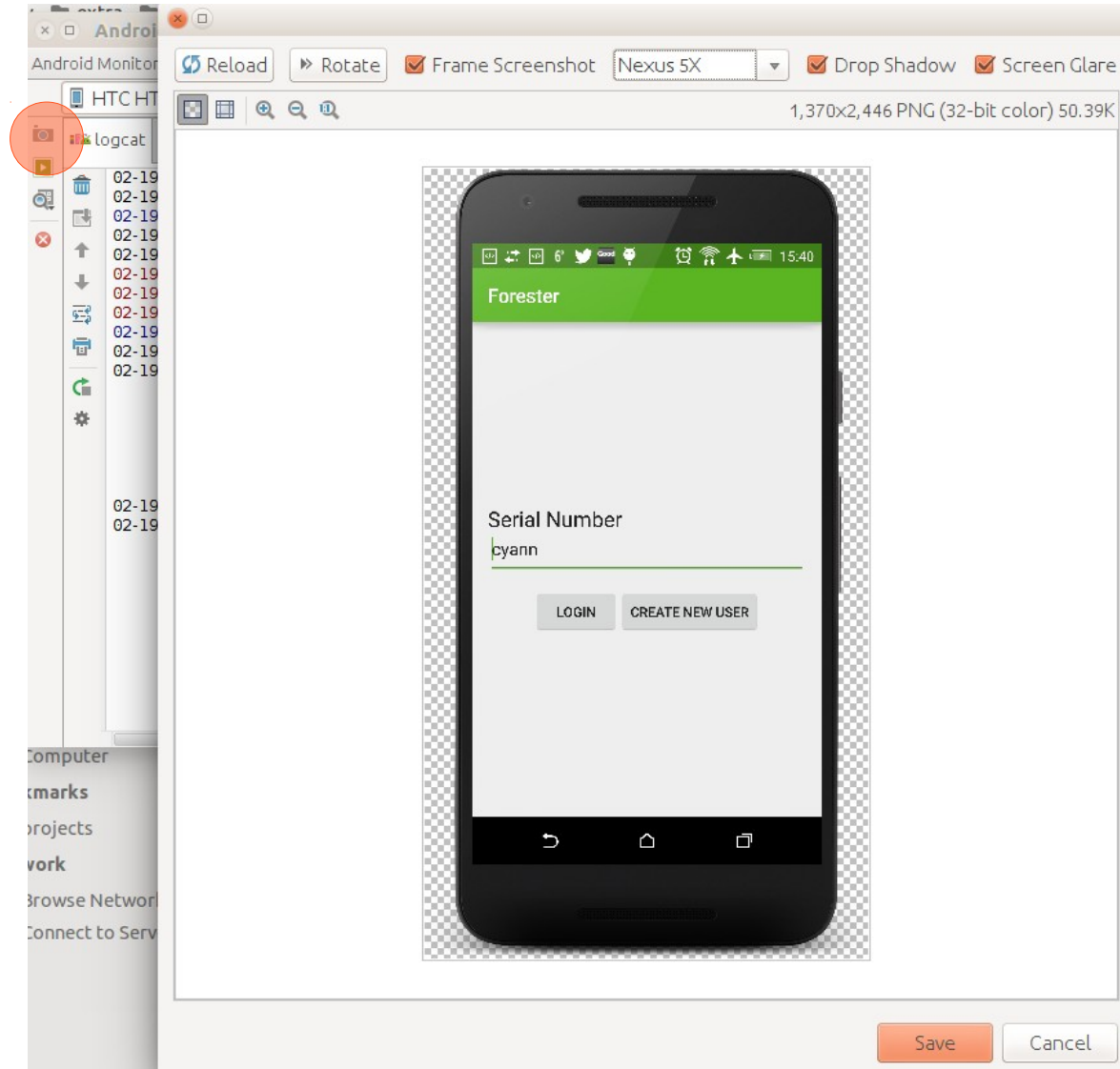
- Google met à disposition des outils HTML
- Disponible ici : <http://romannurik.github.io/AndroidAssetStudio/>



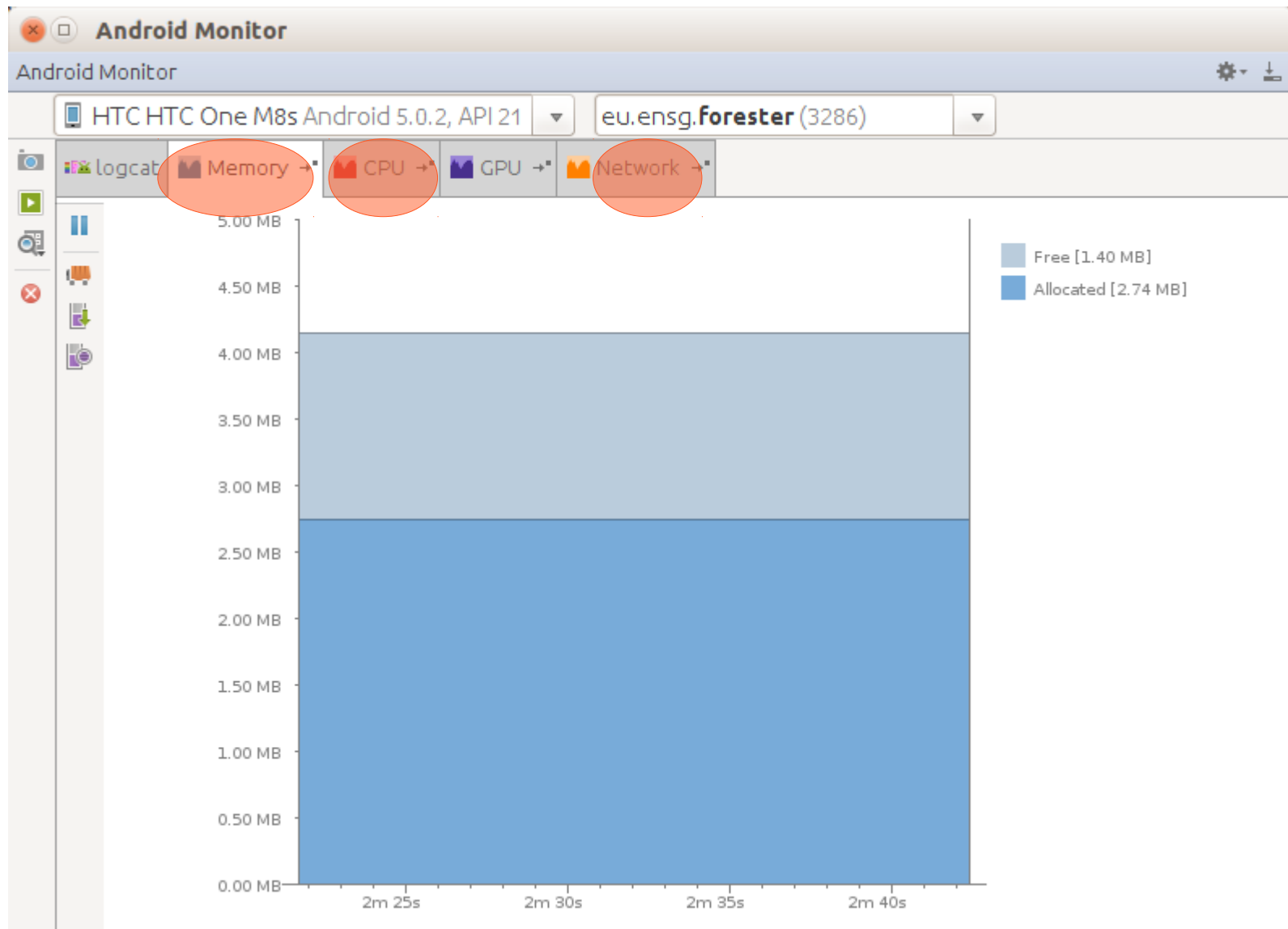
Android Monitor - Logcat



Screenshot



Gestion des ressources



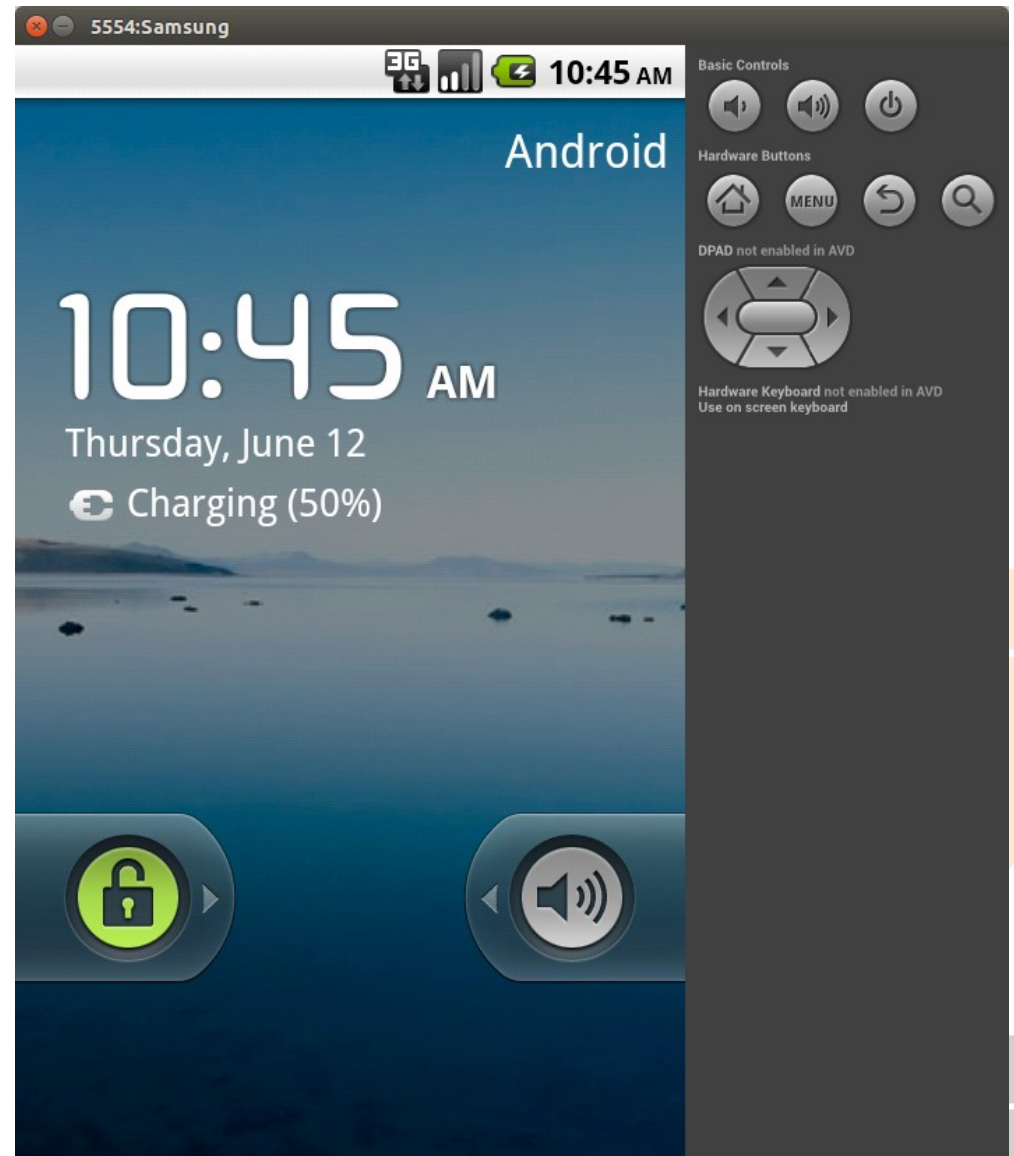
IN01 – Séance 02

Exécution et adb



Exécution

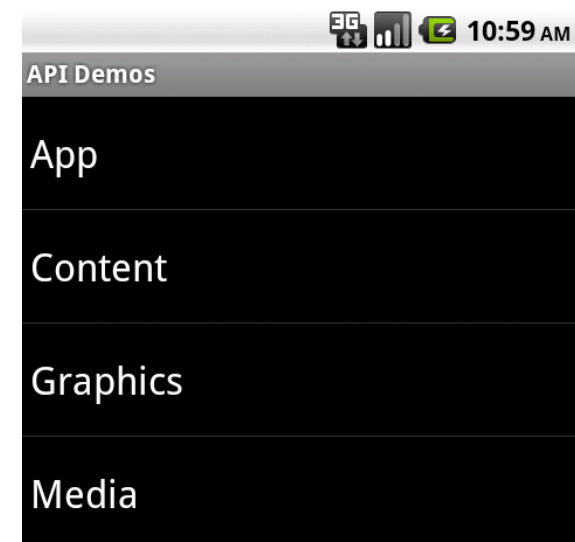
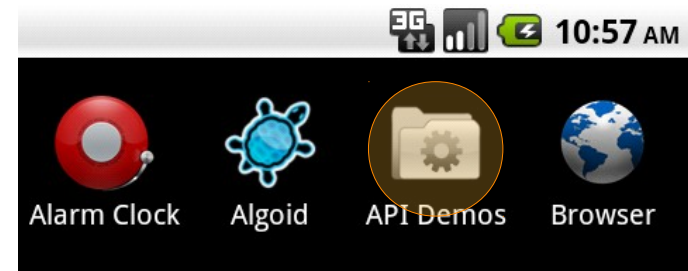
- Deux possibilités
 - ➔ l'émulateur (plus lent, mais plus flexible, capable de simuler diverses résolutions)
 - ➔ sur le device (les vraies performances)



Des exemples dans l'émulateur

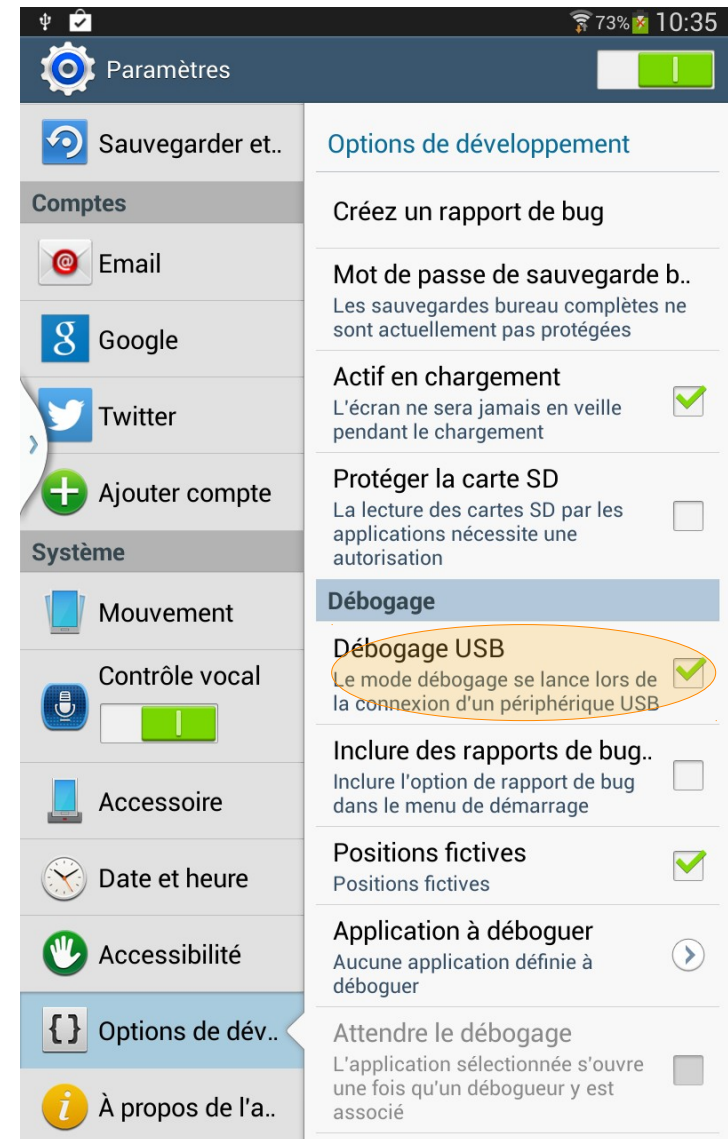
- Des exemples sont disponibles dans l'émulateur
- Il est possible de les exécuter depuis les menus
- Le code est accessible à partir de :

`%SDK%\sample\android-XXX\ApiDemos`



Appareil - Activer le débogage

- Pour développer depuis l'appareil
- Dans le menu Options de développement, il faut activer le menu Débogage USB
- Et brancher le câble USB entre le PC et l'appareil



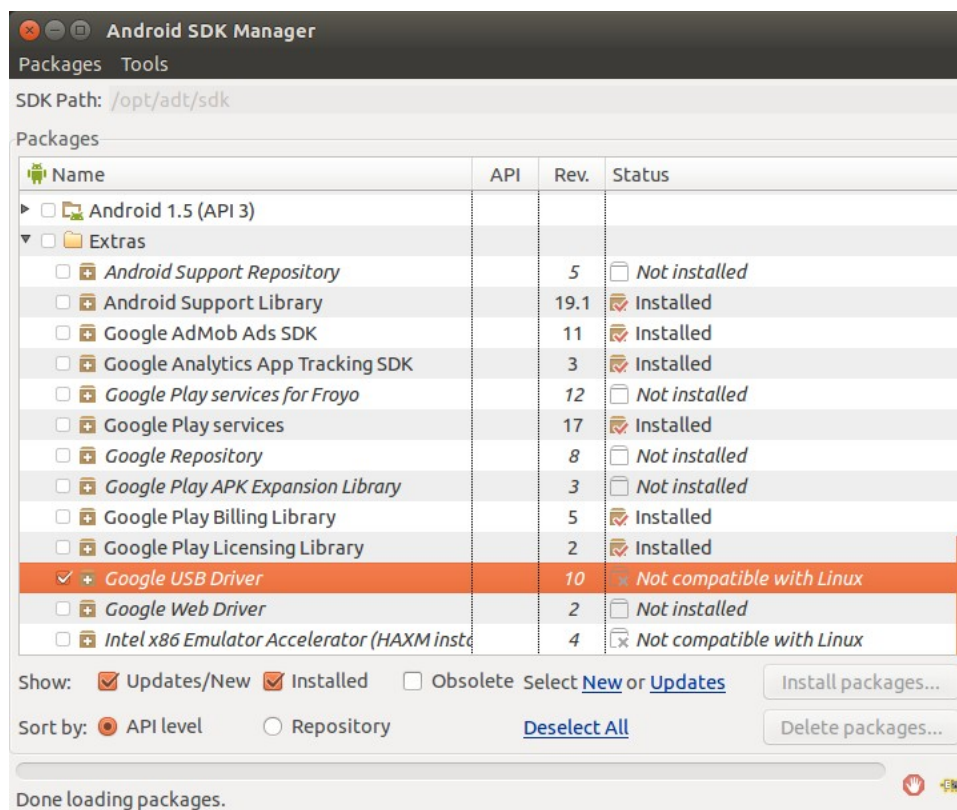
Astuce Samsung

- Sur certains appareils Samsung, le menu “Developer” est caché
- Pour le réactiver, il faut aller dans Settings/About Device et toucher 7 fois (!!!!) le menu Build number
- Sur les appareils en français, Paramètres/À propos de l'appareil, menu Numéro de version



Sous Windows

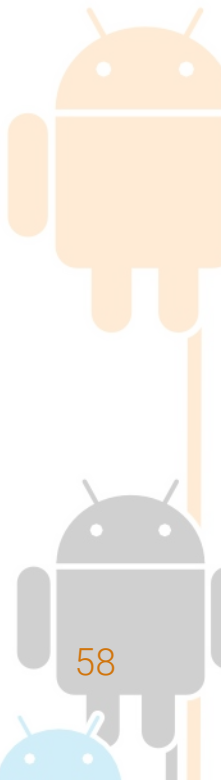
- Les drivers ne sont pas présents par défaut. Il faut aussi les télécharger



adb devices

- Android debug bridge
- Lancer une fenêtre DOS ou un shell
- Tester si l'appareil est connecté **adb devices**

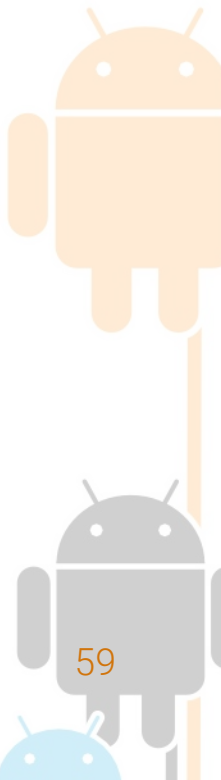
```
cyann@cyann-SG33: ~  
cyann@cyann-SG33:~$ adb devices  
List of devices attached  
320449e154b0c051    device  
cyann@cyann-SG33:~$
```



adb start, kill

- Très utile lors d'un dysfonctionnement
- `adb kill-server` et `adb start-server`

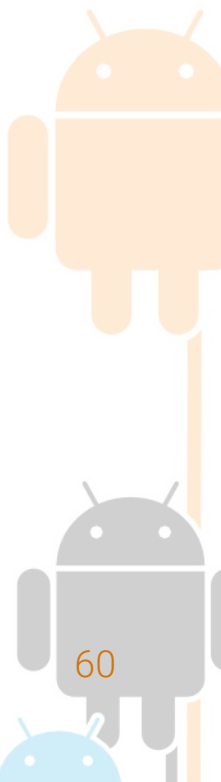
```
cyann@cyann-SG33: ~  
cyann@cyann-SG33:~$ adb kill-server  
cyann@cyann-SG33:~$ adb start-server  
* daemon not running. starting it now on port 5037 *  
* daemon started successfully *  
cyann@cyann-SG33:~$
```



adb shell

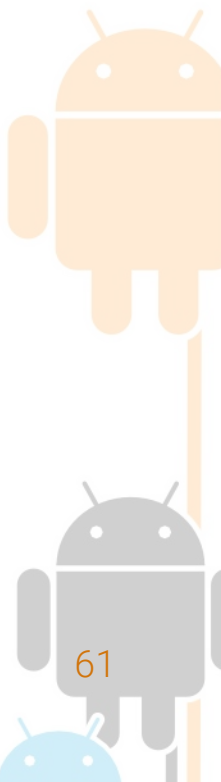
- Un remote shell sur l'appareil depuis le PC
- `adb shell`
- Le reste, c'est du Linux (`ls`, `mkdir`, `chmod`, `cat`, `exit`)

```
cyann@cyann-SG33: ~  
cyann@cyann-SG33:~$ adb shell  
shell@android:/ $ ll  
drwxr-xr-x root      root      2014-05-24 07:21 acct  
drwxrwx--- system    cache     2014-06-12 13:34 cache  
dr-x----- root      root      2014-05-24 07:21 config  
lrwxrwxrwx root      root      2014-05-24 07:21 d -> /sys  
/kernel/debug  
drwxrwx--x system    system    2014-05-24 07:21 data  
-rw-r--r-- root      root      132 1970-01-01 01:00 default.p  
rop  
drwxr-xr-x root      root      2014-05-24 07:21 dev  
drwxrwx--x radio      system    2013-01-01 01:02 efs  
lrwxrwxrwx root      root      2014-05-24 07:21 etc -> /s  
ystem/etc  
lrwxrwxrwx root      root      2014-05-24 07:21 factory -
```



adb transfert

- Installer une APK à distance
 - ➔ `adb install c:\file.apk`
- Envoyer un fichier sur le téléphone
 - ➔ `adb push c:\file.txt /sdcard`
- Télécharger un fichier depuis le téléphone
 - ➔ `adb pull /sdcard/file.txt c:\`



IN01 – Séance 02

Logcat



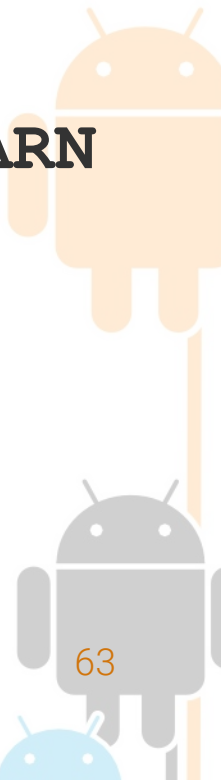
android.util.Log

- Le logger interne à la plateforme
- Accessible par programmation : `android.util.Log`
- Une méthode statique générale :

```
println(Int priority, String tag, String  
msg)
```

- Des priorités `DEBUG`, `ERROR`, `INFO`, `VERBOSE`, `WARN`
- Des méthodes statiques utilitaires

```
d(String tag, String msg), e, i, v ,w
```



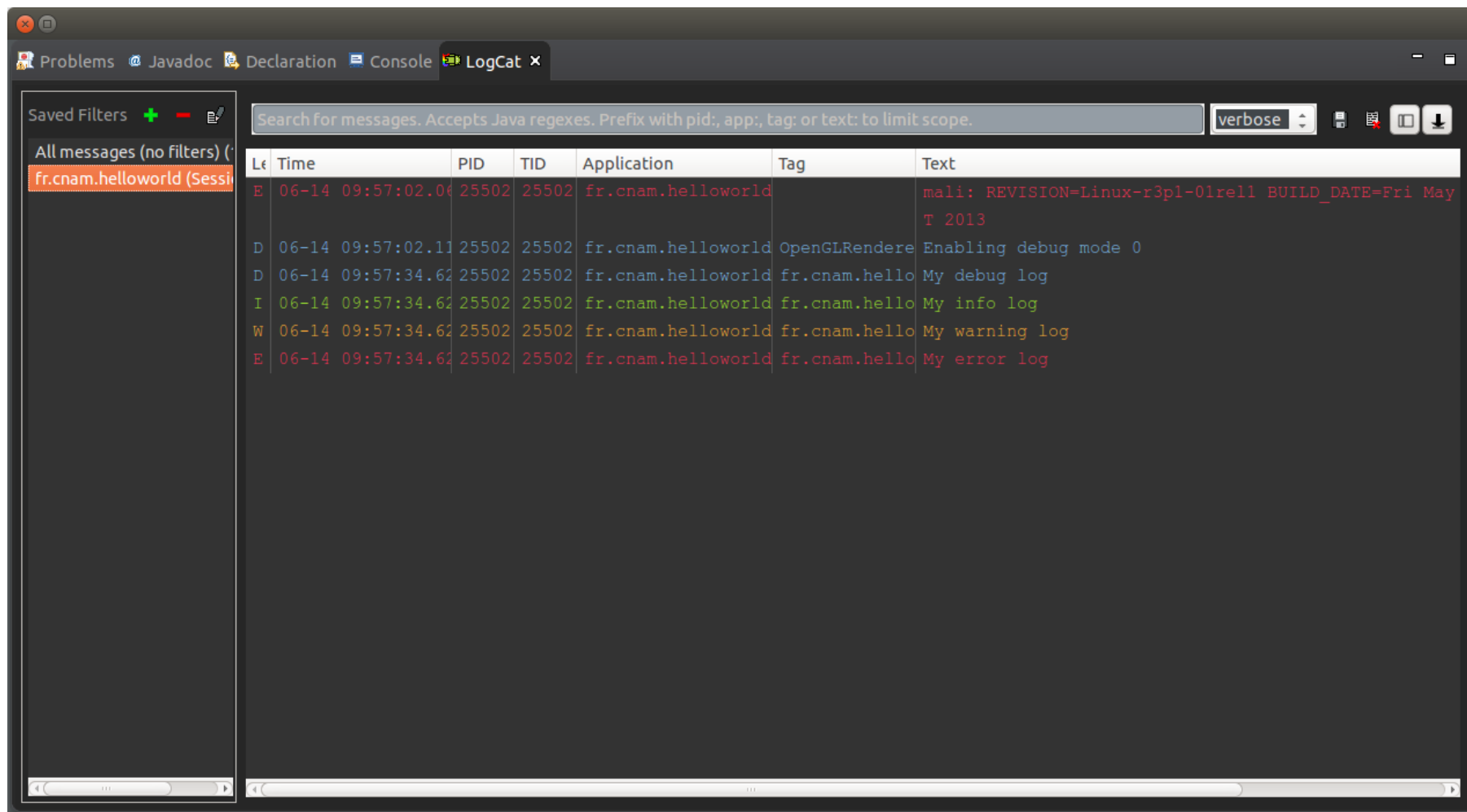
Exemple

```
Log.d(MainActivity.class.getName(), "My debug log");  
Log.i(MainActivity.class.getName(), "My info log");  
Log.w(MainActivity.class.getName(), "My warning log");  
Log.e(MainActivity.class.getName(), "My error log");
```

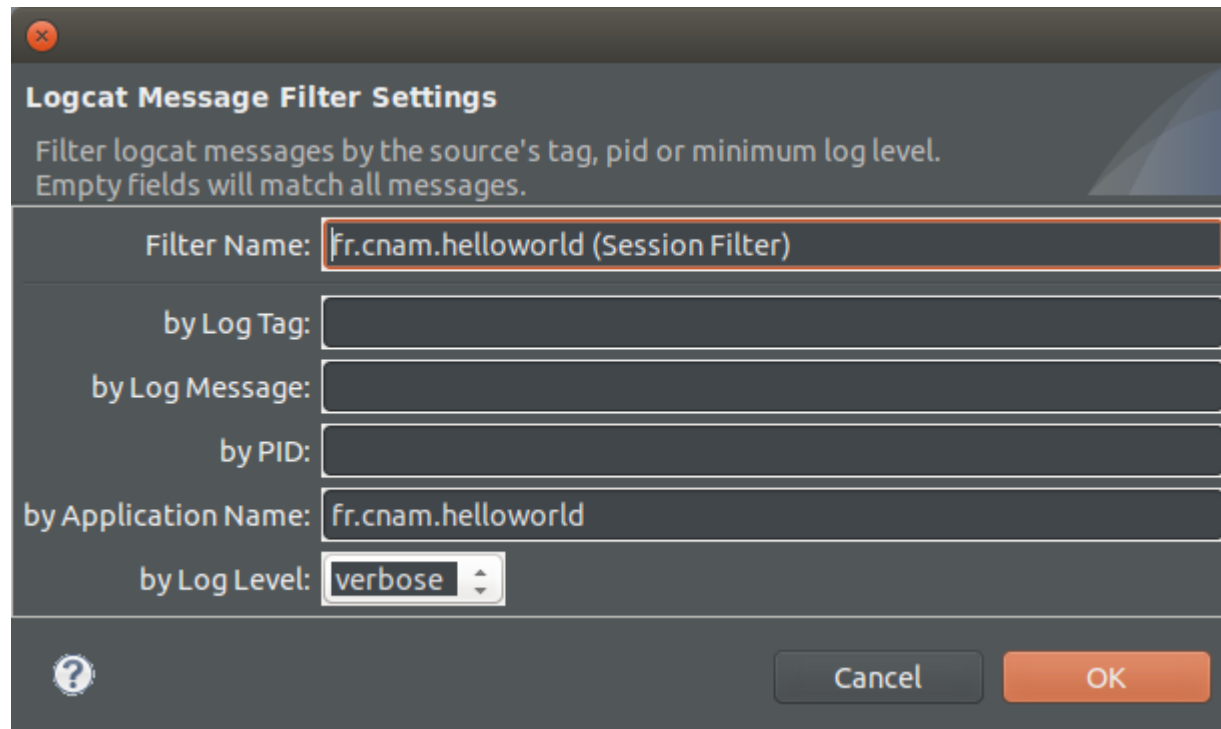
- Utile pour tracer le programme et comprendre ce qu'il se passe, sans utiliser le débogueur

```
D 06-14 09:57:34.62 25502 25502 fr.cnam.helloworld fr.cnam.hello My debug log  
I 06-14 09:57:34.62 25502 25502 fr.cnam.helloworld fr.cnam.hello My info log  
W 06-14 09:57:34.62 25502 25502 fr.cnam.helloworld fr.cnam.hello My warning log  
E 06-14 09:57:34.62 25502 25502 fr.cnam.helloworld fr.cnam.hello My error log
```


DDMS LogCat



Filtres



- Eclipse filtre automatiquement sur le nom de l'application

adb logcat

- Accès au Logcat en ligne de commande
- **adb logcat**

```
cyann@cyann-SG33: ~  
D/accuweather(16817): [Accuweather J]>>> SM:3075 [0:0] onSensorChanged offsetX = -0.0035051114, offsetY = -  
3.0645783E-4  
D/accuweather(16817): [Accuweather J]>>> WR:540 [0:0] onGLDraw : BG texture = 7  
D/STATUSBAR-NetworkController( 2407): refreshSignalCluster: data=0 bt=false  
D/accuweather(16817): [Accuweather J]>>> WR:540 [0:0] onGLDraw : BG texture = 7  
D/STATUSBAR-NetworkController( 2407): refreshSignalCluster: data=0 bt=false  
D/accuweather(16817): [Accuweather J]>>> SM:3075 [0:0] onSensorChanged offsetX = -0.0035625722, offsetY = -  
3.639187E-4  
D/accuweather(16817): [Accuweather J]>>> WR:540 [0:0] onGLDraw : BG texture = 7  
D/STATUSBAR-NetworkController( 2407): refreshSignalCluster: data=0 bt=false  
D/accuweather(16817): [Accuweather J]>>> WR:540 [0:0] onGLDraw : BG texture = 7  
D/accuweather(16817): [Accuweather J]>>> WR:540 [0:0] onGLDraw : BG texture = 7  
D/accuweather(16817): [Accuweather J]>>> SM:3075 [0:0] onSensorChanged offsetX = -0.0033901895, offsetY = -  
4.2137952E-4  
D/accuweather(16817): [Accuweather J]>>> WR:540 [0:0] onGLDraw : BG texture = 7
```

Fin

- Merci de votre attention
- Des questions ?

