Technologies mobiles

Olivier Levitt

Source et pdf de ce cours :
`https://bitbucket.org/olevitt/technologies-mobiles`

Sommaire

Sommaire

Planning

- 30 janvier : 3h de cours, 3h de TP
- 6 février : 3h de cours
- 13 février : 6h de TP
- Validation des sujets de projet avant le 20 février
- 20 février : 6h de TP dédiées au projet
- ? mars : Soutenance du projet

Evaluation

- Projet : création d'une application
- Groupe de 2
- Sujet "libre"
- 6h de TP dédiées au projet + **travail personnel**
- Soutenance / Présentation de l'application

Evaluation, exemples de sujets

- PamplemousseViewer v2
- Gestion d'une bibliothèque
- Quiz
- Tape-taupes
- Serveur SMS
- Achievements

Sommaire

Contexte et objectifs

- Smartphones, tablettes et assimilés (TV, montre, autoradio, consoles de jeu . . .)
- Développement d'application, pas de dev de la plateforme
- 1ère partie : le développement mobile en général
- 2ème partie : application sous android

Sommaire

Des appareils suréquipés

- Téléphonie (SMS, MMS, appels)
- Internet (GPRS, EDGE, 3G, 4G, WIFI)
- Réseaux locaux (Bluetooth, réseaux adhoc, NFC)
- Capteurs (Luminosité, proximité)
- Localisation (GPS, triangulation, SSID wifi)
- Notifications (Vibreur, haut-parleurs, LED)
- Photo / vidéo
- Stockage de données (Mémoire flash, SD externe, SQLite)
- Interactions (Ecran tactile, gestures, boutons physique)
- Et encore d'autres ...

Et des API pour utiliser tout ça !

Des contraintes techniques importantes

- Processeur
- Mémoire RAM
- Stockage de données
- Gestion de la batterie
- Stabilité et débit de la connexion internet
- Cycle de vie de l'application
- Taille d'écran
- Inputs atypiques (clavier virtuel, gestures, peu de boutons . . .)

Contraintes à garder en tête en permanence.

La fragmentation

Une application publiée sur le google playstore cible plus de 2400 appareils différents !

- “Write once, run everywhere” ?
- Comment tester / débugger pour tous ces appareils ?
- Eviter de géner l’utilisateur (versions HD, appareils non compatibles)
- S’adapter quand une fonctionnalité n’est pas disponible

La fragmentation, taille d'écran

Comment gérer toutes les tailles d'écran ?

- Montres connectées : de 1 à 2 pouces
- Smartphones lowcost : 3 pouces (Galaxy pocket, galaxy Y)
- Smartphones high-end : 4 à 5 pouces (IPhone 5, HTC 8X, nexus 4)
- Phablets : 5 à 6 pouces (Galaxy note, HTC butterfly)
- Tablettes : 7 pouces (Nexus 7, IPad mini), 8 pouces (Archos 80g9), 10 pouces (Nexus 10, IPad)

De nombreuses autres sources de fragmentation

- Versions de l'OS
- Résolutions d'écran
- Elements hardware présents
- Puissance
- Modifications constructeur / "rom custom"
- ...

Des Ecosystèmes forts

- Obligation d'utiliser le SDK fourni
- Suivre les guidelines
- Restrictions liées à la plateforme
- Utilisation des services de la plateforme
- Processus de déploiement des applications
- Règles des "store" (validation, monétisation . . .)

Sommaire

iOS

- Soutenu par Apple
- Présenté le 9 janvier 2007
- Dédié aux produits apple (iPhone, iPad, iPod)
- 400 millions d'appareils (Septembre 2012)
- Programmation en objective-C, sur mac OS X uniquement
- Appstore : validation + 100$ / an

Android

- Soutenu par Google
- 1.0 en septembre 2008, 1.5 en avril 2009
- Plus de 2400 appareils officiellement supportés, plus de 50 constructeurs
- 480 millions d'appareils activés (Septembre 2012)
- Programmation en JAVA, sur windows / OS X / linux
- Open-source
- Google playstore : pas de validation + 25$

Windows phone 8

- Soutenu par Microsoft
- Présentation au public le 29 octobre 2012
- Successeur de windows phone 7 (logique)
- Plusieurs constructeurs dont Nokia, HTC et Samsung
- Programmation en C# sur windows
- Windows marketplace : validation + 100$ / an

Blackberry 10

- Soutenu par RIM (Research in motion)
- Présentation au public le 30 janvier 2012 (!)
- Appareils produits par RIM
- C / C++, HTML5, Adobe AIR, Portage android
- Blackberry appworld : validation + gratuit

Ubuntu for phones

- Soutenu par Canonical
- Teaser le 2 janvier 2012, testable sur galaxy nexus fin février
- Premiers ubuntu phones promis pour début 2014
- Facilement utilisable sur les téléphones android ?
- HTML5, C/C++ + QML
- Open-source
- Peu d'infos sur le store

Firefox OS

- Soutenu par Mozilla
- Premiers téléphones présentés le 22 janvier (geeksphone), disponibles en février ?
- Simulateur sous forme d'addon firefox
- HTML5
- Open-source
- Peu d'infos sur le store

Sommaire

Les marque-pages

- www.frandroid.com (actu FR)
- www.androidpolice.com (actu EN)
- www.androidcentral.com (actu EN)
- www.d.android.com (la bible EN)
- www.stackoverflow (Q/A EN)
- #android et #android-dev sur freenode (chat irc EN)
- www.breizhjug.org et www.paug.fr (communautés FR)
- www.google.fr (réservoir à tutoriels)

Avant de commencer, la checklist

Obligatoire :

- Des (bonnes) bases de programmation en JAVA
- Un ordinateur (Windows, Linux, Mac OS X)

Conseillé :

- Un appareil android (l'émulateur est . . . moyen)
- Parler anglais
- Suivre l'actualité

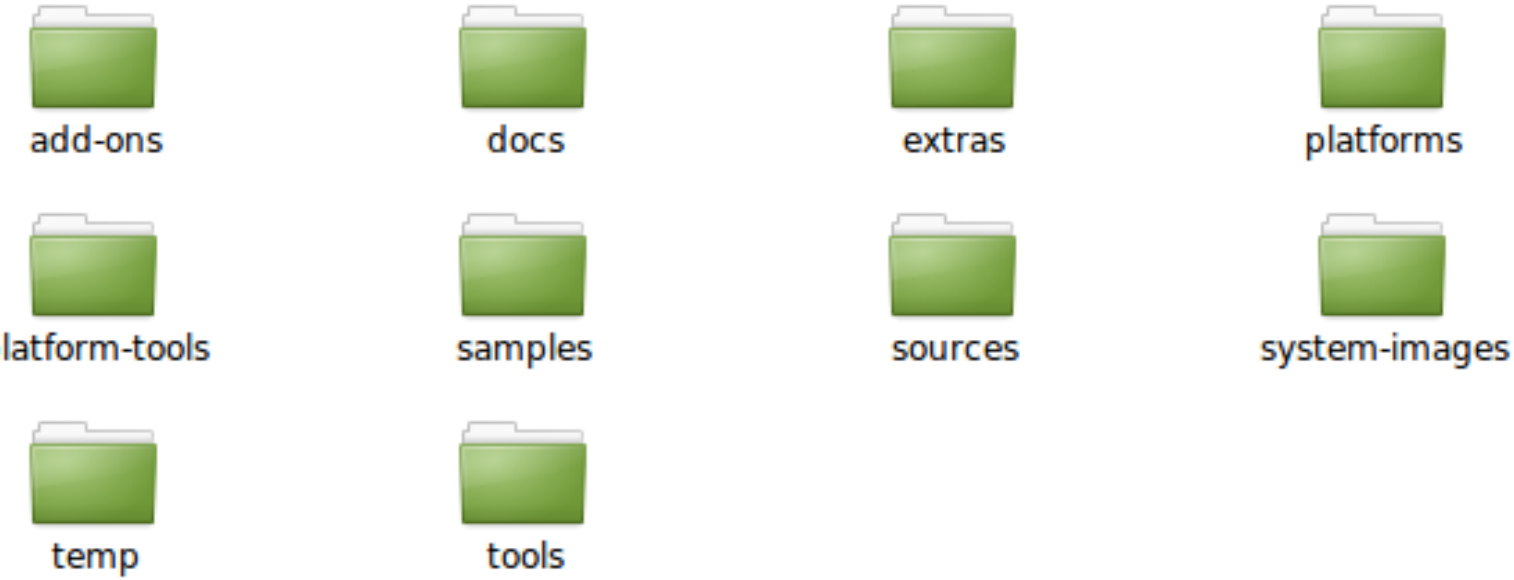
Les niveaux d'API

Version	Nom	API level	Distribution	Cumulé
1.5	Cupcake	3	0%	0%
1.6	Donut	4	0.2%	0.2%
2.1	Eclair	7	2.4%	2.6%
2.2	Froyo	8	9%	11.6%
2.3	Gingerbread	9/10	47.6%	59.2%
3.X	Honeycomb	12/13	1.5%	60.7%
4.0.X	Ice cream sandwich	15	29.1%	89.8%
4.1	Jelly bean	16	9%	98.8%
4.2	Jelly bean	17	1.2%	100%

TABLE: Répartition des versions pour les accès au google play sur la dernière quinzaine de 2012

Présentation du SDK android

Téléchargement gratuit : www.d.android.com/sdk

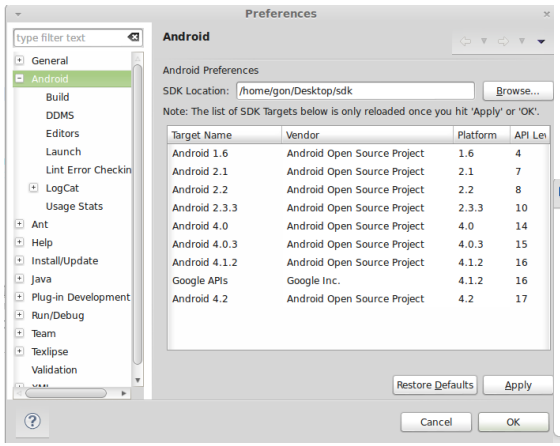


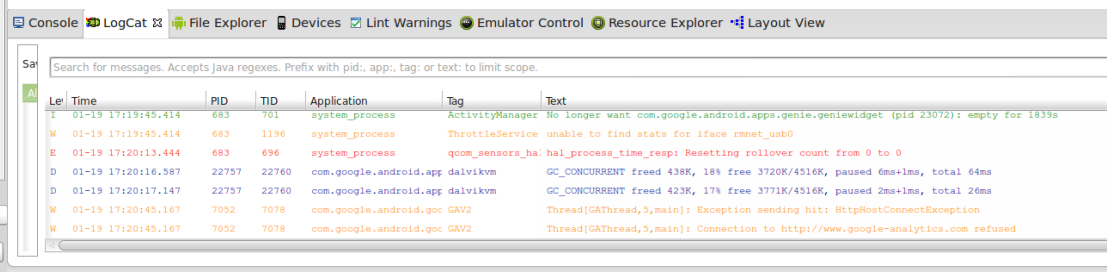
Présentation du SDK android

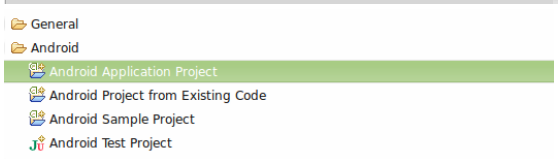
- add-ons : Google APIs
- docs : Copie de la documentation disponible sur d.android.com
- extras : Lib de compatibilité, lib pour les achats in-app . . .
- platform-tools : Binaires de communication avec les appareils android (adb, fastboot . . .)
- platforms : 1 dossier par niveau d'API téléchargé
- samples : Exemples de projets
- sources : Sources de chaque niveau d'API
- system-images : Images pour l'émulateur
- temp
- tools : Outils pour le dev (ddms, apkbuilder, lint . . .)

Plugin android pour eclipse : ADT

Installation comme un plugin eclipse classique
`https://dl-ssl.google.com/android/eclipse/`
ADT fait le lien entre eclipse et le SDK android

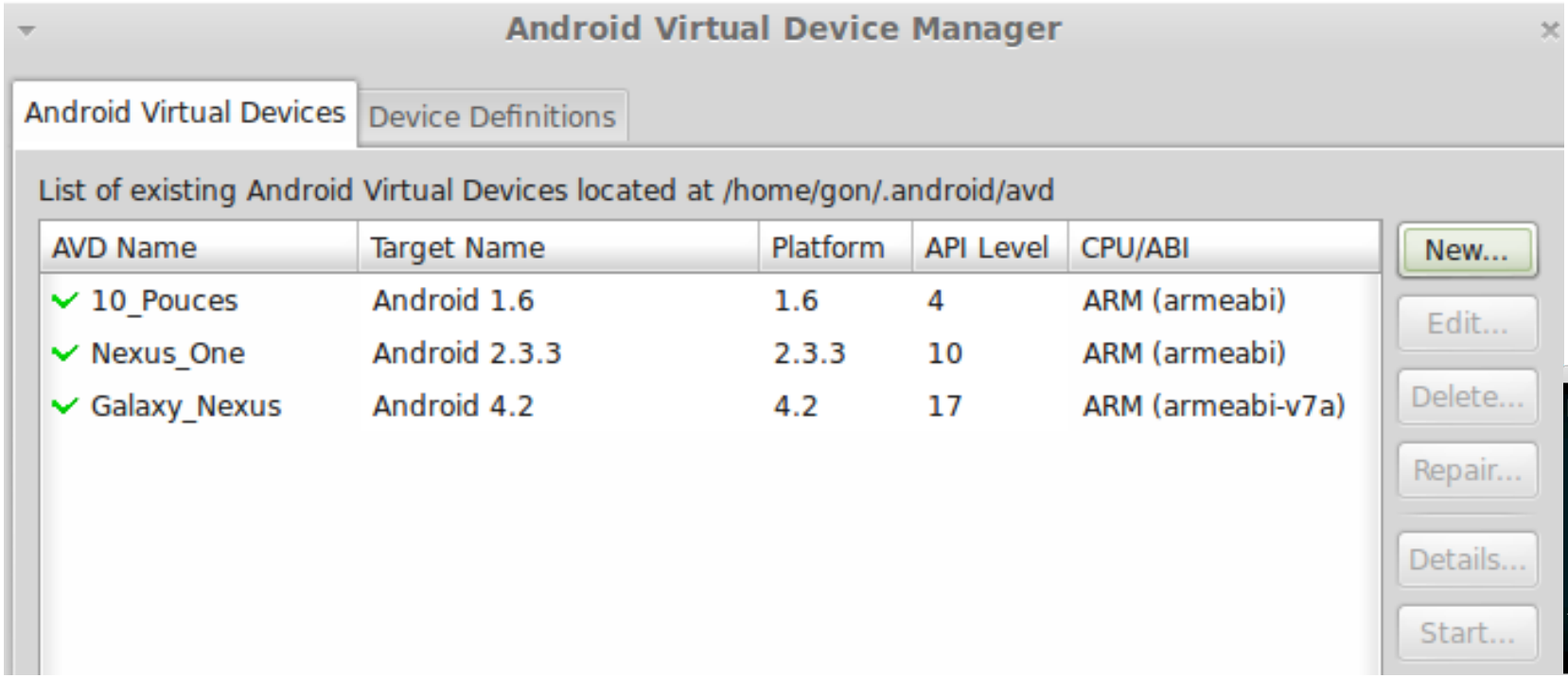






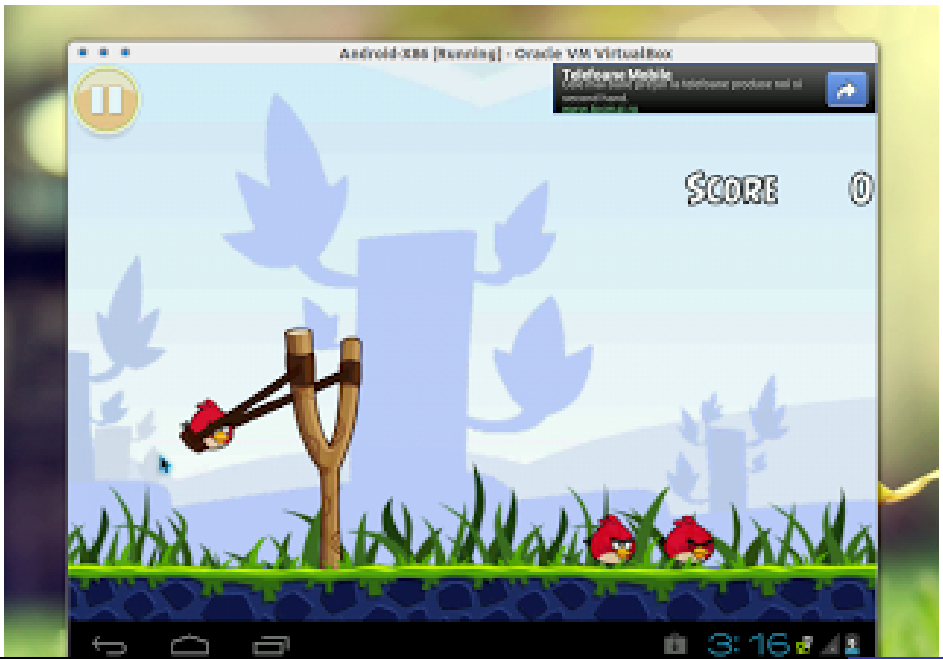
L'émulateur

- Utile pour tester certaines configurations
- ((très) très) lent
- Utiliser un appareil android à la place quand c'est possible



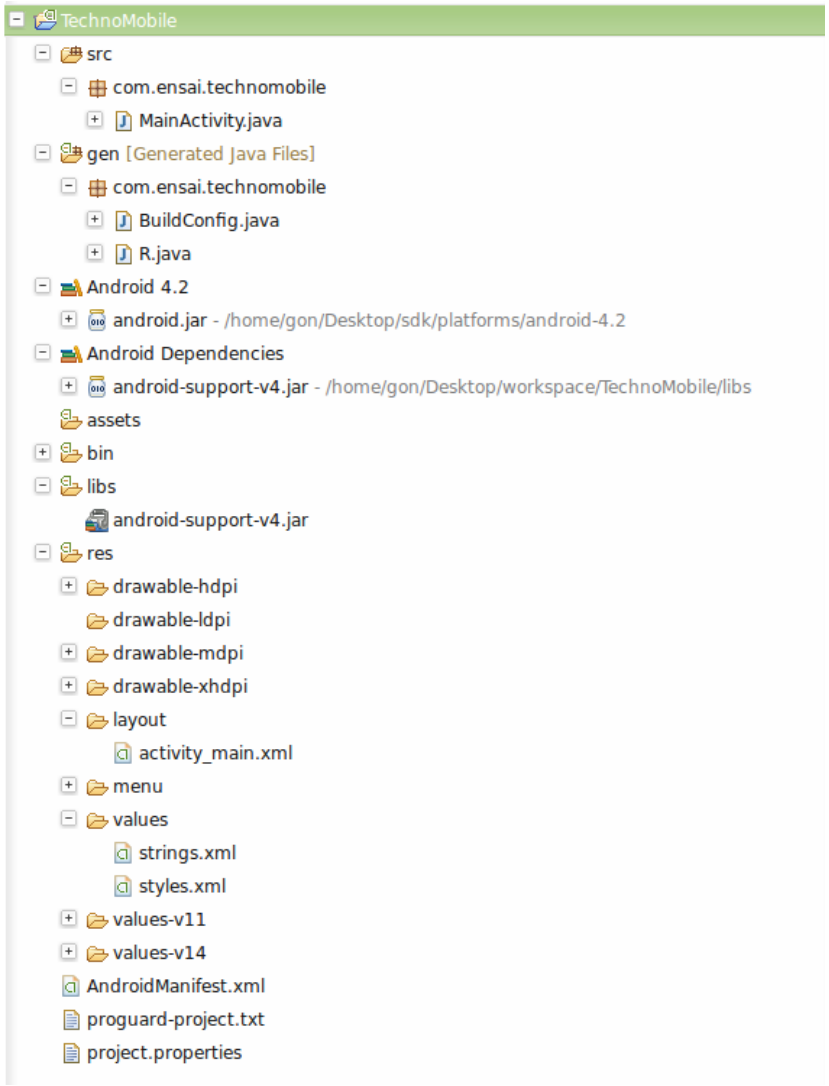
Alternative à l'émulateur

- Problème : émuler de l'ARM sur nos machines x86
- Résultat : émulateur ((très) très) lent
- Solution proposée : porter android sur x86
- `http://www.android-x86.org/`
- `http://www.androvm.org/`
- 2012 : premiers appareils android sous x86 "intel inside"



Sommaire

Organisation d'un projet android



Détail de l'organisation

- src : code source java
- gen : identifiants des ressources (généré par le sdk)
- Android 4.2 : jar correspondant à l'API cible
- Android Dependencies : jar rajoutés, correspond à libs
- assets : fichiers fournis avec l'app
- bin : résultat de la compilation (dont l'apk)
- libs : jar rajoutés
- res : ressources (layouts, strings, images . . .)
- AndroidManifest.xml : métadonnées sur l'application, composants, permissions . . .
- proguard-project.txt : configuration de proguard
- project.properties : généré par le sdk

AndroidManifest.xml : le coeur de l'application

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

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.ensai.technomobile.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>
```

- Déclaration des composants
- Déclaration des permissions
- Déclaration d'autres métadonnées de l'application
- Analysé par l'OS à l'installation

Les permissions

- Obligatoires pour certaines fonctions (internet, géolocalisation, hardware . . .)
- Les applications peuvent définir leurs propres permissions
- L'utilisateur est prévenu à l'installation / mise à jour

```
1 <uses-permission android:name="android.permission.INTERNET" />
2 <uses-permission android:name="android.permission.READ_SMS" />
3 <uses-permission android:name="android.permission.CAMERA" />
```

Le système de ressources

```
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int menu_settings=0x7f070000;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
    public static final class menu {
        public static final int activity_main=0x7f060000;
    }
    public static final class string {
        public static final int app_name=0x7f040000;
        public static final int hello_world=0x7f040001;
        public static final int menu_settings=0x7f040002;
    }
```

- Un identifiant est généré pour chaque ressource (drawable, layout, menu, values, style . . .)
- Toutes les fonctions qui utilisent des ressources sont surchargées pour aussi accepter l'identifiant de la ressource correspondante
- Utiliser des ressources différentes en fonction de la configuration (values et values-fr, drawable et drawable-hdpi)

Exemple de ressources : les strings

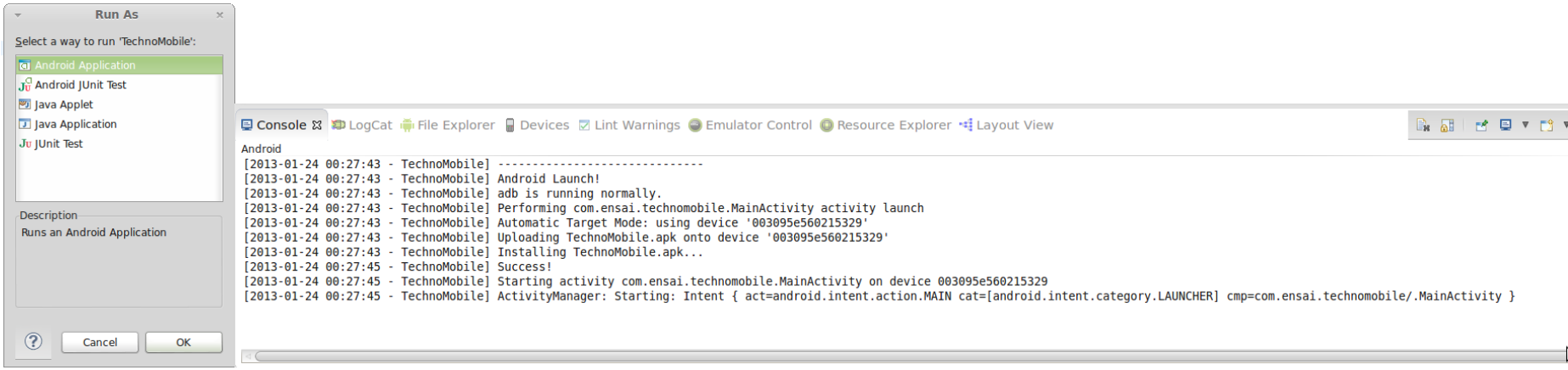
- Eviter au maximum les chaînes de caractères en dur
- Mettre toutes les chaînes dans res/values/strings.xml
- Très facile de traduire ensuite : res/values-en, res/values-fr ... (values et values-fr, drawable et drawable-hdpi)

```
1 <resources>
2     <string name="app_name">Technomobile</string>
3     <string name="hello_world">Hello world!</string>
4     <string name="menu_settings">Settings</string>
5     <string-array name="statuts">
6         <item>Fonctionnaire</item>
7         <item>Ingenieur</item>
8     </string-array>
9 </resources>
```

Déployer l'application

- Une application android = un APK (+/- équivalent d'un jar)
- Une application android doît être signée
- Attention à ne pas perdre la clé !
- Création et signature de l'APK simple sous eclipse (export)

Processus de déploiement en dev

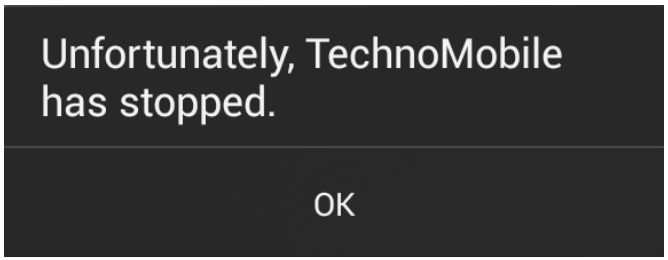


- Comme pour une application JAVA classique, ctrl + F11
- Eclipse demande au SDK de builder l'APK
- Eclipse signe l'APK avec la clé debug
- Eclipse demande à adb (SDK) d'installer l'application
- Soit sur un appareil android connécté soit sur un émulateur

Distribuer l'application

- Distribution directe de l'APK (ex : pour tester, béta fermée)
- Publication sur le playstore, 25$ à l'inscription
- Application gratuite ou payante (30% pour google)

Déboguer l'application



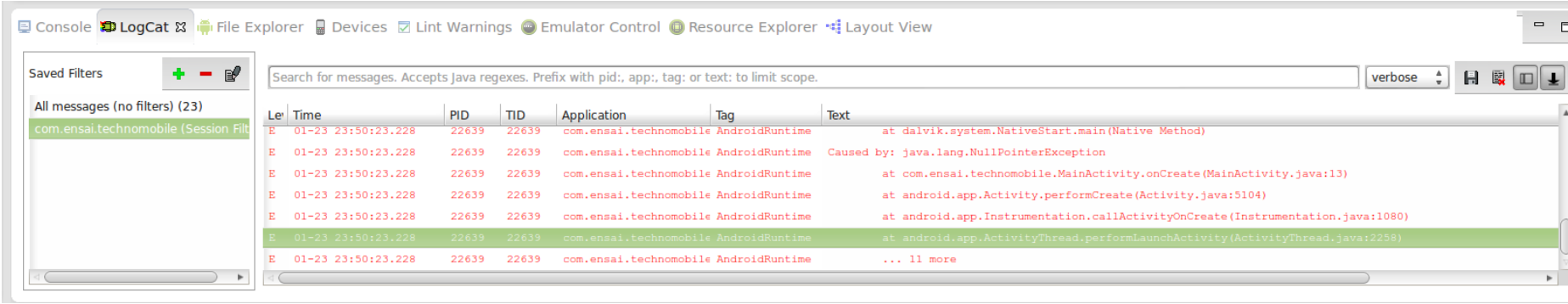
- Si une exception n'est pas rattrapée, android tue l'application
- On parle de "force close" (FC)
- Comment déboguer une application qui tourne sur un appareil (ou émulateur) ?

Stacktrace of GTFO

Logcat, le sauveur

- Le SDK fournit un outil très pratique : logcat
- On appelle logcat par adb : adb logcat ou on utilise la vue LogCat du plugin ADT
- Logcat affiche l'ensemble des logs, système et application

Logcat, exemple




```
gon@gon-Macmini ~ $ adb logcat -s "AndroidRuntime"
--------- beginning of /dev/log/system
--------- beginning of /dev/log/main
D/AndroidRuntime(22495):
D/AndroidRuntime(22495): >>>>>> AndroidRuntime START com.android.internal.os.RuntimeInit <<<<<<
D/AndroidRuntime(22495): CheckJNI is OFF
D/AndroidRuntime(22495): Calling main entry com.android.commands.pm.Pm
D/AndroidRuntime(22495): Shutting down VM
D/AndroidRuntime(22599):
D/AndroidRuntime(22599): >>>>>> AndroidRuntime START com.android.internal.os.RuntimeInit <<<<<<
D/AndroidRuntime(22599): CheckJNI is OFF
D/AndroidRuntime(22599): Calling main entry com.android.commands.am.Am
D/AndroidRuntime(22599): Shutting down VM
D/AndroidRuntime(22639): Shutting down VM
E/AndroidRuntime(22639): FATAL EXCEPTION: main
E/AndroidRuntime(22639): java.lang.RuntimeException: Unable to start activity ComponentInfo{com.ensai.technomobile/com.ensai.technomobile.MainActivity}: java.lang.NullPointerException
E/AndroidRuntime(22639):        at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2304)
E/AndroidRuntime(22639):        at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2354)
E/AndroidRuntime(22639):        at android.app.ActivityThread.access$600(ActivityThread.java:150)
E/AndroidRuntime(22639):        at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1244)
E/AndroidRuntime(22639):        at android.os.Handler.dispatchMessage(Handler.java:99)
E/AndroidRuntime(22639):        at android.os.Looper.loop(Looper.java:137)
E/AndroidRuntime(22639):        at android.app.ActivityThread.main(ActivityThread.java:5191)
E/AndroidRuntime(22639):        at java.lang.reflect.Method.invokeNative(Native Method)
E/AndroidRuntime(22639):        at java.lang.reflect.Method.invoke(Method.java:511)
E/AndroidRuntime(22639):        at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:795)
E/AndroidRuntime(22639):        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:562)
E/AndroidRuntime(22639):        at dalvik.system.NativeStart.main(Native Method)
E/AndroidRuntime(22639): Caused by: java.lang.NullPointerException
E/AndroidRuntime(22639):        at com.ensai.technomobile.MainActivity.onCreate(MainActivity.java:13)
E/AndroidRuntime(22639):        at android.app.Activity.performCreate(Activity.java:5104)
E/AndroidRuntime(22639):        at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1080)
E/AndroidRuntime(22639):        at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2258)
E/AndroidRuntime(22639):        ... 11 more
```

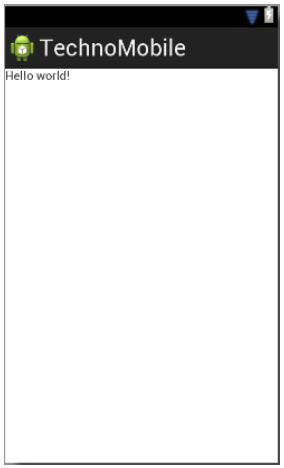
Logcat, logging de l'application

- Oublier System.out.println() !
- 5 niveaux de gravité : ERROR, WARN, INFO, DEBUG, VERBOSE
- Possibilité dans adb logcat de filtrer par gravité et/ou TAG
- Pratique recommandée : un TAG par application

```
1  public void sauvegarderScore(int score) {
2          Log.i(TAG,"Sauvegarde du score "+score);
3          if (score <= 0) {
4              Log.w(TAG,"Le score est negatif");
5          }
6          try {
7              //traitement
8          }
9          catch (Exception e) {
10          Log.e(TAG,"Erreur de score",e);
11         }
12 }
```

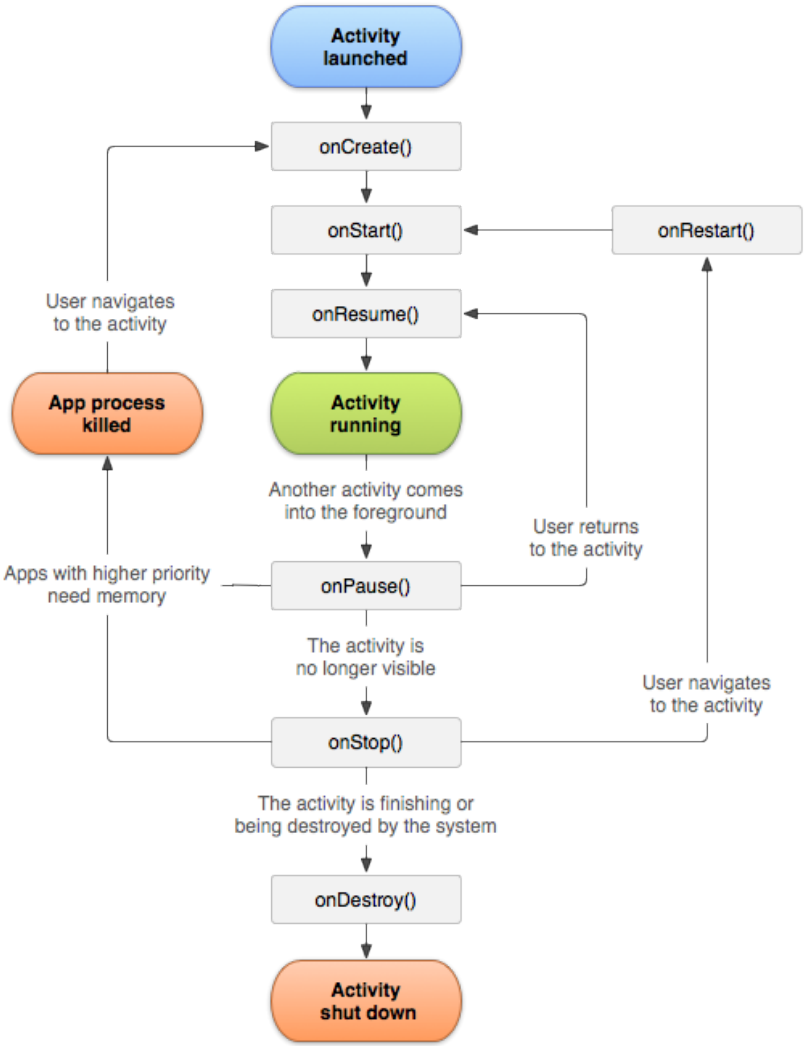
Sommaire

Activity, le composant de base



- 1 activity ~ un écran
- Une application peut avoir 0-n activities
- A ajouter dans le manifest
- Créer une classe java héritant de Activity

Cycle de vie d'une activity



Créer une activity : étendre Activity

```
1  public Class MyActivity extends Activity {
2
3      @Override
4      protected void onCreate(Bundle savedInstanceState) {
5          super.onCreate(savedInstanceState);
6          setContentView(R.layout.activity_main);
7      }
8  }
```

- onCreate est appellé à la création de l'activity (cf cycle de vie)
- appel obligatoire à super.onCreate
- le bundle savedInstanceState contient les informations en cas de relancement de l'activity
- savedInstanceState est null s'il s'agit du premier lancement

L'organisation d'une activity : les layouts

```
1  <LinearLayout
2  xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent">
5
6      <TextView
7          android:layout_width="wrap_content"
8          android:layout_height="wrap_content"
9          android:text="@string/hello_world" />
10
11 </LinearLayout>
```

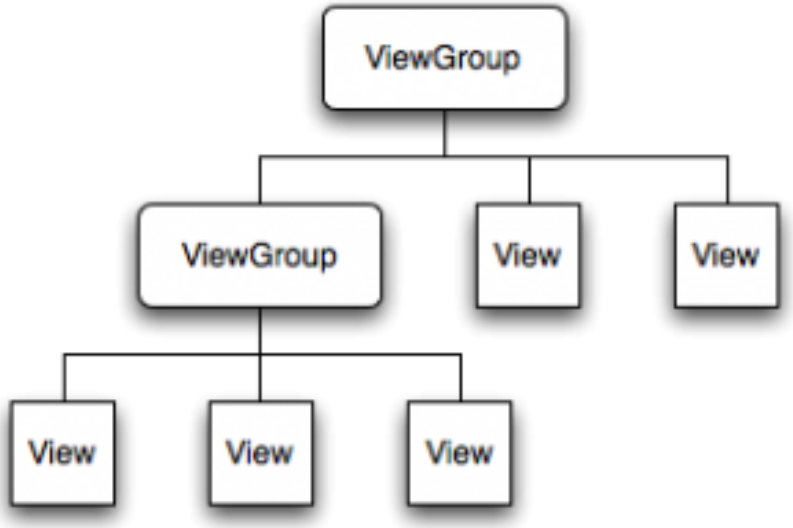
- Ils sont définis en XML dans le dossier res/layout
- Ils définissent l'organisation des vues
- Eviter au maximum de modifier / créer les layouts au runtime

Les Views

Une vue = un élement à l'écran

- TextView = Un texte
- EditText = Un champ de texte remplissable
- ImageView = Une image
- Button
- CheckBox
- Plein d'autres views de base dans android
- Possibilité de créer ses propres views en étendant View ou SurfaceView

Les ViewGroups



- LinearLayout
- RelativeLayout
- ListView
- Plein d'autres
- Les vôtres :)

Manipuler les éléments de l'UI en java

Etape 1 : donner un identifiant à la vue

```
1  <LinearLayout
2  xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:id="@+id/monlayout">
6
7      <Button
8          android:layout_width="wrap_content"
9          android:layout_height="wrap_content"
10         android:id="@+id/monbouton"
11         android:text="@string/hello_world" />
12
13 </LinearLayout>
```

Manipuler les éléments de l'UI en java

Etape 2 : récupérer les réferences vers les views

```
1  public Class MyActivity extends Activity {
2
3      ViewGroup layout = null;
4      Button bouton = null;
5
6      @Override
7      protected void onCreate(Bundle savedInstanceState) {
8          super.onCreate(savedInstanceState);
9          setContentView(R.layout.activity_main);
10         layout = (ViewGroup) findViewById(R.id.monlayout);
11         bouton = (Button) findViewById(R.id.monbouton);
12     }
13 }
```

Manipuler les éléments de l'UI en java

```
1  public Class MyActivity extends Activity {
2
3      ViewGroup layout = null;
4      Button bouton = null;
5
6      @Override
7      protected void onCreate(Bundle savedInstanceState) {
8          super.onCreate(savedInstanceState);
9          setContentView(R.layout.activity_main);
10         layout = (ViewGroup) findViewById(R.id.monlayout);
11         bouton = (Button) findViewById(R.id.monbouton);
12     }
13
14     public void changerTexte(String texte) {
15         bouton.setText(texte);
16     }
17
18     public void cacherTout() {
19         layout.setVisibility(View.INVISIBLE);
20     }
21 }
```

Ecouter les évenements

- Système de listeners (cf swing)
- Il se passe quelque chose sur la vue (touch, focus . . .) : le listener est prévenu
- Pour simplifier, sur android on a en général qu'un listener par évenement et par view (setXListener au lieu de addXListener sous swing)

Ecouter les évenements, guide du bon listener

Etape 1 : Les interfaces XListener

```
1 public Interface OnClickListener {
2     void onClick(View v);
3 }
```

Etape 2 : Implémenter l'interface

```
1 public MaClasse implements OnClickListener {
2     public void onClick(View v) {
3         //Un click a ete fait sur la vue v
4     }
5 }
```

Ecouter les évenements, guide du bon listener

Etape 3 : S'enregistrer comme listener

```
1  public Class MyActivity extends Activity implements
       OnClickListener {
2
3      Button bouton = null;
4
5      @Override
6      protected void onCreate(Bundle savedInstanceState) {
7          super.onCreate(savedInstanceState);
8          setContentView(R.layout.activity_main);
9          bouton = (Button) findViewById(R.id.monbouton);
10         bouton.setOnClickListener(this);
11     }
12
13     public void onClick(View v) {
14         //Un Click a ete fait sur la vue v
15     }
16 }
```

Ecouter les évenements, quelques feintes

Feinte 1 : Utiliser des listeners anonymes

```
1  public Class MyActivity extends Activity implements
       OnClickListener {
2
3      Button bouton = null;
4
5      @Override
6      protected void onCreate(Bundle savedInstanceState) {
7          super.onCreate(savedInstanceState);
8          setContentView(R.layout.activity_main);
9          bouton = (Button) findViewById(R.id.monbouton);
10         bouton.setOnClickListener(new OnClickListener() {
11                 public void onClick(View v) {
12                 //Un Click a ete fait sur la vue v
13                 }
14             });
15     }
16 }
```

Ecouter les évenements, quelques feintes

Feinte 2 : Définir le listener directement dans le layout

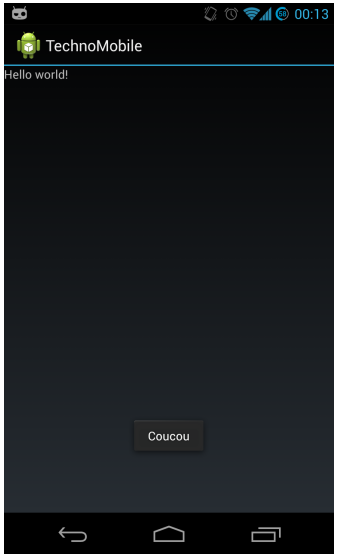
```
1  <LinearLayout
2  xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:id="@+id/monlayout">
6
7      <Button
8          android:layout_width="wrap_content"
9          android:layout_height="wrap_content"
10         android:id="@+id/monbouton"
11         android:text="@string/hello_world"
12         android:onClick="clickSurLeBouton" />
13
14 </LinearLayout>
```

```
1  public void clickSurLeBouton(View v) //dans MyActivity
```

La classe abstraite context

- La plupart des fonctions d'android (accéder à une ressource, lancer une activité ...) nécessitent une instance de Context
- Un Context regroupe des informations globales sur l'environnement de l'application
- Android se charge de créer les contextes
- Activity hérite (indirectement) de Context
- Les views ont toutes une réference vers un context

Affichage d'un court message : le toast




```
1  public MyActivity extends Activity {
2
3      public void faireCoucou() {
4          Toast.makeText(this,"Coucou",Toast.LENGTH_LONG).show
               ();
5          Toast.makeText(this,R.string.coucou,Toast.
               LENGTH_SHORT).show();
6      }
7  }
```

Les autres composants d'une application : les services

- Tâche en arrière plan (attention, pas automatiquement dans un thread séparé)
- Cycle de vie différent de celui d'une activity
- Pas d'interface graphique (sauf si une activity interroge le service)
- Exemples : lecteur MP3, client torrent, système de mise à jour, taskkiller ...

Les autres composants d'une application : les broadcast receivers

- Composant recevant les annonces système et les annonces des autres applications
- Permet de réagir à certains évenements
- Possibilité de lancer des activités ou des services depuis le broadcast receiver
- Exemples : indicateur de batterie, antivirus, lancement au démarrage ...

Les autres composants d'une application : les content-providers

- Composant servant à distribuer les données
- Peu importe comment les données sont stockées (sqlite, préferences, fichier . . .)
- Respecte une interface d'utilisation des données
- Peut permettre la récupération, la modification et/ou la suppression de données
- Principalement déstiné à une utilisation entre applications
- Gestion de la sécurité et des droits (permissions)
- Exemple : accès au données système (SMS, contacts . . .)

Les intents

- On déclare son intention, android réagit en conséquence
- Intents implicites “Je veux ouvrir la page web `https://twitter.com/Ensai35`”
- “Je veux envoyer un mail à jlegouic@ensai.fr avec le titre URGENT : FOOT”
- Intents explicites “Je veux lancer l’activity MyActivity”

Lancer un intent implicite

```
1  public MyActivity extends Activity {
2
3      public void envoyerMail() {
4          Intent i = new Intent(Intent.ACTION_SEND);
5          i.setType("message/rfc822");
6          i.putExtra(Intent.EXTRA_EMAIL   , "annee2@ensai.fr");
7          i.putExtra(Intent.EXTRA_SUBJECT,"URGENT : FOOT");
8          i.putExtra(Intent.EXTRA_TEXT    , "...");
9          try {
10             startActivity(i);
11         }
12         catch (ActivityNotFoundException ex) {
13             //Pas de client mail installe
14         }
15     }
16 }
```

Lancer un intent explicite

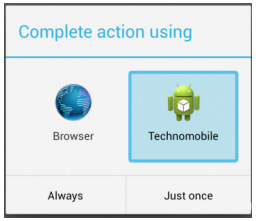
```
1  public MyActivity extends Activity {
2
3      public void lancerMyActivity2() {
4          Intent intent = new Intent(this, MyActivity2.class);
5          //this fait reference a un context
6          startActivity(intent);
7      }
8  }
```

Filtrer les intents, le principe

- startActivity(), startService() et sendBroadcast() déclenchent des intents
- android filtre les composants succeptibles de recevoir l'intent en fonction à partir des intent-filters de chaque composant

```
1  <activity android:name="com.ensai.technomobile.MainActivity"
2              android:label="@string/app_name" >
3              <intent-filter>
4                  <action android:name="android.intent.action.
                       MAIN" />
5                  <category android:name="android.intent.
                       category.LAUNCHER" />
6              </intent-filter>
7          </activity>
```

Filtrer les intents, exemple




```
1  <activity
2              android:name="com.ensai.technomobile.
                   TwitterActivity"
3              android:label="@string/app_name" >
4              <intent-filter>
5                  <data
6                      android:host="twitter.com"
7                      android:scheme="http" />
8                  <category android:name="android.intent.
                       category.DEFAULT" />
9                  <category android:name="android.intent.
                       category.BROWSABLE" />
10                 <action android:name="android.intent.action.
                       VIEW" />
11             </intent-filter>
12         </activity>
```

Sommaire

Il existe plusieurs façons de stocker les données sur un appareil android

- **SharedPreferences** : stockage de données primitives
- **Fichiers** : contenus sur la mémoire interne ou externe
- **Bases de données** : SQLite
- **Stockage distant** : webservice

Documentation android officielle sur le stockage de données : `http://developer.android.com/guide/topics/data/data-storage.html`

SharedPreferences

- Equivalent des Preferences de java
- Stockage de données primitives (int, boolean, String . . .)
- Mapping clé / valeur
- Très facile de créer un écran de paramètres (PreferenceActivity)
- Système de listeners pour être prévenu lors d'un changement
- Léger à implémenter

SharedPreferences, accéder aux préférences

Accéder simple aux préferences :

```
1 public void sauvegarderScore() {
2     SharedPreferences preferences = PreferenceManager.
          getDefaultPreferences(context);
3 }
```

Accéder en précisant le nom et le mode :

```
1 public void sauvegarderScore() {
2     SharedPreferences preferences = context.
          getSharedPreferences("toto",Context.MODE_PRIVATE);
3 }
```

L'utilisation d'autres modes que MODE_PRIVATE est découragé depuis l'API 17 (4.2) pour des raisons de sécurité.

SharedPreferences, utilisation des préférences

Lire les préférences :

```
1  public void afficherMeilleurScore() {
2      SharedPreferences preferences = PreferenceManager.
           getDefaultPreferences(context);
3      int record = preferences.getInt("meilleurScore");
4      Toast.makeText(context, "Record : "+record,Toast.
           LENGTH_LONG).show();
5  }
```

Modifier les préférences

```
1  public void sauvegarderScore() {
2      SharedPreferences preferences = PreferenceManager.
           getDefaultPreferences(context);
3      Editor editor = preferences.edit();
4      editor.putInt("meilleurScore");
5      editor.commit();
6  }
```

Les fichiers

- API classiques de java
- Ne jamais utiliser de chemin absolu, utiliser les fonctions android pour déterminer les dossiers
- Garder en tête les contraintes de place libre
- Choisir entre stockage interne et externe

Stockage interne

```
1 FileOutputStream fos = context.openFileOutput("monFichier.
      ext", Context.MODE_PRIVATE);
2 fos.write("texte".getBytes());
3 fos.close();
```

- Comme pour les préférences, ne pas utiliser les modes WORLD_READABLE et WORLD_WRITEABLE
- MODE_APPEND est utilisable pour écrire à la fin du fichier
- Les fichiers crées sur le stockage interne sont liés à l'application (en particulier, ils sont supprimés lors de la désinstallation de l'application)

- getCacheDir() : répertoire pour le cache
- getFilesDir() : dossier où sont stockés les fichiers de l'application
- deleteFile(String nom)
- fileList() : String[] des fichiers crées par l'application

Stockage externe

- La jungle : l'utilisateur et toutes les applications peuvent lire / écrire tout le contenu
- La présence d'un stockage externe n'est pas garanti
- Même si il est présent, le stockage externe peut ne pas être disponible
- 2 permissions : WRITE_EXTERNAL_STORAGE et READ_EXTERNAL_STORAGE

```
1  String state = Environment.getExternalStorageState();
2  if (Environment.MEDIA_MOUNTED.equals(state)) {
3      //Lecture et ecriture possibles
4  }
5  else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state))
       {
6      //Ecriture impossible, lecture possible
7  }
8  else {
9      //Lecture et ecriture impossibles
10 }
```

Stockage externe, utilisation

Pour le contenu utilisé uniquement par l'application :

```
1 File file = new File(getExternalFilesDir(null), "DemoFile.
      jpg");
```

Ces fichiers seront supprimés lors de la désinstallation de l'application
Pour le contenu déstiné à être partagé (donc persistant même après une désinstallation) :

```
1 File file = new File(getExternalStoragePublicDirectory(
      Environment.DIRECTORY_PICTURES), "DemoFile.jpg");
```

L'argument utilisé dans les 2 cas correspond au type de données et est une variable static de Environment
Exemples : Environment.DIRECTORY_PICTURES, DIRECTORY_MUSIC, DIRECTORY_DOWNLOADS, DIRECTORY_DCIM ...

SQLite, le moteur de base de données embarqué

- SQLite est un moteur de base de données spécialement conçu pour le mobie et l'embarqué
- On retrouve à peu près l'ensemble des fonctions de base d'un moteur de BDD
- Android propose en plus une API facilitant les tâches courantes : SQLiteOpenHelper
- Pour les requêtes, android offre une API proche de JDBC

Mise en place d'une base de données SQLite

Etape 1 : étendre SQLiteOpenHelper

```
1  public class MyOpenHelper extends SQLiteOpenHelper {
2
3      private static final int DATABASE_VERSION = 1;
4      private static final String DATABASE_NAME = "mabase";
5
6      public MyOpenHelper(Context context) {
7          super(context, DATABASE_NAME, null, DATABASE_VERSION
               );
8      }
9  }
```

Etape 2 : implémenter onCreate

```
1      @Override
2      public void onCreate(SQLiteDatabase db) {
3          db.execSQL("CREATE TABLE exemple (nom TEXT PRIMARY
               KEY, score INTEGER));
4      }
```

Mise en place d'une base de données SQLite

Etape 3 : Organiser les mises à jour de la base

```
1  public class MyOpenHelper extends SQLiteOpenHelper {
2      @Override
3      public void onUpgrade(SQLiteDatabase db, int oldVersion,
           int newVersion) {
4          //Mise a jour de la base si besoin
5      }
6  }
```

- L'entier DATABASE_VERSION permet de savoir si une mise à jour de la base est nécessaire
- Android appelle directement onUpgrade si le numéro de version est supérieur au numéro de version actuel de la base
- onUpgrade doît alors faire les mises à jour qui s'impose en fonction des entiers oldVersion / newVersion

Utilisation de SQLite

Récupérer une instance de SQLiteDatabase

```
1 SQLiteOpenHelper helper = new SQLiteOpenHelper(context);
2 SQLiteDatabase writableDB = helper.getWritableDatabase();
3 SQLiteDatabase readableDB = helper.getReadableDatabase();
```

- Android propose un certain nombre de fonctions utilitaires pour le requêtage, l'insertion, la mise à jour et la suppresion de données
- Penser à fermer la base avec .close() pour libérer des ressources

Utilisation de SQLite, les requêtes

- 2 méthodes au choix en fonction du besoin
- Ces méthodes sont déclinées en de multiples méthodes (surcharge)
- rawQuery(String sql, String[] selectionArgs) pour du sql dur (~ PreparedStatement)
- query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy) pour que android génère le sql

```
1 SQLiteDatabase writableDB = helper.getWritableDatabase();
2 Cursor cursor1 = writableDB.rawQuery("SELECT nom, score FROM
      table WHERE id = ?",new String[] {"42"});
3 Cursor cursor2 = writableDB.query("table",new String[] {"
      nom","score"}, "id = ?",new String[] {"42"}, null, null
      , null );
```

Utilisation de SQLite, les cursors

- Wrapper autour d'un ResultSet
- Utilisation très proche des resultSets
- Penser à le fermer (close()) pour libérer des ressources

```
1 int nbRows = cursor.getCount();
2 while (cursor.moveToNext()) {
3     String nom = cursor.getString(0);
4     int score = cursor.getInt(1);
5 }
6 cursor.close();
```

Utilisation de SQLite : update, insert et delete

long insert (String table, String nullColumnHack, ContentValues values)

```
1 ContentValues values = new ContentValues();
2 values.put("nom", "bob");
3 values.put("score", 42);
4 long rowID = bdd.insert(table, null, values);
```

int update(String table, ContentValues values, String whereClause, String[] whereArgs)

```
1 ContentValues values = new ContentValues();
2 values.put("score", 42);
3 int nbRowsAffected = bdd.update(table, values, "nom = ?",new
      String[] {"42"});
```

int delete (String table, String whereClause, String[] whereArgs)

```
1 int nbRowsAffected = bdd.delete(table,"nom = ?", new String
      [] {"bobette});
```