ANDROID
Développement

Mars-Avril-Mai 2019

Formation Android – Tristan SALAUN

ynov AIX-EN-PCE

Présentations

- Qui suis-je
  - Historique
  - Actuellement
- Qui vous êtes:
  - Tour de table:
    - Niveau en Java
    - Niveau en mobile en général

Développeur/Formateur Android

Ce que je fais maintenant

https://light4events.fr/

STARTUP
MARSEILLE

https://startupmarseille.com/

STDev

Objectifs

- Maîtriser la programmation d'interfaces
- Connaître les techniques fondamentales du développement Android
- Être capable de développer une application conviviale et dynamique fonctionnant sur la plateforme Android
- Savoir gérer les spécificités des différents supports, smartphones et tablettes

Chapitres

- Présentation d'Android
- Environnement de Développement ANDROID STUDIO
- Point sur la structure, l'exécution et le déploiement d'une application
- Application statique sous Android
- Application dynamique sous Android
- Notifications utilisateur
- Gestion de l'interaction utilisateur

Chapitres

- Activity, ListView et RecyclerView
- Interfaces adaptables et évolutives : les Fragments
- Persistance temporaire
- Persistance durable
- Tâches asynchrones et tâches de fond
- Des programmes sans interface : Les Services
- Autres fonctions du Terminal

Chapitre 1 : La base

- Présentation
- Historique:
  - http://fr.wikipedia.org/wiki/Historique_des_versions_d'Android
  - https://developer.android.com/about/versions/pie/
  - https://developer.android.com/about/dashboards/index.html
- Google play: https://play.google.com/apps/publish
- Documentation: http://developer.android.com/index.html
- Cours en ligne : http://android.developpez.com/cours/
- Livre de référence : http://www.editions-eni.fr/livre/android-coffret-de-2-livres-apprenez-a-developper-vos-applications-en-java-9782409001017

Présentation d'Android

Mars-Avril-Mai 2019

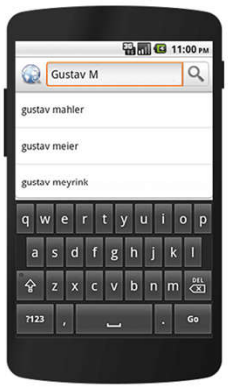
Formation Android – Tristan SALAUN

ynov AIX-EN-PCE

Historique de la plate-forme

1.6 Donut (2009)

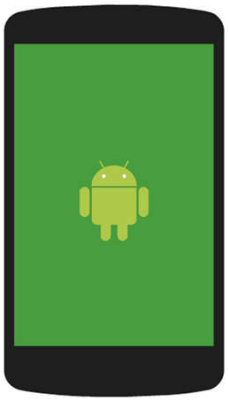
Apparu dès la version Donut, le champ de recherche rapide, accessible à tout moment sur l'écran d'accueil, permet d'effectuer des recherches à la fois sur le Web et dans le contenu local de votre téléphone.



Historique de la plate-forme

1.6 Donut (2009)

La plate-forme Android s'adapte à l'ensemble des formes et tailles d'écran grâce aux nouvelles fonctionnalités de Donut qui la rendent compatible avec toute une variété de formats et de résolutions, ouvrant ainsi la porte aux téléphones dotés d'une résolution différente de 320 x 480 en portrait.



Historique de la plate-forme

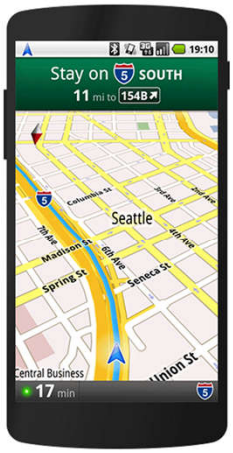
1.6 Donut (2009)

Avant Google Play, il y avait l'Android Market. Lancé en 2008, ce dernier est repensé pour la sortie de Donut afin de présenter le top des applications gratuites et payantes au moment où commence à exploser le catalogue d'applications tierces Android.

Historique de la plate-forme

2.1 Eclair (2010)

Google Maps Navigation donne un nouveau sens au mot "smartphone". Ce système de navigation propose des itinéraires détaillés s'appuyant sur des données Google Maps, avec également de nombreuses fonctionnalités propres aux systèmes embarqués classiques, telles qu'une vue en 3D novatrice, un guidage vocal et des informations sur la circulation, le tout gratuitement.



Historique de la plate-forme

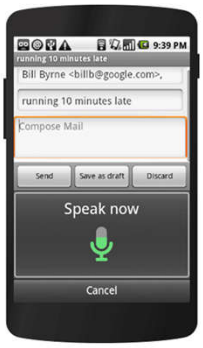
2.1 Eclair (2010)

On peut désormais obtenir une plus hautes résolution 854*480 et des fonds d'écrans animés.

Historique de la plate-forme

2.1 Eclair (2010)

Bien avant "Ok Google", vous pouviez appuyer sur l'icône en forme de micro de votre téléphone pour énoncer des commandes vocales. Eclair a remplacé la touche Virgule du clavier virtuel par un micro. Il vous suffit d'appuyer sur cette icône pour que le texte énoncé s'affiche directement à l'écran.



Historique de la plate-forme

2.2 Froyo (2010-2011)

Froyo intègre non seulement la compilation à la volée dans la machine virtuelle Dalvik pour des applications jusqu'à cinq fois plus performantes, mais également le moteur JavaScript V8 dans le navigateur Android pour des performances JavaScript entre deux et trois fois supérieures.

Historique de la plate-forme

2.2 Froyo (2010-2011)

- Froyo intègre aussi les commandes vocales et le partage de connection wifi

Historique de la plate-forme

2.3 Gingerbread (2010-2011)

- API de Jeu
- NFC
- Gestion de la batterie

Historique de la plate-forme

3.0 HoneyComb (2011-2012)

- Design adapté aux tablettes
- Barre système (fini le temps des boutons physique de retour et de menu.
- Fenêtre de configuration rapide

Historique de la plate-forme

4.0 Ice Cream Sandwich (2011-2012)

- Écran d'accueil personnalisé
- Consommation des données.
- Android BEAM (partager des données via NFC)

Historique de la plate-forme

4.0 Jelly Bean (2012-2013)

- Google Now
- Notifications interactives
- Changement de compte

Historique de la plate-forme

4.0 KitKat (2012-2013)

- Commande vocale « OK GOOGLE »
- Mode immersif
- Clavier intelligent

Historique de la plate-forme

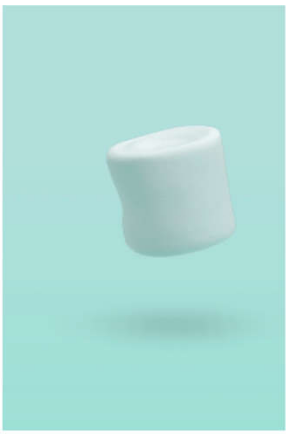
5.0 Lollipop (2014-2015)

- Material Design
- Multi-écran
- Notifications

Historique de la plate-forme

6.0 Marshmallow(2015-2016)

- Now on tap
- Permissions
- Battery



Historique de la plate-forme

7.0 Nougat (Août 2016)

- Multi-Windows UI
- Réponse directe dans les notifications
- Battery
- Quick Settings Tile API
- Number Blocking
- New Emojis

https://developer.android.com/about/versions/nougat/android-7.0.html
https://www.android.com/versions/nougat-7-0/
http://www.phonandroid.com/android-7-1-nougat-nouvelles-fonctionnalites-allez-adorer.html

Historique de la plate-forme

8.0 Oreo (Août 2017)

- Picture-in-picture
- Better notifications
- Autofill
- Better colors
- Better webviews
- Emoji
- More security

https://developer.android.com/about/versions/oreo/index.html
http://www.frandroid.com/android/419174_android-o-developer-preview-quelles-sont-les-nouveautes

Historique de la plate-forme

9.0 Pie (Août 2018)

- Indoor positioning with Wi-Fi RTT
- Display cutout support
- Notifications
- Multi-camera support and camera updates
- ImageDecoder for drawables and bitmaps
- Animation
- HDR VP9 Video, HEIF image compression, and Media APIs
- Data cost sensitivity in JobScheduler
- Neural Networks API 1.1
- Autofill framework
- Security enhancements
- Android backups
- Accessibility
- Rotation
- Text
- …

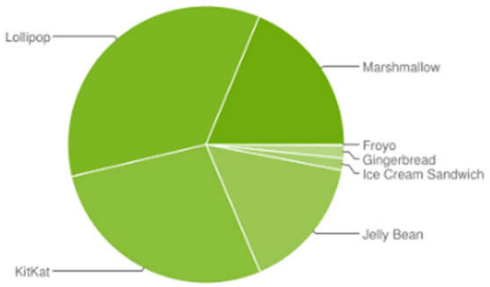
https://developer.android.com/about/versions/pie/

Security : Unification des modules de sécurité (biométriques, ..,)

Quelle cible choisir ?

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.4%
4.1.x	Jelly Bean	16	5.6%
4.2.x		17	7.7%
4.3		18	2.3%
4.4	KitKat	19	27.7%
5.0	Lollipop	21	13.1%
5.1		22	21.9%
6.0	Marshmallow	23	18.7%

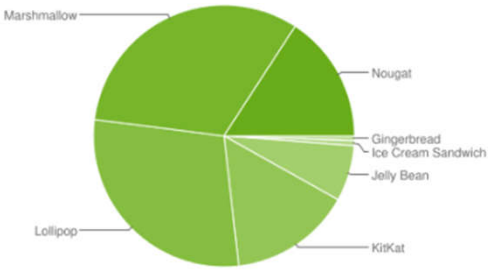
Data collected during a 7-day period ending on September 5, 2016.
Any versions with less than 0.1% distribution are not shown.



https://developer.android.com/about/dashboards/index.html

Quelle cible choisir ?

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.6%
4.1.x	Jelly Bean	16	2.4%
4.2.x		17	3.5%
4.3		18	1.0%
4.4	KitKat	19	15.1%
5.0	Lollipop	21	7.1%
5.1		22	21.7%
6.0	Marshmallow	23	32.2%
7.0	Nougat	24	14.2%
7.1		25	1.6%

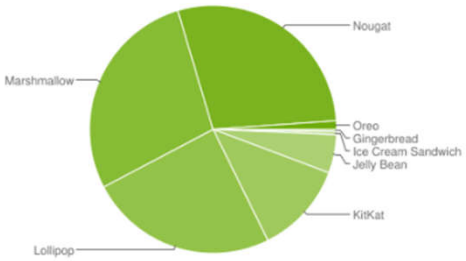


Data collected during a 7-day period ending on September 11, 2017.
Any versions with less than 0.1% distribution are not shown.

https://developer.android.com/about/dashboards/index.html

Quelle cible choisir ?

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.7%
4.2.x		17	2.6%
4.3		18	0.7%
4.4	KitKat	19	12.0%
5.0	Lollipop	21	5.4%
5.1		22	19.2%
6.0	Marshmallow	23	28.1%
7.0	Nougat	24	22.3%
7.1		25	6.2%
8.0	Oreo	26	0.8%
8.1		27	0.3%

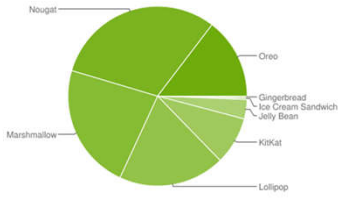


Data collected during a 7-day period ending on February 5, 2018.
Any versions with less than 0.1% distribution are not shown.

https://developer.android.com/about/dashboards/index.html

Quelle cible choisir ?

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.8%
4.3		18	0.5%
4.4	KitKat	19	8.6%
5.0	Lollipop	21	3.8%
5.1		22	15.4%
6.0	Marshmallow	23	22.7%
7.0	Nougat	24	20.3%
7.1		25	10.5%
8.0	Oreo	26	11.4%
8.1		27	3.2%

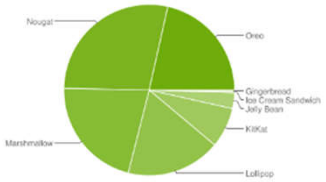


Data collected during a 7-day period ending on August 31, 2018.
Any versions with less than 0.1% distribution are not shown.

https://developer.android.com/about/dashboards/index.html

Quelle cible choisir ?

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x		17	1.5%
4.3		18	0.4%
4.4	KitKat	19	7.6%
5.0	Lollipop	21	3.5%
5.1		22	14.4%
6.0	Marshmallow	23	21.3%
7.0	Nougat	24	18.1%
7.1		25	10.1%
8.0	Oreo	26	14.0%
8.1		27	7.5%

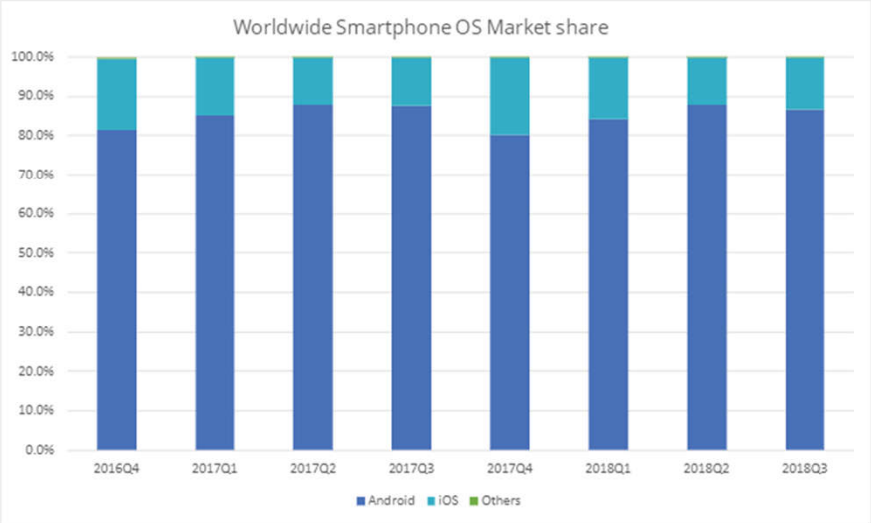


Data collected during a 7-day period ending on October 26, 2018 (update coming soon: data feed under maintenance).
Any versions with less than 0.1% distribution are not shown.

https://developer.android.com/about/dashboards/index.html

Diffusion actuelle

Android c'est + de 80% du marché des téléphones mobiles



http://www.idc.com/prodserv/smartphone-os-market-share.jsp
http://www.idc.com/prodserv/smartphone-market-share.jsp

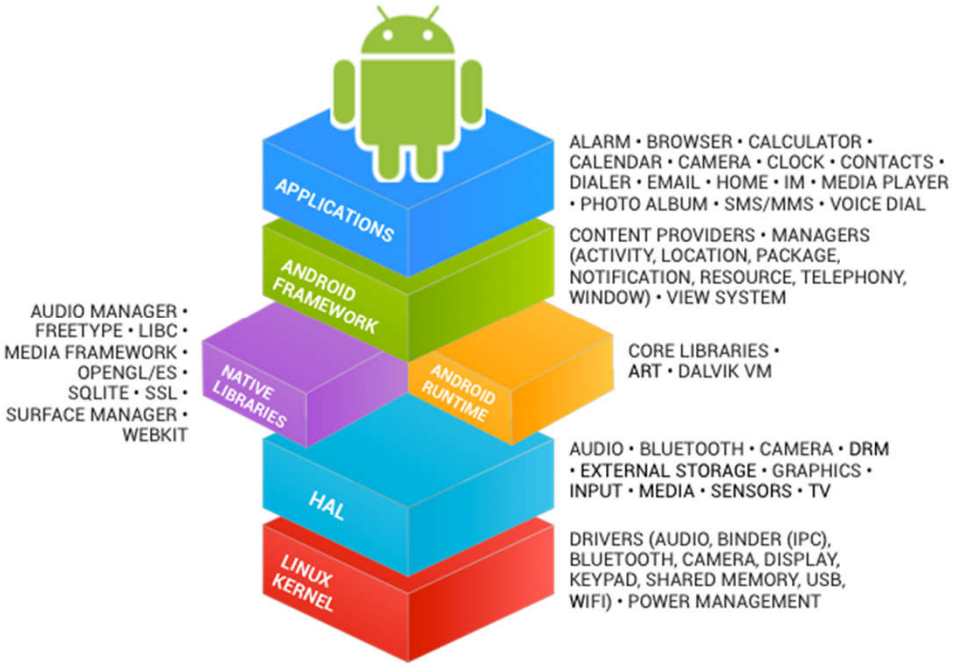
Chapitre 2 : Installation

- Android Studio : https://developer.android.com/studio/index.html
- Emulateurs : http://www.androidauthority.com/best-android-emulators-for-pc-655308/

Chapitre 3 : Les principes

- Optimisation
  - Mémoire
  - Processeur
  - Radio (WiFi, 3G, BT)
  - Graphique

Architecture et aspects techniques

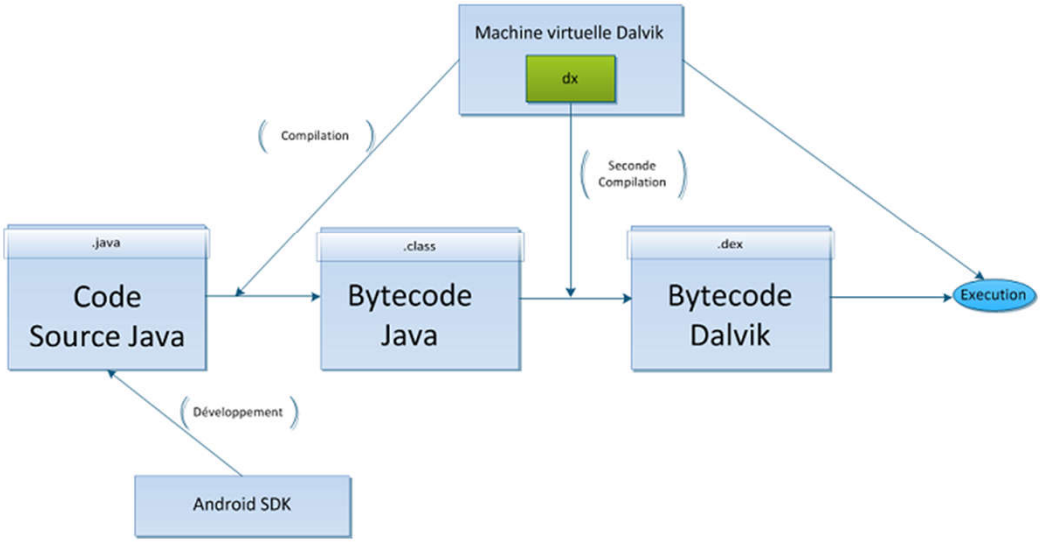


HAL : Hardware Abstraction Layer

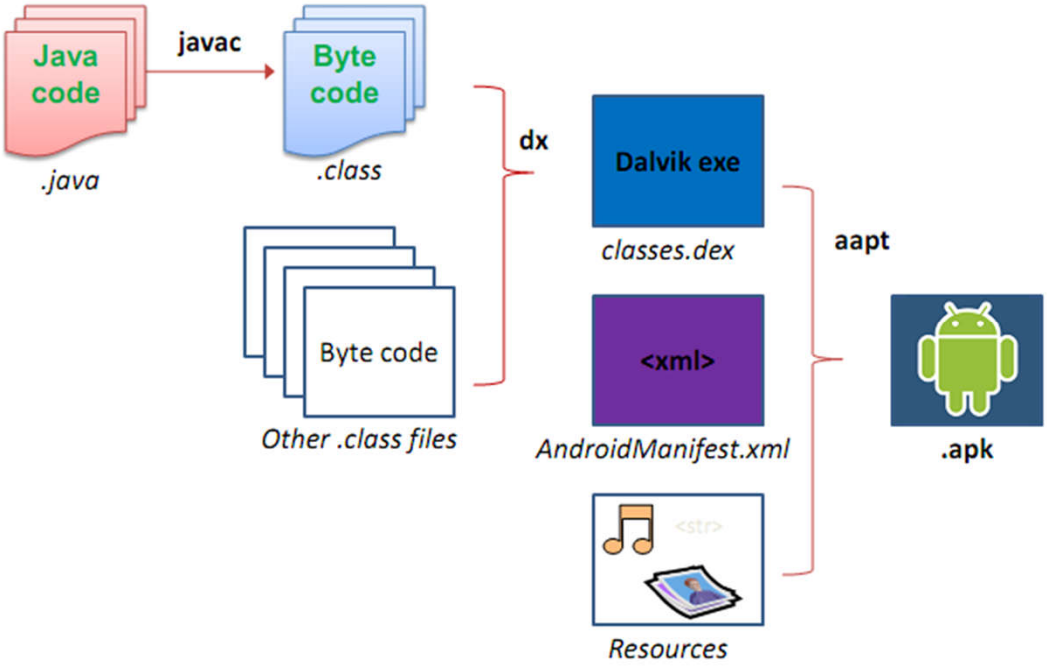
Architecture et aspects techniques

- Le noyau Linux : cette partie contient les drivers et l'accès au matériel. Par exemple, si un constructeur souhaite rajouter de nouveaux matériels, il n'aura que cette partie à modifier.
- Les bibliothèques : cette partie permet de founir des briques logicielles pratiques pour le développeur.
- Runtime (moteur d'exécution).

Architecture et aspects techniques

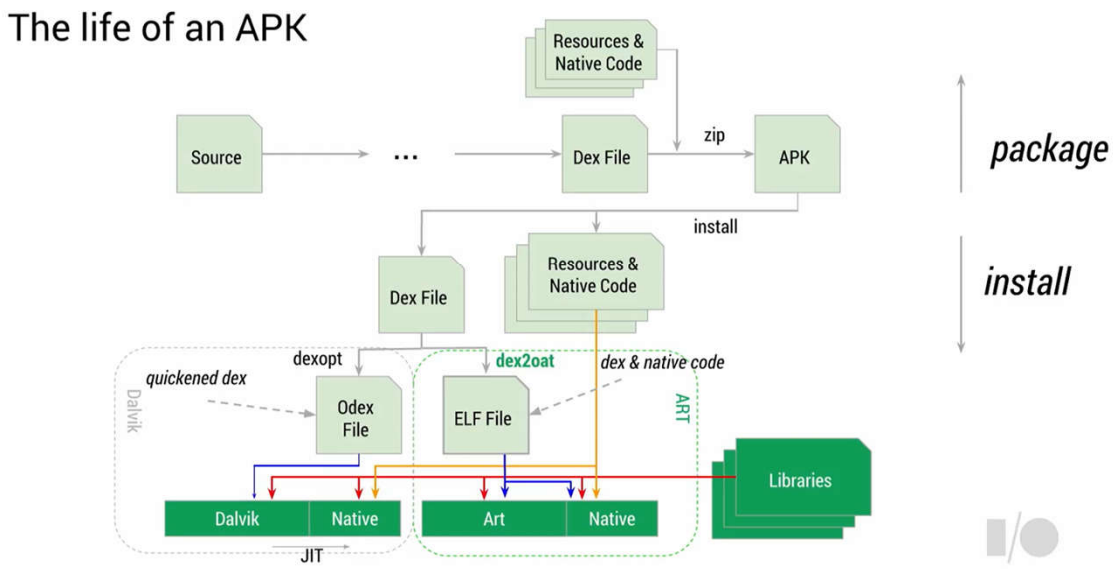


Architecture et aspects techniques



http://www.theappguruz.com/blog/android-compilation-process
https://manifestsecurity.com/android-application-security-part-3/

Architecture et aspects techniques



https://www.anandtech.com/show/8231/a-closer-look-at-android-runtime-art-in-android-l

Objectif : booster la vitesse.
Comment : compilation lors de l'installation.

Mises à jour OTA : A/B partitions
https://source.android.com/devices/tech/ota/ab
https://www.xda-developers.com/how-a-b-partitions-and-seamless-updates-

affect-custom-development-on-xda/

NDK

- C/C++
- JNI
- Lien: http://developer.android.com/tools/sdk/ndk/index.html

APK

- ZIP
- Code
- Ressources
- Assets
- Certificats
- Manifeste

Ressources : Gestion des différentes versions par l'OS (taille écran, rotation, langue, …), utilisation d'un ID => pas d'erreur de nommage.
Assets : Rien de tout cela, données brutes. On peut utiliser des répertoires.

Composantes Android

- Activity
- Fragment (3.0+)
- Service
- Content providers
- Broadcast receivers
- Intent
- Processes et Threads
- AndroidManifest
- Permissions
- App Widgets

Chapitre 4 : HelloWorld

- Création de l'application
- Lancement de l'application sur émulateur
- Lancement de l'application sur téléphone
- Détail de l'architecture d'un projet

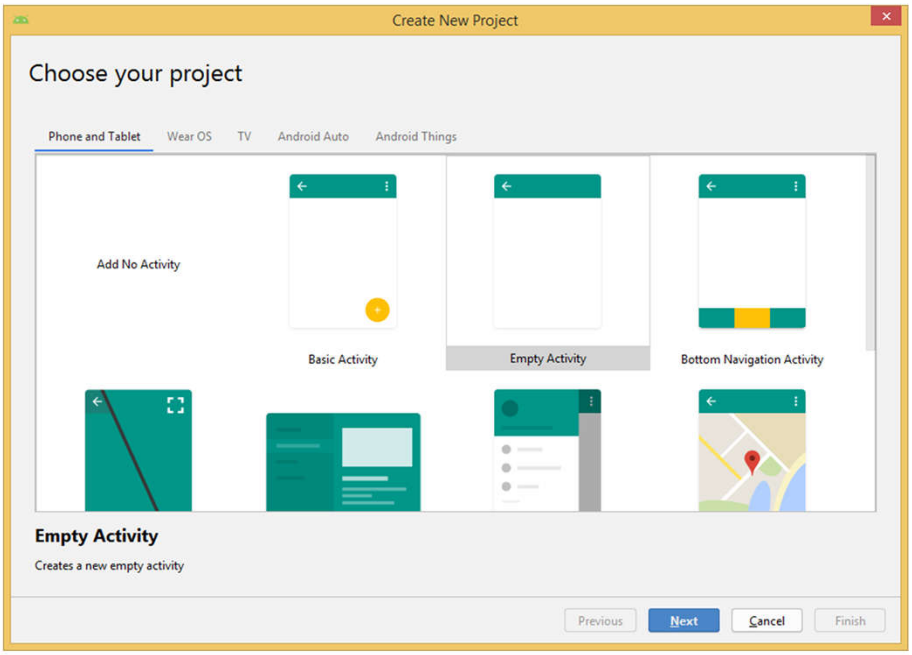
EDI Android Studio

- Environnement de Développement Intégré
- Maintenant avec « Instant Run »
- Éditeur de code intelligent
- Test de votre application sur de nombreux émulateurs
- Configuration des projets flexibles(Gradle)
- Template de code depuis Github pour démarrer un projet.

Attention à l' « Instant Run » qui cause souvent des problèmes.

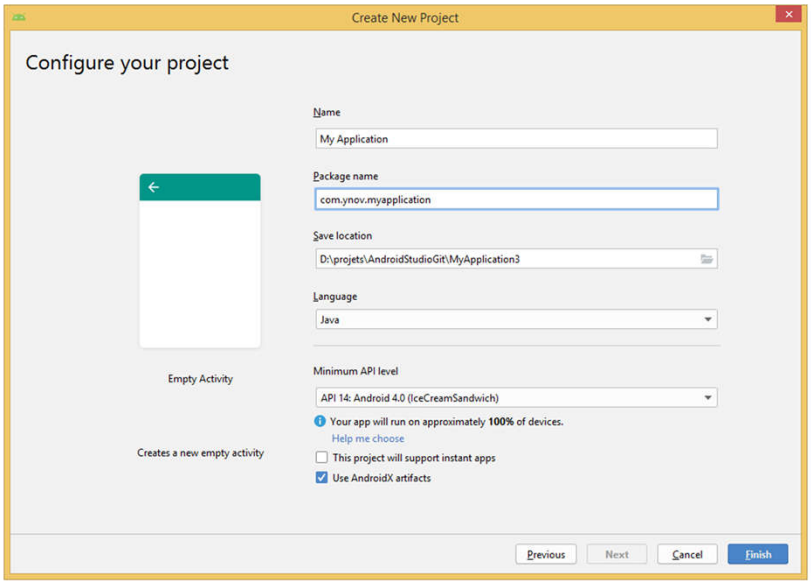
EDI Android Studio

Type d'Activity



EDI Android Studio

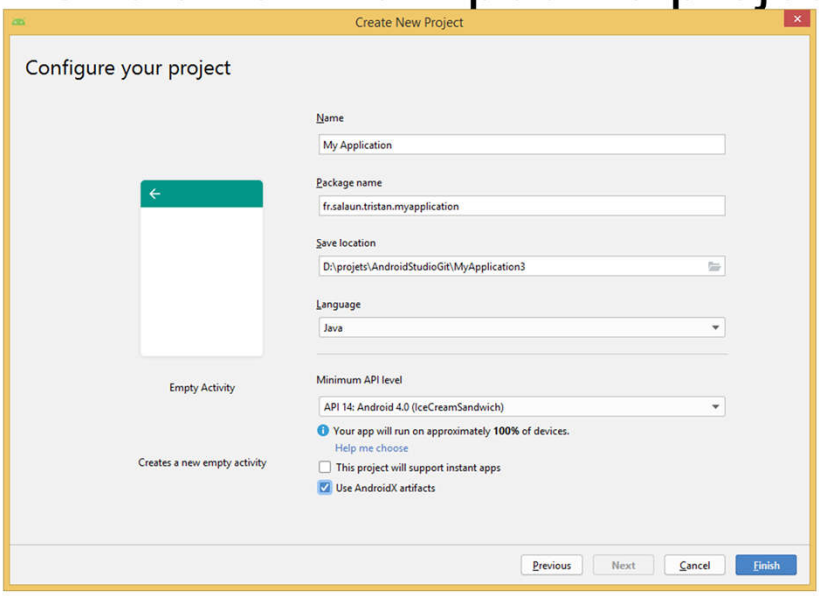
Choisir un nom pour le projet



Instant App : application en streaming sans installer sur le téléphone : https://developer.android.com/topic/google-play-instant

EDI Android Studio

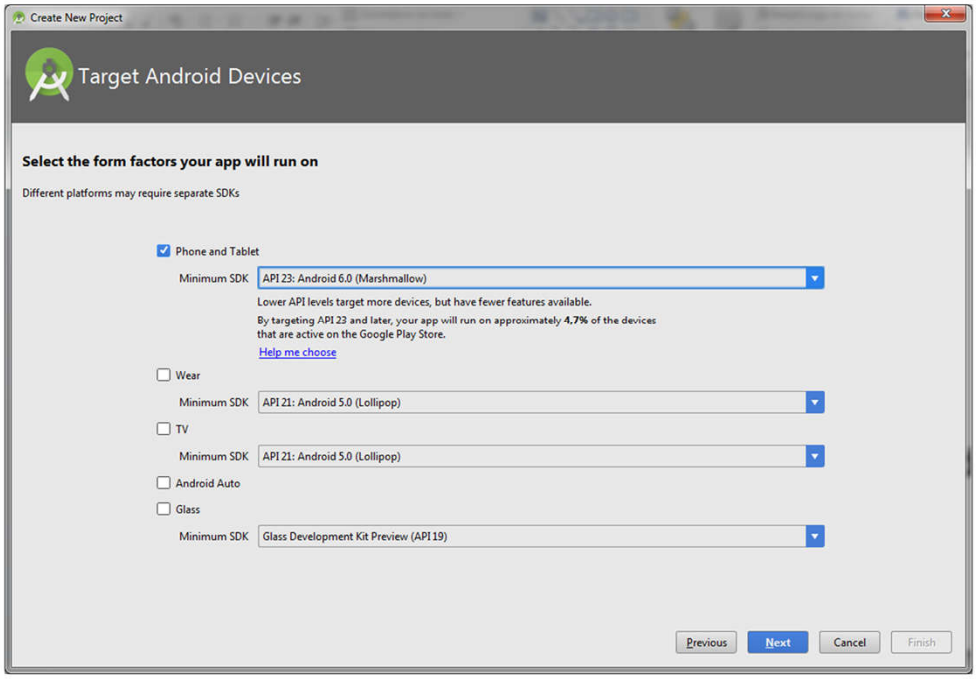
Choisir un nom pour le projet



Instant App : application en streaming sans installer sur le téléphone : https://developer.android.com/topic/google-play-instant

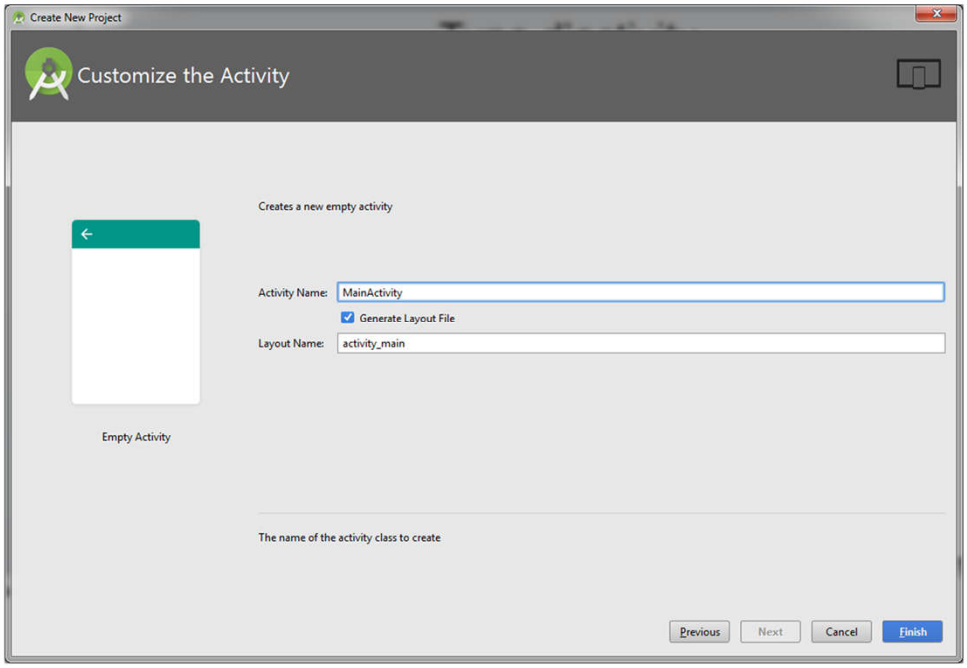
Ensuite il faut :
Choisir le niveau d'API
Choisir la version du SDK
Choisir éventuellement un type d'activity de départ.

Niveau API



Remarquez l'information sur le % d'équipements supportés.

Nom des fichiers



Lancement de l'application (Emulateur)

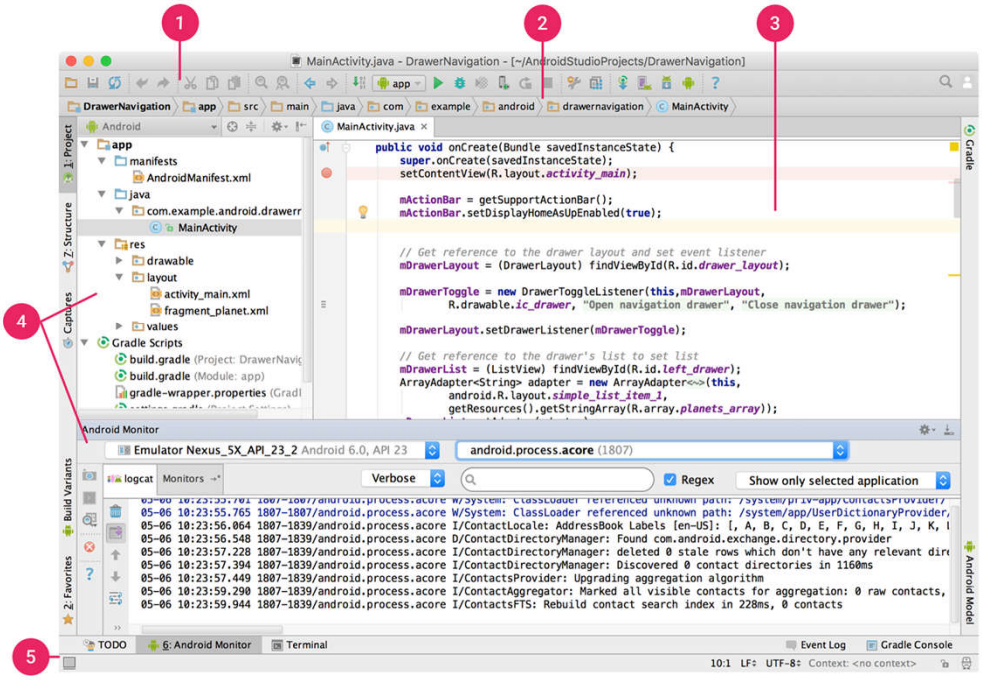
- Lancez l'application
  - Flèche verte
  - Run/Run 'app'
  - Maj + F10
- AVD : Create new Emulator
  - Choix du device
  - Choix de la version de l'OS
  - Nom et paramètres

Lancement de l'application (Téléphone)

- Branchez le téléphone
- Activez le mode développeur

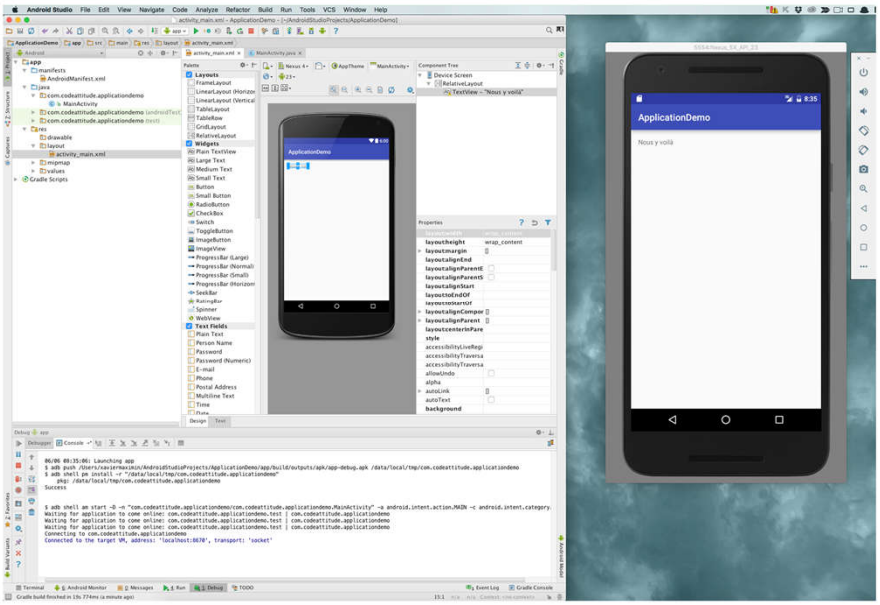
https://www.frandroid.com/comment-faire/tutoriaux/184906_comment-acceder-au-mode-developpeur-sur-android
Cliquer sur le numéro de build

EDI Android Studio



1 : La barre d'outils, pour compiler, lancer, débuguer, …
2 : Le chemin complet du fichier ouvert
3 : Fenêtre de code. Double click sur le titre pour passer en plein écran
4 : Les fenêtres de projet, logs. Quand on clique sur la petite cible, on sélectionnes directement le fichier ouvert.
5 : Onglet pour changer d'outils

EDI Android Studio

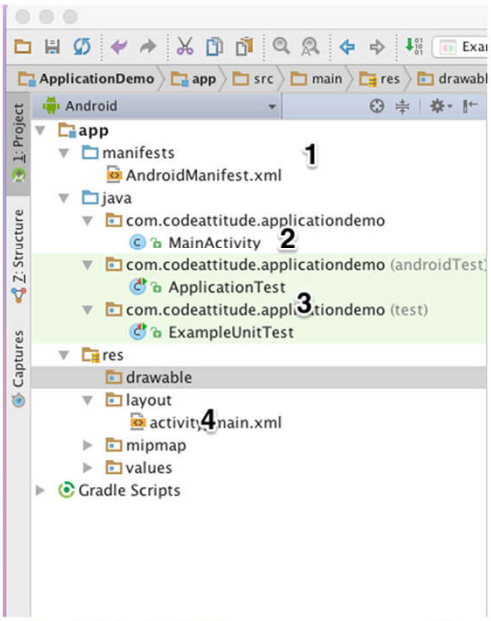


Mars-Avril-Mai 2019 Formation Android – Tristan SALAUN

ynov AIX-EN-PCE

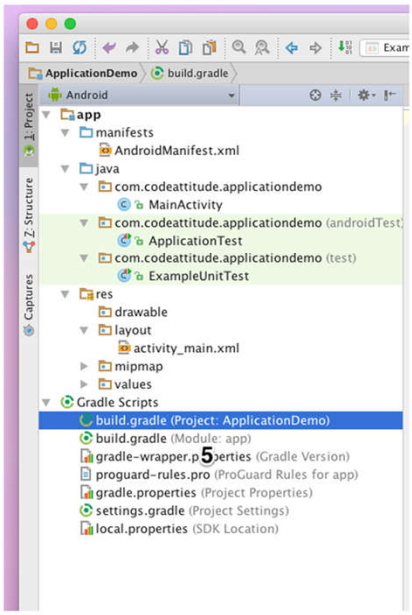
Préview et émulateur.

Point sur la structure, l'exécution et le déploiement d'une application



1. AndroidManifest.xml : fichier de configuration de l'application
2. Activité (écran) de l'application
3. Classes de Test (pour les tests unitaires et d'intégration de l'application)
4. activity_main.xml (le design de l'écran)

Point sur la structure, l'exécution et le déploiement d'une application



5. Gradle : les fichiers qui permettent de configurer le processus de Build d'une application. Le plus souvent ce fichier est utilisé pour ajouter des librairies à l'application.

Point sur la structure, l'exécution et le déploiement d'une application

Les ressources sont des données utilisables par l'application, elles sont intégrées au binaire de l'application

- res/anim : animations de transition au format XML
- res/color : couleurs définies dans des fichiers XML
- res/drawable : images, plusieures densités d'écran sont disponibles (hdpi, mdpi et ldpi)
- des ressources images peuvent être spécifiées pour une configuration matérielle et/ou logicielle précise, le nom du dossier de ressource est alors complété par des qualificatifs séparés par des traits d'union : largeur d'écran, orientation de l'écran, langue système, ...
- res/layout : interfaces graphiques
- res/values : valeurs utilisées par l'application

=> les valeurs sont définies dans un fichier XML

Point sur la structure, l'exécution et le déploiement d'une application

Chaque ressource du dossier res possède un identifiant entier unique

- permet d'atteindre depuis le code Java, ou XML, une ressource – cette valeur peut changer au grès des ajouts et suppression de ressources
- pour éviter le codage en dur de ces valeur, des constantes sont créées dans une classe R.java

```
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 layout {
public static final int main=0x7f030000;
}
public static final class string {
public static final int app_name=0x7f040001;
public static final int hello=0x7f040000;
}
}
```

Point sur la structure, l'exécution et le déploiement d'une application

Le nom complet de la constante est composé :

- du nom du package
  - facultatif car il s'agit du package courant
- du type de ressource
  - correspond au sous dossier res
- du nom de la ressource
  - pour les ressources situées dans res/values, le nom est celui défini dans le fichier xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="hello">Hello World, HelloActivity!</string>
<string name="app_name">Hello</string>
</resources>
```

Point sur la structure, l'exécution et le déploiement d'une application

- syntaxe Java pour accéder à une ressource :
  [package.]R.type.nom
  R.string.app_name
- syntaxe XML pour accéder à une ressource :
  @[package:]type/nom
  @string/app_name

Point sur la structure, l'exécution et le déploiement d'une application

Le fichier AndroidManifest.xml

- contient la configuration principale de l'application

– composants applicatifs, point d'entrée principal, sécurité et droits, etc.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.codeattitude.android.hello"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk android:minSdkVersion="4" />
<application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name" >
<activity
android:label="@string/app_name"
android:name=".HelloActivity" >
<intent-filter >
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

Les balises et attributs de ce fichier XML sont très nombreux

- beaucoup sont facultatives
- Certains seront détaillées au cours de cette formation
  - au fur et à mesure de leur utilisation
- Balise racine : manifest
  - attributs
    - package : package de l'application
    - versionCode : version de l'application (numérique)
    - versionName : version de l'application (string)

Point sur la structure, l'exécution et le déploiement d'une application

Balise application

- informations sur l'application
- attributs

– icon : icône de l'application

- valeur : *@drawable/ic_launcher* qui référencie la ressource ic_launcher.png située dans un des répertoire res/drawable, en fonction de la densité d'écran

– label : titre de l'application – il est préférable d'utiliser des références à des ressources plutôt qu'une chaine de caractères

- valeur : *@string/app_name* qui référencie la ressource de type string contenue dans le fichier *res/values/string.xml*
- application contient des balises enfants

Point sur la structure, l'exécution et le déploiement d'une application

Balises filles

- activity
  – défini une activité (écran)
  – l'activité est lancée par une intention (intent)
- service
  – défini un service
- provider
  – défini un fournisseur de données
- receiver
  – défini un récepteur de messages
- uses-library
  – librairies utilisées

Le contexte applicatif est accessible par la classe Context

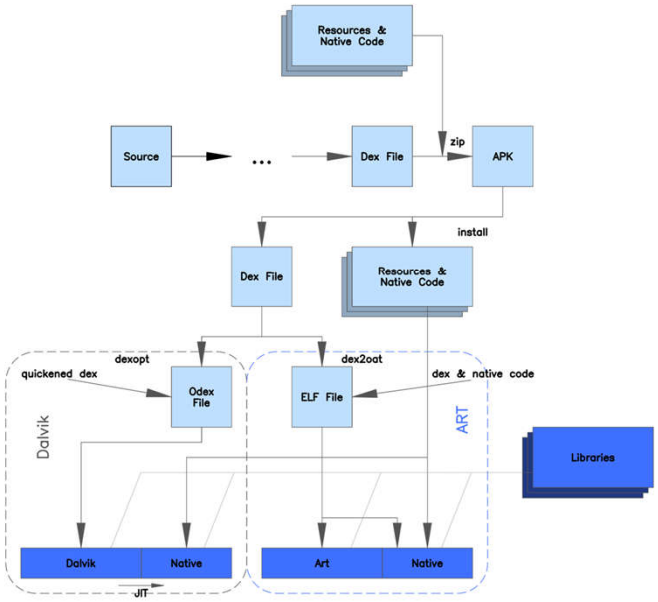
- les types *Activity* et *Services* dérivent de *Context*
- permet de lancer des intentions vers un autre composant
  – qui appartient à l'application ou non
- permet de récupérer les ressources de l'application
  – répertoire, base de données, ...

Point sur la structure, l'exécution et le déploiement d'une application

La configuration de l'application est décrite dans le fichier AndroidManifest.xml

- nom de l'application
- icônes
- compatibilité : SDK, écrans, présence d'un clavier, ...
- déclaration des modules applicatifs
- déclarations des filtres de messages
- déclaration des permissions : utilisation GPS, internet, etc.
- classe d'instrumentation de l'application

Modèle d’exécution ART



Modèle d’exécution ART

Dalvik utilise le compilateur JIT (just in time)
Ce compilateur va compiler les parties les plus utiliser
Dans de petit segment de code appelés « traces ».
Le compilateur ART lui va tout compiler en code natif.
Le résultat est une meilleur gestion de la mémoire (gar
utilisation efficacité énergétiques des applications etc.
La rétro compatibilité est garanti par le fait que ART c

La classe de base : Activity

C’est la classe de base
Celle qui gère ce que l’on peut appeler le Controller
Si l’on se place dans cadre du modèle MVC

La classe de base : Activity

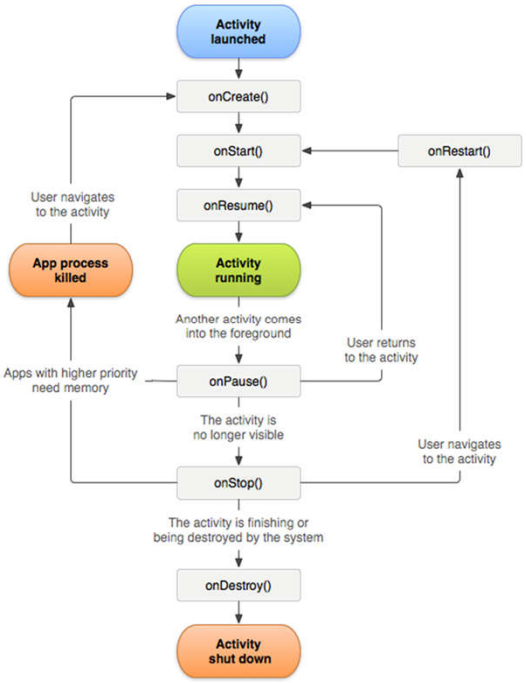
Une activité est un composant de l'application qui fournit un écran avec lequel les utilisateurs peuvent interagir et faire une action comme téléphoner, prendre une photo, envoyer un email, ou voir une carte.
Chaque activitée est associé à une « view » dans laquelle on peut dessiner une interface utilisateur.
La fenêtre remplit l'écran traditionnellement, mais elle peut être flottante et surnager au dessus d'autres fenêtres.

La classe de base : Activity

Une application est généralement constituée de nombreuses activitées qui sont faiblement couplées entre elles.
Typiquement une activitée dans une application est désignée comme l'activitée principale. L'activitée « root ».
La première lancé lors du lancement de l'application.

La classe de base : Activity

Activity :
Cycle de vie



Action

android_lifecycle_activity

Utilisation des LiveTemplates

- Récupération des templates
  - File/Import Settings
  - Settings.jar
- Visualisation/Edition
  - File/Settings
  - Editor/LiveTemplates
  - AndroidTristan

https://www.bignerdranch.com/blog/android-studio-live-templates/

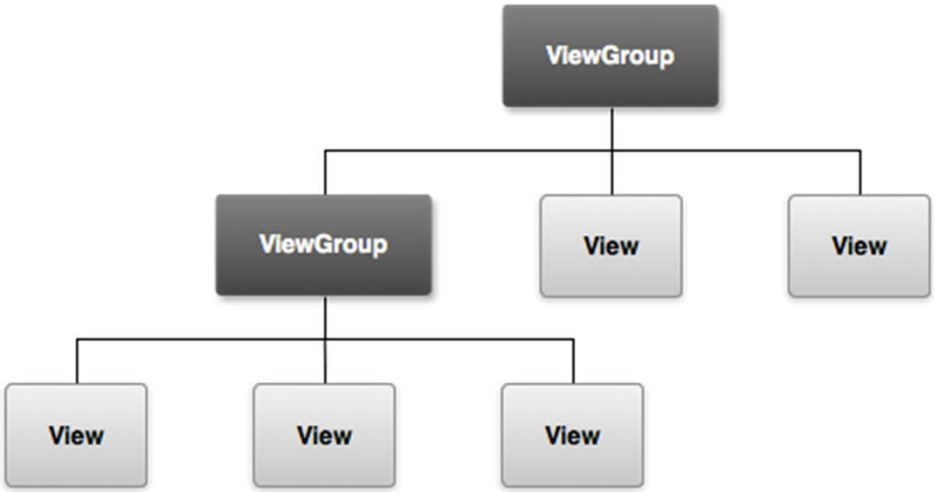
Méthodes disponibles :
https://www.jetbrains.com/help/idea/2016.2/creating-and-editing-template-variables.html

Outils

- DDMS
- Logcat (test sur le « bonjour le monde»)
- Android Lint
- ADB
- Déverminage (debogage)
- Téléphone physique
- Tests unitaires
- Monkey http://developer.android.com/tools/help/monkey.html
- MonkeyRunner
- Draw 9-patch
- Hierarchy Viewer (hierarchyviewer)
- ProGuard
- Robotium
- ...
- Live Templates
- Tools : https://blog.stylingandroid.com/tool-time-part-1-2/
- Screenshoots : https://github.com/Karumi/Shot
- http://developer.android.com/tools/help/index.html
- http://romannurik.github.io/AndroidAssetStudio/index.html

https://developer.android.com/studio/write/draw9patch.html

La GUI d'une application Android: View et ViewGroup



https://developer.android.com/guide/topics/ui/overview.html

La GUI d’une application Android: View et ViewGroup

Action

Widgets Simples

- http://developer.android.com/guide/topics/ui/controls.html
- TextView
- EditText
- Button
- CheckBox
- RadioButton et RadioGroup

Utilitaire simple

https://openclassrooms.com/courses/creez-des-applications-pour-android/les-widgets-les-plus-simples

Evénements

- View.OnClickListener
- View.OnLongClickListener

- Par héritage
- Classe anonyme
- Attribut
- XML

Par héritage

```
import android.view.View.OnTouchListener;
import android.view.View.OnClickListener;
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;

// Notre activité détectera les touchers et les clics sur
les vues qui se sont inscrites
public class Main extends Activity implements
View.OnTouchListener, View.OnClickListener {
  private Button b = null;

  @Override
  public void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);

    b = (Button) findViewById(R.id.boutton);
    b.setOnTouchListener(this);
    b.setOnClickListener(this);
  }

  @Override
  public boolean onTouch(View v, MotionEvent event) {
    /* Réagir au toucher */
    return true;
  }

  @Override
  public void onClick(View v) {
    /* Réagir au clic */
  }
}


public void onClick(View v) {
  // On récupère l'identifiant de la vue, et en fonction de cet
identifiant…
  switch(v.getId()) {

    // Si l'identifiant de la vue est celui du premier bouton
    case R.id.bouton1:
    /* Agir pour bouton 1 */
    break;

    // Si l'identifiant de la vue est celui du deuxième bouton

    case R.id.bouton2:
```

```
    /* Agir pour bouton 2 */
    break;

    /* etc. */
  }
}
```

---

classe anonyme

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class AnonymousExampleActivity extends Activity {
  // On cherchera à détecter les touchers et les clics sur ce
bouton
  private Button touchAndClick = null;
  // On voudra détecter uniquement les clics sur ce bouton
  private Button clickOnly = null;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    touchAndClick =
(Button)findViewById(R.id.touchAndClick);
    clickOnly = (Button)findViewById(R.id.clickOnly);

    touchAndClick.setOnLongClickListener(new
View.OnLongClickListener() {
      @Override
```

```
      public boolean onLongClick(View v) {
        // Réagir à un long clic
        return false;
      }
    });

    touchAndClick.setOnClickListener(new
View.OnClickListener() {
      @Override
      public void onClick(View v) {
        // Réagir au clic
      }
    });

    clickOnly.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        // Réagir au clic
      }
    });
  }
}
```

Par un attribut

```
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;

public class Main extends Activity {
  private OnClickListener clickListenerBoutons = new
View.OnClickListener() {
```

```
    @Override
    public void onClick(View v) {
      /* Réagir au clic pour les boutons 1 et 2*/
    }
  };

  private OnTouchListener touchListenerBouton1 = new
View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
      /* Réagir au toucher pour le bouton 1*/
      return onTouch(v, event);
    }
  };

  private OnTouchListener touchListenerBouton3 = new
View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
      /* Réagir au toucher pour le bouton 3*/
      return super.onTouch(v, event);
    }
  };

  Button b1 = null;
  Button b2 = null;
  Button b3 = null;

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

    setContentView(R.layout.main);

    b1 = (Button) findViewById(R.id.bouton1);
```

```
    b2 = (Button) findViewById(R.id.bouton2);
    b3 = (Button) findViewById(R.id.bouton3);

    b1.setOnTouchListener(touchListenerBouton1);
    b1.setOnClickListener(clickListenerBoutons);
    b2.setOnClickListener(clickListenerBoutons);
    b3.setOnTouchListener(touchListenerBouton3);
  }
}
```

En XML

```
android:onClick="launchRAZ"

public void launchRAZ(View view) {

}
```

Implémentation de l'application IMC

- Action boutons
- Récupération des valeurs texte
- Récupération valeur radio bouton
- Récupération valeur checkbox
- Ecoute champ texte
- Vérifications des valeurs entrées
- Formater le résultat

http://openclassrooms.com/courses/creez-des-applications-pour-android/les-widgets-les-plus-simples

Actions boutons
```
envoyer.setOnClickListener(envoyerListener);
raz.setOnClickListener(razListener);
```

Récupération des valeurs texte
```
// On récupère la taille
String t = taille.getText().toString();
```

Récupération valeur radio bouton
```
if(group.getCheckedRadioButtonId() == R.id.radio2)
```

Récupération valeur checkbox
```
mega.isChecked()
```

Afficher du texte:

tvResult.setText()

Ecoute champ texte
taille.addTextChangedListener(textWatcher);

```
private TextWatcher textWatcher = new TextWatcher() {

    @Override
    public void onTextChanged(CharSequence s, int start, int
before, int count) {
      result.setText(defaut);
    }

    @Override
    public void beforeTextChanged(CharSequence s, int start,
int count,
      int after) {

    }

    @Override
    public void afterTextChanged(Editable s) {

    }
  };
```

Formater les nombres
http://android.okhelp.cz/formatting-a-number-to-string-java-android-example-code/

```
NumberFormat numberFormat  = new DecimalFormat("##");
String str = numberFormat.format(-01234.567);         // -1235
System.out.print(str + "\n");

str = numberFormat.format(00);                // 0
```

```
System.out.print(str + "\n");

numberFormat = new DecimalFormat("##00");
str = numberFormat.format(0);                // 00
System.out.print(str + "\n");

numberFormat = new DecimalFormat(".00");
str = numberFormat.format(-.4567);           // -.46
System.out.print(str + "\n");

numberFormat = new DecimalFormat("0.000");
str = numberFormat.format(-.34567);          // -0.346
System.out.print(str + "\n");

numberFormat = new DecimalFormat("#.######");
str = numberFormat.format(-012.34567);       // -12.34567
System.out.print(str + "\n");

numberFormat = new DecimalFormat("#.000000");
str = numberFormat.format(-1234.567);        // -1234.567000
System.out.print(str + "\n");

numberFormat = new DecimalFormat("#,###,###");
str = numberFormat.format(-01234567.890);    // -1 234 568
System.out.print(str + "\n");

numberFormat = new DecimalFormat("'text'#");
str = numberFormat.format(+1234.567);        // text1235

System.out.print(str + "\n");

// Exponential notation
numberFormat = new DecimalFormat("00.00E0");
str = numberFormat.format(-012345.67);       // -12.35E2
```

```
System.out.print(str + "\n");

// set locale format
// FRANCE  locale
Locale locale = Locale.FRANCE;
str = NumberFormat.getNumberInstance(locale).format(-
123456.789);  // -123 456,789
System.out.print(str + "\n");
```

Visibilité

- XML :
  - android:visibility="gone"
  - android:visibility="invisible"
  - android:visibility="visible"
- Java
  - setVisibility(View.GONE);
  - setVisibility(View.INVISIBLE);
  - setVisibility(View.VISIBLE);

Bonnes pratiques

- Utiliser les ressources
  - Drawable
  - Values
    - strings.xml
      - Plurals, getQuantityString
      - Arguments
      - HTML
      - Tableaux
    - dimens.xml
      - px, in, pt, dp, sp
    - colors.xml
      - #RGB, #RRGGBB, #ARGB, #AARRGGBB
    - styles.xml

Strings
http://developer.android.com/guide/topics/resources/string-resource.html
http://www.tutos-android.com/la-gestion-strings-android-chaines-caracteres-strings-xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals name="numberOfSongsAvailable">
<!– zero, one, two, few, many, other -->
        <!--
            As a developer, you should always supply "one" and "other"
            strings. Your translators will know which strings are actually
            needed for their language. Always include %d in "one" because
            translators will need to use %d for languages
```

```
where "one"
            doesn't mean 1 (as explained above).
          -->
        <item quantity="one">%d song found.</item>
        <item quantity="other">%d songs found.</item>
    </plurals>
</resources>
```

```
int count = getNumberOfsongsAvailable();
Resources res = getResources();
String songsFound =
res.getQuantityString(R.plurals.numberOfSongsAvailable,
count, count);
```

Formater et mise en forme

Apostrophes et guillemets

```
<string name="good_example">"This'll work"</string>
<string name="good_example_2">This\'ll also work</string>
<string name="bad_example">This doesn't work</string>
<string name="bad_example_2">XML encodings don&amp;apos;t
work</string>
```

Formater :

```
<string name="welcome_messages">Hello, %1$s! You have
%2$d new messages.</string>
```

```
Resources res = getResources();
String text =
String.format(res.getString(R.string.welcome_messages),
username, mailCount);
```

Mise en forme HTML

```
<string name="welcome">Welcome to
<b>Android</b>!</string>
```

<b> for **bold** text.
<i> for *italic* text.
<u> for underline text.
…

Tableaux

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
    </string-array>
</resources>
```

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

DIMENS :
Faire des tests sur les preview en utilisant des PX, des DP et SP.

COLORS :
Test de différentes couleurs sur un layout de test

STYLES :
Regrouper différents paramètres sur un style (couleur bold, taille, alignement, ….)

Bonnes pratiques

- Hint
- Typer les champs texte
- Gérer le bouton OK/Send, …
- Gérer le focus

https://developer.android.com/training/keyboard-input/style.html

```
<EditText
    android:id="@+id/phone"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/phone_hint"
    android:inputType="phone" />

android:hint="@string/phone_hint"
android:inputType="phone"
android:imeOptions="actionSend"

EditText editText = (EditText)
findViewById(R.id.search);
editText.setOnEditorActionListener(new
OnEditorActionListener() {
```

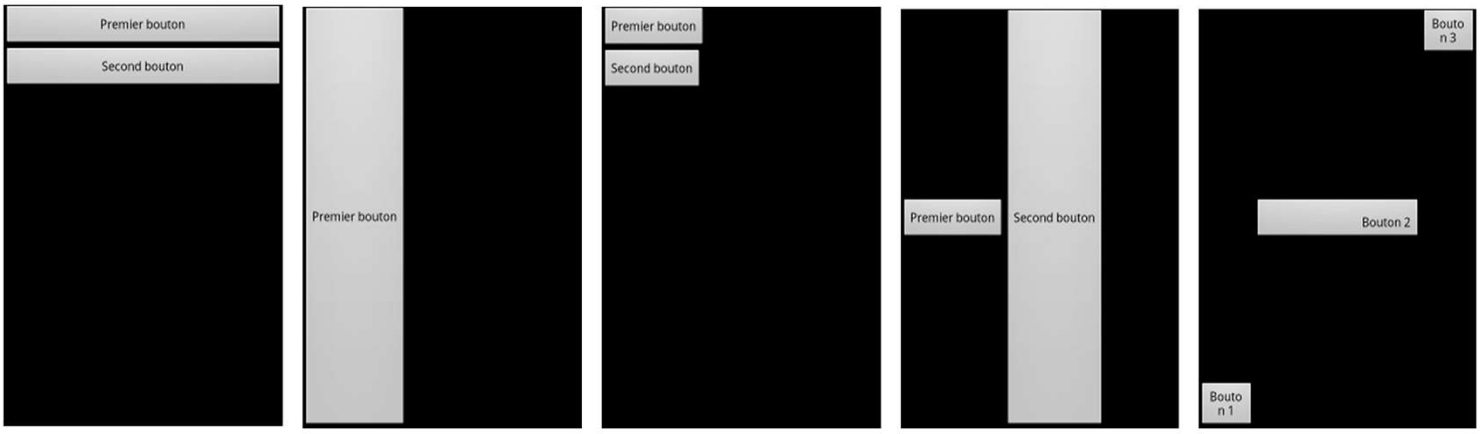
```
    @Override
    public boolean onEditorAction(TextView v, int actionId,
KeyEvent event) {
        boolean handled = false;
        if (actionId == EditorInfo.IME_ACTION_SEND) {
            sendMessage();
            handled = true;
        }
        return handled;
    }
});
```

android:nextFocusUp
android:nextFocusDown
android:nextFocusLeft
android:nextFocusRight

Différents Layouts

- XML VS Java
- Id
- Layouts
  - LinearLayout
  - RelativeLayout
  - FrameLayout
  - TableLayout
  - GridLayout
  - …

LinearLayout



Ex 1 :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/an
droid"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.ynov.test.MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Premier button" />
```

```
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Second button" />
</LinearLayout>
```

Ex 2 :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.ynov.test.MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:text="Premier button" />

</LinearLayout>
```

Ex 3 :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.ynov.test.MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Premier button" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Second button" />
</LinearLayout>
```

Ex 4 :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    tools:context="com.ynov.test.MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:text="Premier button" />
```

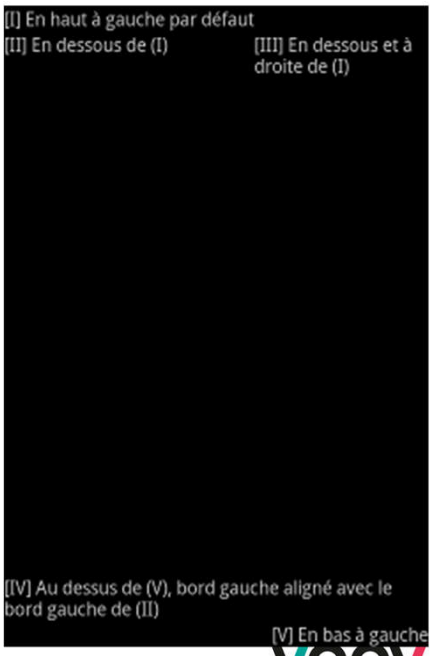
```
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:text="Second button" />
</LinearLayout>
```

Ex 5 :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    tools:context="com.ynov.test.MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="80dp"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:text="button 1" />

    <Button
        android:id="@+id/button2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_weight="1"
        android:gravity="right|bottom"
        android:text="button 2" />

    <Button
```

```
        android:id="@+id/button3"
        android:layout_width="80dp"
        android:layout_height="wrap_content"
        android:layout_gravity="top"
        android:text="button 3" />
</LinearLayout>
```

RelativeLayout



Ex 1 :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/an
droid"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.ynov.test.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:text="Centré horizontalement" />

    <TextView
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:text="Centré verticalement" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Centré dans le parent" />

</RelativeLayout>
```

Ex 2 :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.ynov.test.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:text="En haut" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="En bas" />
```

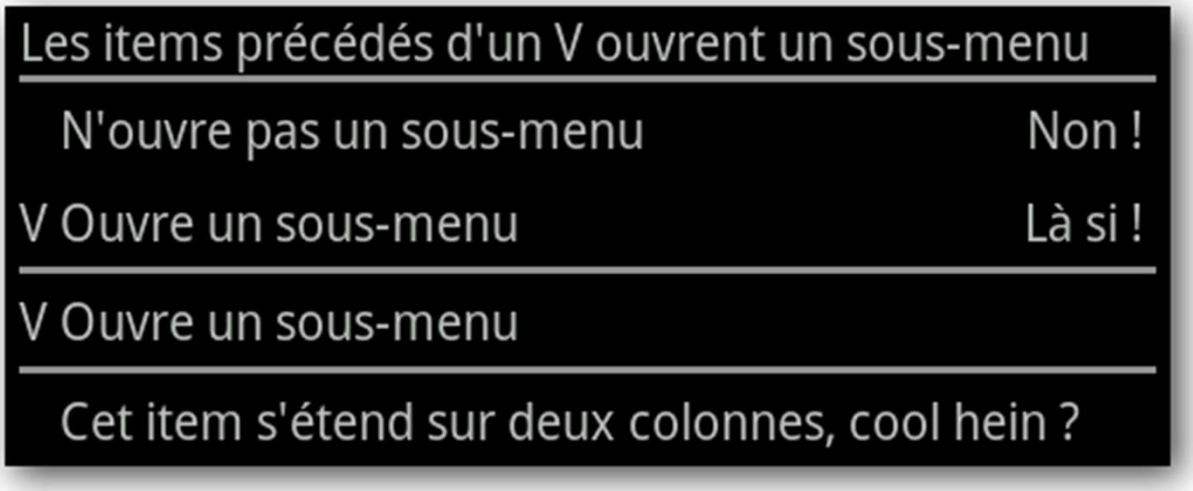
```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:text="A gauche" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:text="A droite" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Ces soirées là !" />
</RelativeLayout>
```

Ex 3 :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.ynov.test.MainActivity">

    <TextView
        android:id="@+id/ex_rel_3_textView_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:text="[I] En haut a gauche par défaut" />
```

```
    <TextView
        android:id="@+id/ex_rel_3_textView_2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/ex_rel_3_textView_1"
        android:text="[II] En dessous de [I]" />

    <TextView
        android:id="@+id/ex_rel_3_textView_3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/ex_rel_3_textView_1"
        android:layout_toRightOf="@id/ex_rel_3_textView_1"
        android:text="[III] En dessous et à droite de [I]" />

    <TextView
        android:id="@+id/ex_rel_3_textView_4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@id/ex_rel_3_textView_5"
        android:layout_alignLeft="@id/ex_rel_3_textView_2"
        android:text="[IV] au dessus de [V], bord aligné avec le
bord gauche de [II]" />

    <TextView
        android:id="@+id/ex_rel_3_textView_5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:text="[V] En bas à droite" />
</RelativeLayout>
```

TableLayout



ScrollView

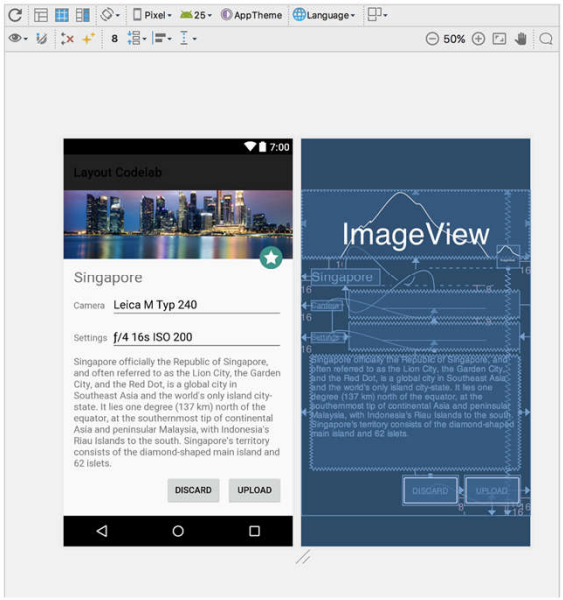
- Faire défiler notre interface

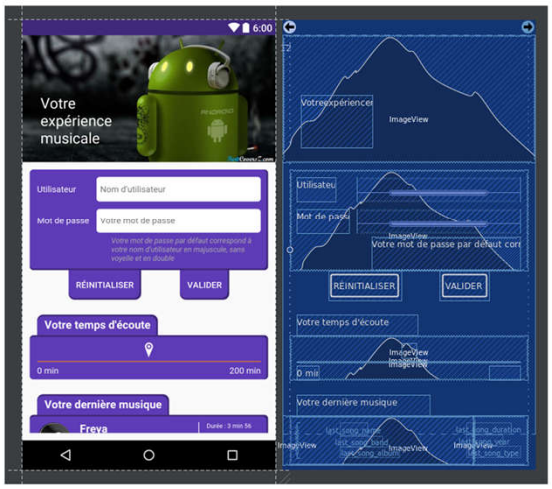
Constraint Layout

- Mai 2016
- Interfaces avancées
  - Alignements
  - Biais
  - Taille/Ratio
  - Guides

https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html
https://developer.android.com/training/constraint-layout/index.html

Constraint Layout





Mars-Avril-Mai 2019 Formation Android – Tristan SALAUN

ynov AIX-EN-PCE

https://codelabs.developers.google.com/codelabs/constraint-layout/index.html
http://blog.soat.fr/2017/01/transformez-vos-layouts-en-constraintlayout-partie-1/

Autres éléments graphiques

- Éléments graphiques
  - Liste de choix
  - Barre de progression
  - Notation (étoiles)
  - Date et heure
  - WebView

Utiliser les templates pour les spinners

WebView
Ajout de la permission dans le fichier Manifest :
<uses-permission
android:name="android.permission.INTERNET"
/>

// Usage le plus simple
webview.loadUrl("http://iut.univ-amu.fr/diplomes/dut-informatique");

Eviter que le navigateur externe soit ouvert :
webview.setWebViewClient(new WebViewClient());

```
// HTML directement
String summary = "<html><body>Votre score est de
<b>192</b> points.</body></html>";
webview.loadData(summary, "text/html", null);
```

La GUI d’une application Android: View et ViewGroup

Pour résumer:

Tous les éléments de l’interface utilisateur d’une application Android sont construits avec des View et des ViewGroup.
Une View sert à dessiner quelques chose sur l’écran avec laquelle l’utilisateur peut interagir
Un ViewGroup sert simplement à définir le l’organisation d’une interface.

La Classe R

R.java est une classe généré dynamiquement, créer durant le processus de build dynamiquement pour identifier tout les assets (des strings aux widgets android aux layouts) afin que les classes de l'application Android puissent les utilisés.
C'est évidement une caractéristiques d'une application Android sur les autres plateforme ce fichier de liaison n'existe pas nécessairement.

Démo

Mars-Avril-Mai 2019

Formation Android – Tristan SALAUN

ynov AIX-EN-PCE

La Classe R

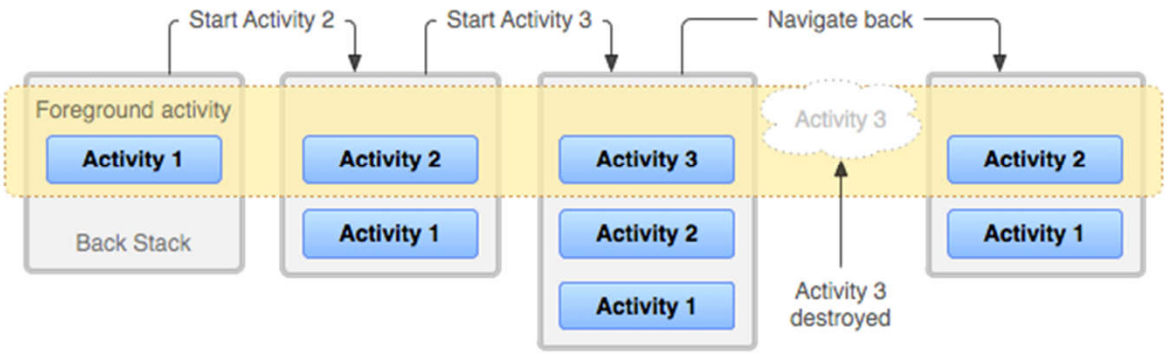
Pour résumer:
Nous avons créé une ressources string.
Nous avons regardé le fichier R.java et constaté qu'après compilation il s'était mis à jour avec une référence vers ce fichier.

Application dynamique sous Android

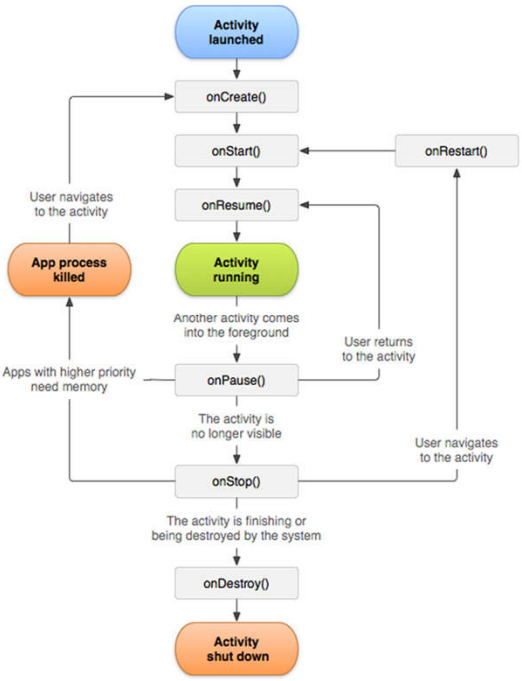
La Backstack des activités

Une application contient de multiples activités. Lorsqu'on en lance une, celle précédemment ouverte se stoppe (pause). C'est une pile de type « last in, first out » c'est à dire « dernier arrivé, premier sorti »

Application Dynamique sous Android



Le cycle de vie d'une activité



Les Intents

Les différents modules applicatifs ne sont pas directement instanciés par le développeur

- Un bus de messages permet au système de choisir
  - le composant à monter en mémoire
  - les messages sont de type Intent
  - les intentions décrivent quelle est l'opération qui devra être effectuée
    - plusieurs composants applicatifs peuvent répondre à un même Intent
      - l'utilisateur aura alors le choix

Les Intents

Un message est principalement constitué

- d'une action

– affichage, composition d'un numéro de téléphone, démarrage d'un module, ...

- d'une catégorie

– informations supplémentaires pour l'action à mener préférences utilisateur, affichage dans les applications, …

- d'autres données : type MIME, composant à invoquer, données supplémentaires, URI

Les Intents

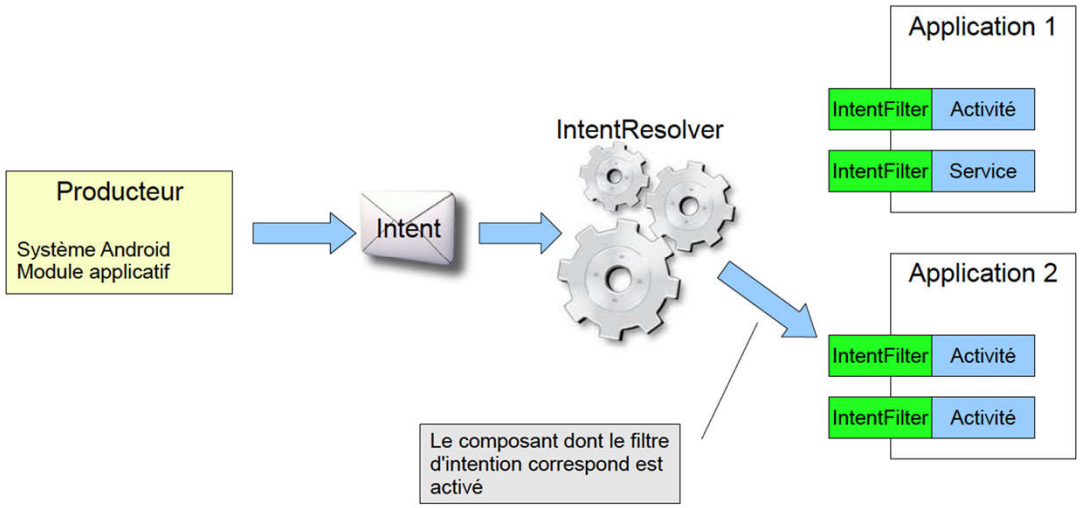
Un composant décrit une configuration de messages auxquels il est susceptible de répondre

- dans le fichier AndroidManifest.xml
- balise <intent-filter>

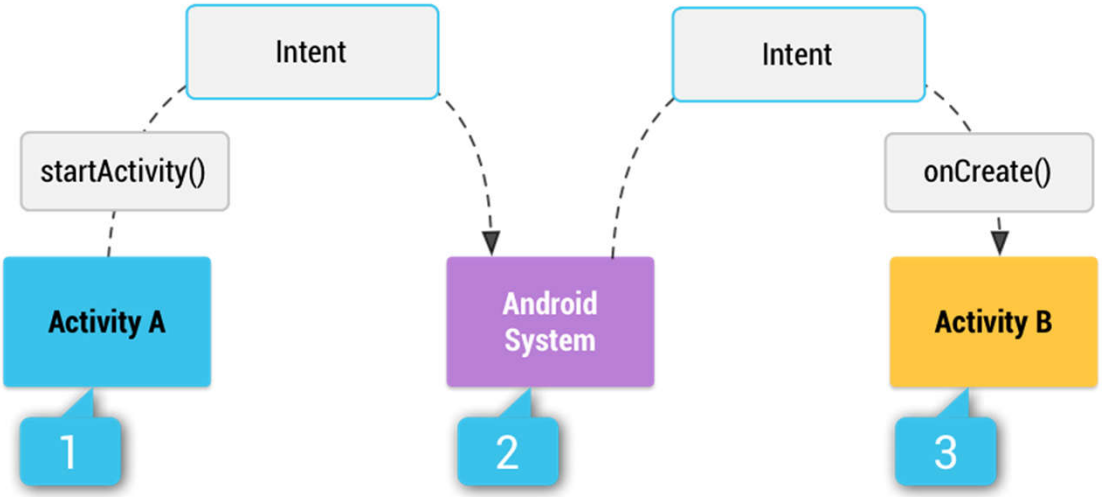
– exemple : point d'entrée d'un application

```
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Les Intents


Application 1
IntentFilter
Activité
IntentFilter
Service
IntentResolver
Producteur
Système Android
Module applicatif
Intent
Application 2
IntentFilter
Activité
IntentFilter
Activité
Le composant dont le filtre d'intention correspond est activé

Les Intents


Intent
Intent
startActivity()
onCreate()
Activity A
Android System
Activity B
1
2
3

Le bus des Intentions

Intention explicite :

```
// Executed in an Activity, so 'this' is the Context
// The fileUrl is a string URL, such as "http://www.example.com/image.png"
Intent downloadIntent = new Intent(this, DownloadService.class);
downloadIntent.setData(Uri.parse(fileUrl));
startService(downloadIntent);
```


https://developer.android.com/guide/components/intents-filters.html

Le bus des Intentions

Intention implicite :

```
Intent sendIntent = new Intent(Intent.ACTION_SEND);
...

// Always use string resources for UI text.
// This says something like "Share this photo with"
String title = getResources().getString(R.string.chooser_title);
// Create intent to show the chooser dialog
Intent chooser = Intent.createChooser(sendIntent, title);

// Verify the original intent will resolve to at least one activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
  startActivity(chooser);
}
```


Templates intents

Action

Mars-Avril-Mai 2019 Formation Android – Tristan SALAUN

ynov AIX-EN-PCE

Utiliser les macros fournies pour lancer différents intents

Action

ynov AIX-EN-PCE

android_intent_activity
android_onCreate

Passage d'information Intent Retour

```
Activity1
Intent intent=new Intent(MainActivity.this,SecondActivity.class);
startActivityForResult(intent, 2);// Activity is started with requestCode 2


   @Override
     protected void onActivityResult(int requestCode, int resultCode, Intent data)
     {
            super.onActivityResult(requestCode, resultCode, data);

if (resultCode == RESULT_OK) {

            // check if the request code is same as what is passed  here it is 2
             if(requestCode==2)
                {
                  String message=data.getStringExtra("MESSAGE");
                  textView1.setText(message);
                }
}
    }


Activity2
String message=editText1.getText().toString();
              Intent intent=new Intent();
              intent.putExtra("MESSAGE",message);
              setResult(RESULT_OK,intent);
              finish();//finishing activity
```


https://developer.android.com/training/basics/intents/result.html

Notifications utilisateur

Une notification est un message que l'on peut afficher à l'utilisateur à l'extérieur de UI classique dans la « notification area ».
Pour voir le détail de la notification l'utilisateur va glisser le « notification drawer »

Notifications utilisateur

Les notifications avertissent l'utilisateur

Android propose plusieurs types de notifications

- Le toast : notification rapide sur l'écran
- Le crouton => Snackbar
- la boite de dialogue : permet une interaction avec l'utilisateur
- la barre de notifications : permet une notification par des tâches de fond sans perturber l'utilisateur
- Les notifications sont ajoutées les unes aux autres

Toast

Message apparaissant durant quelques secondes
- ne possède aucun bouton, l'utilisateur ne peut que lire le message
- le message s'affiche en premier plan
- méthodes utiles de la classe Toast
- public static Toast makeText(Context ctx,T t, int duration)

– ctx est le contexte graphique
– t peut être de type int ou CharSequence, pour identifiant de ressource ou un message
- show()

– affiche le message
- d'autres méthodes permettent de changer le layout par défaut d'un toast

Toast
https://developer.android.com/guide/topics/ui/notifiers/toasts.html

Notifications utilisateur

Toast

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

Boîte de dialogue

Fenêtre modale qui possède des boutons pour interagir avec l'utilisateur

- Les boîtes de dialogues héritent de la classe Dialog
- Android fournit des boites de dialogues pour les besoins les plus courants
- AlertDialog : alerte
- DatePickerDialog : choix d'une date
- TimePickerDialog : choix d'une heure
- ProgressDialog : information sur un traitement en cours
- CharacterPickerDialog : choix d'un caractère accentué

AlertDialog

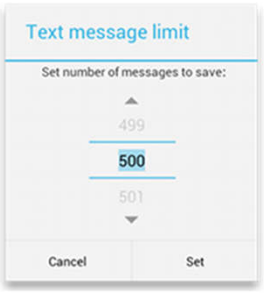
La classe interne Builder permet de simplifier la création de la boite de dialogue

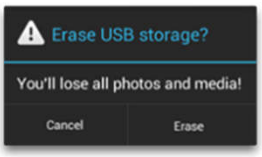
- De nombreuses méthode permettent d'ajouter une message, titre, liste d'items, etc.
- Les transparents suivants présentent des exemples de créations et utilisation de boites d'alerte
- alerte avec un message
- alerte avec une liste de choix
- L'affichage de la boite de dialogue est effectué avec sa méthode show()
- La boite de dialogue est fermée automatiquement
- cf. les méthodes dismiss() et cancel()

Notifications utilisateur

Dialog

```
public class FireMissilesDialogFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Use the Builder class for convenient dialog construction
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        builder.setMessage(R.string.dialog_fire_missiles)
        .setPositiveButton(R.string.fire, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                // FIRE ZE MISSILES!
            }
        })
        .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                // User cancelled the dialog
            }
        });
        // Create the AlertDialog object and return it
        return builder.create();
    }
}
```

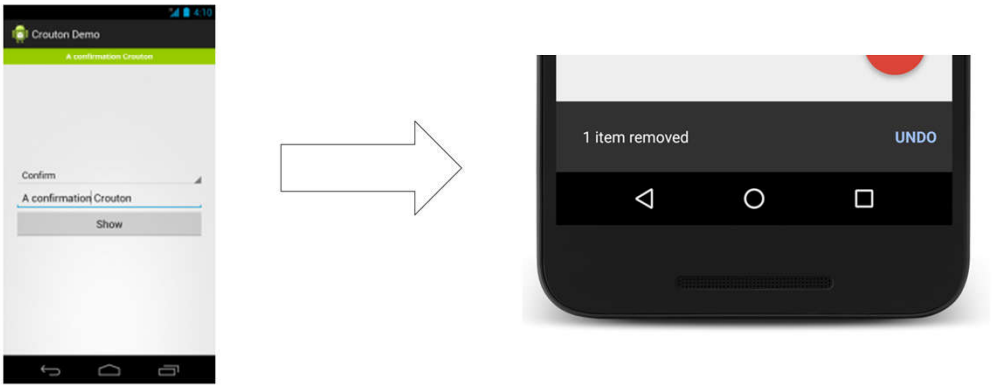




Notifications utilisateur

Crouton/Snackbar

```
Snackbar mySnackbar = Snackbar.make(findViewById(R.id.myCoordinatorLayout),
                                    R.string.email_archived, Snackbar.LENGTH_SHORT);
mySnackbar.setAction(R.string.undo_string, new MyUndoListener());
mySnackbar.show();
```

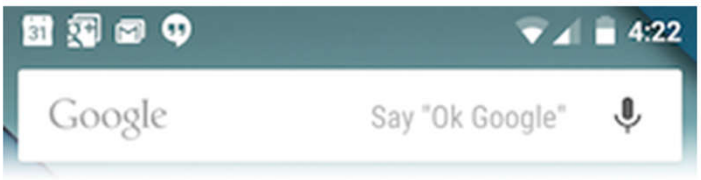


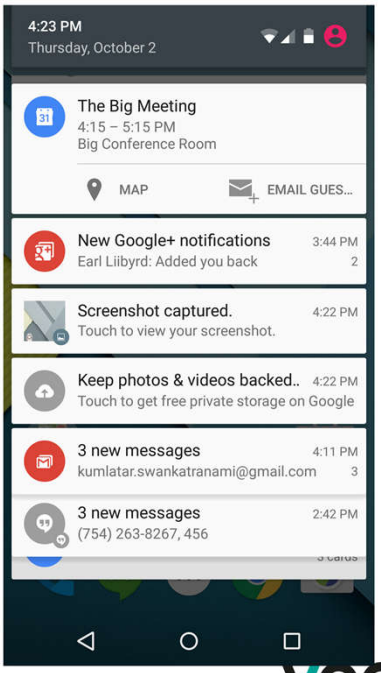
Crouton
https://www.grokkingandroid.com/useful-android-libraries-crouton/

Snackbar
https://developer.android.com/training/snackbar/action.html
https://www.androidhive.info/2015/09/android-material-design-snackbar-example/

Barre de Notification

La barre de notification permet l'affichage de
notifications qui sont empilées par le système, sans
perturber l'utilisateur
depuis Android 3.0 les notifications sont affichées par la
barre d'état (en bas de l'écran)





Barre de Notification

Création d'une simple notification

```
NotificationCompat.Builder mBuilder =
new NotificationCompat.Builder(this)
.setSmallIcon(R.drawable.notification_icon)
.setContentTitle("My notification")
.setContentText("Hello World!");
// Creates an explicit intent for an Activity in your app
Intent resultIntent = new Intent(this, ResultActivity.class);

// The stack builder object will contain an artificial back stack for the
// started Activity.
// This ensures that navigating backward from the Activity leads out of
// your application to the Home screen.
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
// Adds the back stack for the Intent (but not the Intent itself)
stackBuilder.addParentStack(ResultActivity.class);
// Adds the Intent that starts the Activity to the top of the stack
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent =
stackBuilder.getPendingIntent(
0,
PendingIntent.FLAG_UPDATE_CURRENT
);
mBuilder.setContentIntent(resultPendingIntent);
NotificationManager mNotificationManager =
(NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
// mId allows you to update the notification later on.
mNotificationManager.notify(mId, mBuilder.build());
```


Notification :
File, New, UI Component, Notification

Correction Version 8.0+

Utilisation des channels

```
private static void createNotificationChannel(Context context) {
    // Create the NotificationChannel, but only on API 26+ because
    // the NotificationChannel class is new and not in the support library
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = context.getString(R.string.channel_name);
        String description = context.getString(R.string.channel_description);
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
        channel.setDescription(description);
        // Register the channel with the system; you can't change the importance
        // or other notification behaviors after this
        NotificationManager notificationManager = context.getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    }
}
```

Ajouter l'appel au début de la méthode de notification.

```
public static void notify(final Context context,
                          final String exampleString, final int number) {
    final Resources res = context.getResources();

    createNotificationChannel(context);
```

Compléter l'appel du constructeur

```
final NotificationCompat.Builder builder = new NotificationCompat.Builder(context, CHANNEL_ID)
```

Définir la constante

```
private static final String CHANNEL_ID = "channel_name";
```


Notification :
File, New, UI Component, Notification

Gestion de l’interaction utilisateur

Les évènements de touché (down/move/up....)
Gérer les événements de touché : Listeners
Gestes prédéfinis (GestureDetector)

Par défault sur l’activity la méthode « onTouchEvent » est à overrider
On peut également écouter les events sur une view

Gestion de l’interaction utilisateur

Les évènements de touché (down/move/up....)
Gérer les événements de touché : Listeners
Gestes prédéfinis (GestureDetector)

Listactivity et Listview => RecyclerView

Si l’on souhaite afficher une liste d’éléments de nombreux mécanismes doivent être mis en place. Nous allons voir cela en détail.

https://developer.android.com/training/material/lists-cards.html
https://www.androidhive.info/2016/01/android-working-with-recycler-view/
http://www.vogella.com/tutorials/AndroidRecyclerView/article.html

Item Click
https://stackoverflow.com/questions/24471109/recyclerview-onclick

```
RecyclerView recyclerView =
findViewById(R.id.recycler);
recyclerView.addOnItemTouchListener(
    new RecyclerItemClickListener(context, recyclerView
,new RecyclerItemClickListener.OnItemClickListener() {
     @Override public void onItemClick(View view, int
position) {
```

```
        // do whatever
      }

      @Override public void onLongItemClick(View view, int
position) {
        // do whatever
      }
    })
);
```

RecyclerItemClickListener

```
import android.view.MotionEvent;
import android.view.View;


public class RecyclerItemClickListener implements
RecyclerView.OnItemTouchListener {
  private OnItemClickListener mListener;

  public interface OnItemClickListener {
    public void onItemClick(View view, int position);

    public void onLongItemClick(View view, int position);
  }

  GestureDetector mGestureDetector;

  public RecyclerItemClickListener(Context context, final
RecyclerView recyclerView, OnItemClickListener listener) {
    mListener = listener;
    mGestureDetector = new GestureDetector(context, new
GestureDetector.SimpleOnGestureListener() {
        @Override
        public boolean onSingleTapUp(MotionEvent e) {
```

```
            return true;
        }

        @Override
        public void onLongPress(MotionEvent e) {
            View child =
recyclerView.findChildViewUnder(e.getX(), e.getY());
            if (child != null && mListener != null) {
                mListener.onLongItemClick(child,
recyclerView.getChildAdapterPosition(child));
            }
        }
    });
}

  @Override public boolean
onInterceptTouchEvent(RecyclerView view, MotionEvent e) {
    View childView = view.findChildViewUnder(e.getX(),
e.getY());
    if (childView != null && mListener != null &&
mGestureDetector.onTouchEvent(e)) {
      mListener.onItemClick(childView,
view.getChildAdapterPosition(childView));
      return true;
    }
    return false;
  }

  @Override public void onTouchEvent(RecyclerView view,
MotionEvent motionEvent) { }

  @Override
  public void onRequestDisallowInterceptTouchEvent
(boolean disallowIntercept){}
}
```

ET / OU
https://antonioleiva.com/recyclerview-listener/

Méthodologie

File/New/New project …
…
Basic Activity
Cocher « Use Fragments »
File/New/Fragment/Fragment (List)
Remplacer dans content_main.xml le fragment utilisé avec notre nouveau fragment.
Implémenter dans MainActivity l'interface :
ItemFragment.OnListFragmentInteractionListener

RecyclerView

Composant graphique permettant de représenter des listes d'items

- composant important
- permet à l'utilisateur de choisir un item
- est muni d'un ascenseur

RecyclerView

Une vue RecyclerView est constituée de lignes

- prend en charge l'affichage des données d'un élément de la liste
- Les données à afficher sont de provenances diverses
- La liste utilise un RecyclerView.Adapter

– lie la vue aux données

- Android ne fournit pas plusieurs Adapters comme pour les ListView qui fournissait :

– ArrayAdapter<T> pour les données de type tableau
– CursorAdapter pour les données stockées en base
– SimpleCursorAdapter pour les données de type String ou image stockées en base
– etc.

Listactivity et Listview

ListView peut être géré par une activité

- ListActivity simplifie la gestion de l'activité en fournissant des méthodes support
- récupère une ListView depuis les ressources layout
  - doit être identifiée par @android:id/list
- La liaison d'une liste avec les données à afficher est effectué par un adaptateur de type Adapter
  - ListAdapter, SimpleAdapter et SimpleCursorAdapter
  - comme le Spinner

Listactivity et Listview

Chaque adaptateur possède son constructeur propre

- La classe ListView possède une méthode setAdapter(...) qui lie la vue avec les données
- si l'activité hérite de ListActivity, la méthode est dans l'activité

– en cas d'utilisation des layout par défaut l'identifiant de la ListView doit être android:id/list

Listactivity et Listview

Chaque adaptateur possède son constructeur propre

- La classe ListView possède une méthode setAdapter(...) qui lie la vue avec les données
- si l'activité hérite de ListActivity, la méthode est dans l'activité

– en cas d'utilisation des layout par défaut l'identifiant de la ListView doit être android:id/list

```
<ListView
android:id="@+android:id/list"
android:layout_width="match_parent"
android:layout_height="wrap_content" >
</ListView>
```

Listactivity et Listview

Exemple avec ArrayAdapter<T>

- Android fournit des gestionnaire de positionnement par défaut (l'identifiant de la liste doit être android:id/list)
  – android.R.layout.simple_list_item_1 qui comprend un objet unique de type TextView
  – android.R.layout.simple_list_item_2 qui comprend deux objets de type TextView, le second en dessous du premier dans une taille plus petite

```
public class ListViewActivity01 extends ListActivity {
    private static final String[] items = {"item 1","item 2","item 3","item 4"};
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,items);
        setListAdapter(adapter);
    }
}
```

ListActivity et Listview

Utilisation du layout par défaut
android.R.simple_list_item_2
● ce layout par défaut utilise un adaptateur qui relie une Map<String,?> avec deux TextView, appelés
text1 et text2
● il faut utiliser un adaptateur adéquat
– ArrayAdapter ne possède qu'un seul champ de type texte
– nous utilisons un SimpleAdapter qui est construit avec
● le contexte
● la liste de type List<Map<String,?>> ou la clé correspond respectivement à la ligne 1 puis 2
● le layout simple_list_item_2
● les clés des deux lignes sous forme de String[]
● les deux identifiants de TextView sous forme de int[]

Listactivity et Listview

Exemple d'utilisation du layout par défaut android.R.simple_list_item_2

```
public class ListView02Activity extends ListActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        List<HashMap<String, String>> liste = new ArrayList<HashMap<String,String>>();
        String[] from = {"line 1","line 2"};
        int[] to = {android.R.id.text1,android.R.id.text2};
        HashMap<String, String> map = new HashMap<String, String>();
        map.put("line 1","Item 1");
        map.put("line 2", "Sub item 1");
        liste.add(map);
        map.put("line 1","Item 2");
        map.put("line 2", "Sub item 2");
        liste.add(map);
        SimpleAdapter adapter = new SimpleAdapter(this,liste,android.R.layout.simple_list_item_2,from, to);
        setListAdapter(adapter);
    }
}
```

Action

Gestion des différents devices

- Rotation
  - layout-land
  - Bloquer
  - Gérer manuellement
    - configChanges= "orientation "
    - onConfigurationChanged()
    - onSaveInstanceState/ onRestoreInstanceState

Des interfaces adaptables et évolutives : les fragments

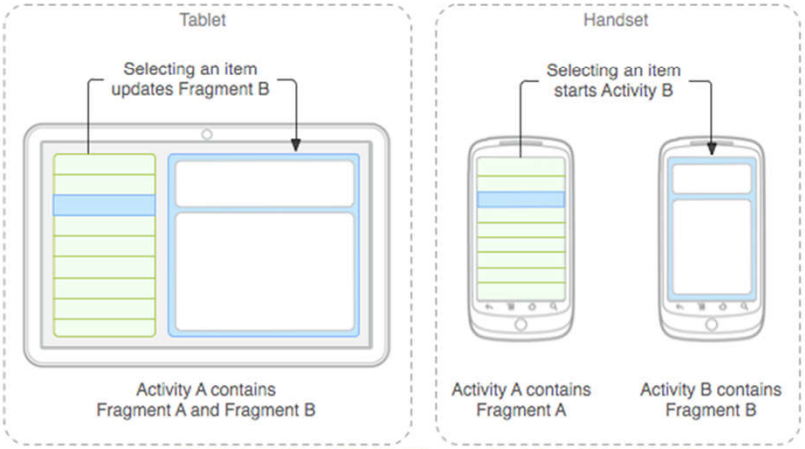
Existent depuis Android 3.0

- API niveau 11
- Un fragments représente une portion d'activité
- portion d'une IHM
- portion d'un comportement
- Permet la réutilisation d'un fragment dans des activités
- ajout statique ou dynamique
- permet de mieux gérer les écrans de tailles différentes
- Le fragment est embarqué dans une activité et suit le cycle de vie de l'activité

Des interfaces adaptables et évolutives : les fragments

Une application peut utiliser

- un fragment pour afficher une liste
- un fragment pour afficher les détails des éléments d'une liste

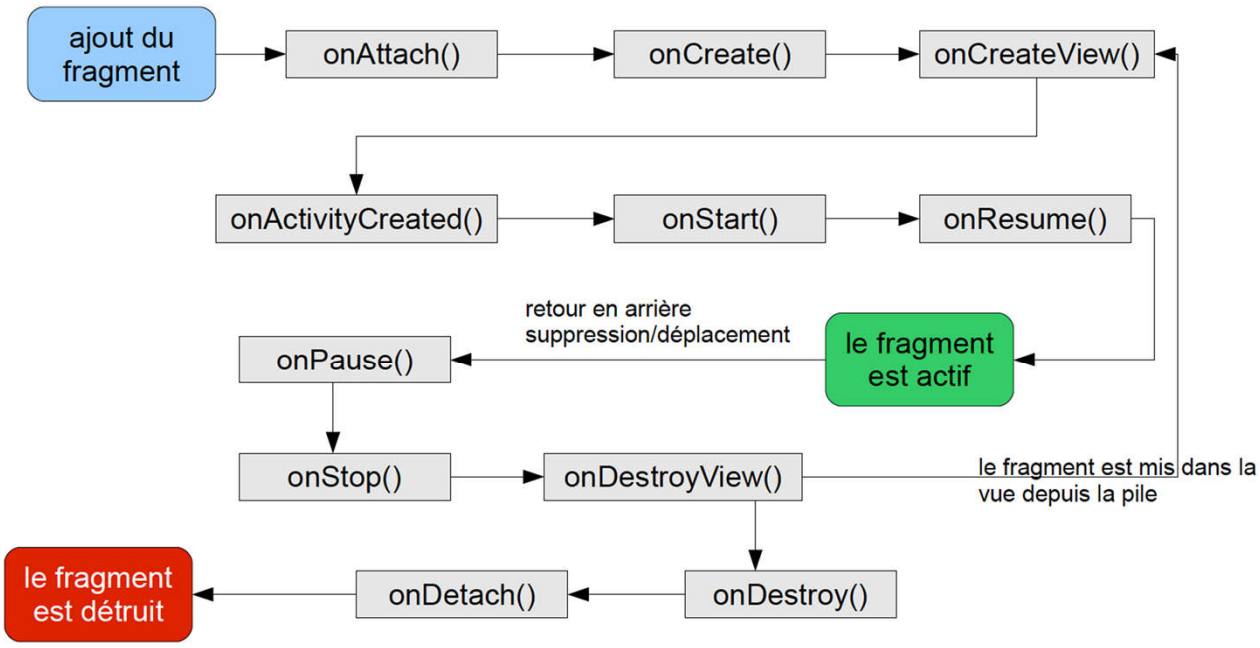


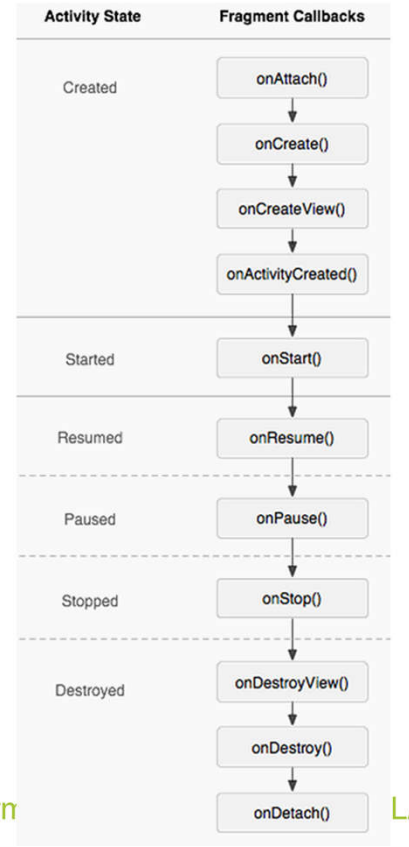
Les fragments

- Il est important
- de ne jamais manipuler un fragment directement depuis un autre
- de coder le comportement du fragment dans le fragment et non pas dans l'activité
- Pour éviter les appels directs
- définir une interface callback dans chaque fragment
- implémenter l'interface au niveau de l'activité

Les fragments

- Le template File/New/Activity/Basic Activity, cocher « Use a Fragment » montre comment ajouter un fragment
- Le fragment est constitué
  - d'un fichier xml décrivant l'IHM du fragment
  - d'une classe de gestion du fragment
    - la vue est créée dans la méthode onCreateView()
  - L'ajout du fragment est ici statique
  - effectué dans le fichier xml de l'IHM de l'activité principale


ajout du fragment
onAttach()
onCreate()
onCreateView()
onActivityCreated()
onStart()
onResume()
retour en arrière
suppression/déplacement
le fragment est actif
onPause()
onStop()
onDestroyView()
le fragment est mis dans la vue depuis la pile
le fragment est détruit
onDetach()
onDestroy()


Activity State
Fragment Callbacks
Created
onAttach()
onCreate()
onCreateView()
onActivityCreated()
Started
onStart()
Resumed
onResume()
Paused
onPause()
Stopped
onStop()
Destroyed
onDestroyView()
onDestroy()
onDetach()

Des interfaces adaptables et évolutives : les fragments

Démo

File New/Activity => Basic + Fragment coché
File New/Activity => Tabbed

File New/Fragment => Fragment (Blank)

File New/Activity => Master/Detail Flow

Pourquoi on utilise newInstance : https://stackoverflow.com/questions/9245408/best-practice-for-instantiating-a-new-android-fragment

Persistance des données

La persistance des données peut être effectuée

- par des fichiers de préférence
  – sauvegarde des types primitifs dans des fichiers XML
  – framework fournit par Android
- par la lecture/écriture directe sur le système de fichiers
  – sauvegarde d'images, ...
  – Android présente une API pour la gestion des fichiers temporaires, copie du stockage interne vers la carte SD, etc.
- par persistance en base de données
  – base de données SQLite

Préférences utilisateur

Android fournit un framework pour gérer les préférences utilisateur

- les préférences sont de type primitif et String
- sauvegarde sous forme de clé / valeur
- les fichiers de préférences sont au format XML
- Fichier de préférences
- par défaut un fichier de préférence est créé par application

– nom complètement qualifié de l'application

- SharedPreference permet la gestion du fichier

Préférences utilisateur

- Utiliser le fichier des préférences
- récupérer une instance de SharedPreferences via la méthode getPreferences(...) de l'activité

```
SharedPreferences prefs = getPreferences(MODE_PRIVATE);
```

– modes d'ouverture
- MODE_PRIVATE : uniquement l'application
- MODE_WORLD_READABLE : lecture par les autres applications
- MODE_WORLD_WRITEABLE : écriture par les autres application
- Pour utiliser un autre fichier que celui par défaut, il faut utiliser la méthode getSharedPreferences(...) qui prend comme

– premier paramètre : le nom du fichier
– second paramètre : le mode d'ouverture

Préférences utilisateur

Écriture des préférences

- utiliser la classe SharedPreferences.Editor
  – un éditeur est récupéré par la méthode edit()
- puis utiliser une série de méthode putXxx(String key, T value) pour mettre à jour les paires clé/valeur
  – où Xxx vaut Boolean, String, Int, Long, Float, …
  – et T est de type Boolean, String, Int, Long, Float, …
- et écrire les données par un commit()
  – sinon rien ne sera sauvegardé

Préférences utilisateur

- Exemple d'écriture de préférences
- la paire clé/valeur est lue dans un formulaire
- l'écriture est déclenchée par un bouton

```
public void enregistrerPreference(View v){
   EditText key = (EditText) findViewById(R.id.key);
   EditText value = (EditText) findViewById(R.id.value);
   SharedPreferences prefs = getPreferences(MODE_PRIVATE);
   Editor editor = prefs.edit();
   editor.putString(key.getText().toString(),value.getText().toString());
   editor.commit();
}
```

Préférences utilisateur

Lecture des préférences

- la lecture est effectuée sur un objet de type SharedPreference
- une série de méthodes permet de lire les valeurs :

public T getXxx(String key, T defValue)

– T type retourné par la méthode
– Xxx vaut Boolean, String, Int, Long, Float, …
– defValue est la valeur par défaut

- lecture de toutes les paires clé/valeur par
  – public Map<String ?> getAll()
- vérification de l'existence d'une entrée
  – public boolean contains(String key)

Préférences utilisateur

- Exemple de lecture de toutes les préférences
- déclenchement par un bouton
- écriture dans les logs

```
public void lirePreferences(View v){
  SharedPreferences prefs = getPreferences(MODE_PRIVATE);
  Map<String, ?> valeurs = prefs.getAll();
  for(String k : valeurs.keySet())
  Log.i("PreferencesActivity", k+" : "+prefs.getString(k, ""));
}
```

Préférences utilisateur

- Partage de préférences
- côté fournisseur de préférences

```
public void enregistrerPreference(View v) {
    EditText key = (EditText) findViewById(R.id.key);
    EditText value = (EditText) findViewById(R.id.value);
    SharedPreferences prefs = getSharedPreferences("CustomPref", MODE_WORLD_READABLE);
    Editor editor = prefs.edit();
    editor.putString(key.getText().toString(), value.getText().toString());
    editor.commit();
}
```

- côté consommateur de préférences

```
myContext = createPackageContext("com.codeattitude.android.preferences", 0);
    SharedPreferences sharedPref = myContext.getSharedPreferences("CustomPref",
    Context.MODE_WORLD_READABLE);
    for (Entry<String, ?> pair : sharedPref.getAll().entrySet()) {
    addPreference(pair.getKey(), pair.getValue());
}
```

Système de fichiers

- Les fichiers permettent de sauvegarder des données plus complexes que des types de base
  - son, image, …
- Les fichiers peuvent être créés sur le stockage interne ou externe (carte SD)
  - par défaut sur le stockage interne
  - l'accès au fichier interne est restreint à l'application
    - l'utilisateur et les applications ne peuvent pas y accéder
- Prise en charge de la gestion des fichiers temporaires

Système de fichiers – stockage interne

- Écriture dans le fichier
- Ouverture du flux fichier

– méthode de la classe Context :
– public FileOutputStream onpenFileOutput(String name, int mode)

- Écriture dans le flux retourné par la méthode

– méthodes write(...)

- Fermeture du flux

– méthode close()

- L'ensemble de ces méthodes est susceptible de lever des exceptions

Système de fichiers – stockage interne

- Lecture depuis le fichier
- Ouverture du flux fichier

– méthode de la classe Context :
– public FileInputStream onpenFileInput(String name)

- Lecture depuis le flux retourné par la méthode

– méthodes read(...)

- Fermeture du flux

– méthode close()

- L'ensemble de ces méthodes est successible de lever des exceptions
- Suppression d'un fichier
- méthode de Context

– public boolean deleteFile(String name)

Système de fichiers – stockage interne

- Exemple de lecture - écriture

```
public void ecrireFichier(View v){
  byte[] datas = {1,2,3,4,5,6,7,8,9,10};
  try {
    FileOutputStream out = this.openFileOutput("fichierInterne.dat", MODE_APPEND);
    out.write(datas);
    out.close();
  } catch (Exception e) {
    Log.e("FichierInterneActivity","Erreur sur fichier",e);
  }
}
public void lireFichier(View v){
  try {
    FileInputStream in = this.openFileInput("fichierInterne.dat");
    int b;
    do{
      b=in.read();
      Log.i("FichierInterneActivity","Octet lu : "+b);
    } while(b!=-1);
    in.close();
  } catch (Exception e) {
    Log.e("FichierInterneActivity","Erreur sur fichier",e);
  }
}
```

Système de fichiers – stockage sur SD

- Les fichiers stockés sur la carte SD sont publics
- l'utilisateur et les autres applications peuvent y accéder
- le stockage externe peut aussi être interne à l'appareil
  – carte SD intégrée
- le stockage externe peut être un support amovible
  – l'utilisateur peut retirer à tout moment la carte
  – l'application doit tenir compte de l'absence de support
- Il faut interroger l'environnement pour connaître la disponibilité de la carte SD
- méthode de la classe Environment
  - public static String getExternalStorageState()
    – renvoie l'état sous forme de String

Système de fichiers – stockage sur SD

- Constantes de la classe Environment représentant l'état du support

Etat	Description
MEDIA_BAD_REMOVAL	média retiré sans avoir été correctement démonté
MEDIA_CHECKING	média en cours de vérification
MEDIA_MOUNTED	média correctement monté en lecture/écriture
MEDIA_MOUNTED_READ_ONLY	média correctement monté en lecture seule
MEDIA_NOFS	média présent, mais système de fichiers non supporté
MEDIA_REMOVED	média absent
MEDIA_SHARED	média présent, non monté et partagé en tant périphérique USB
MEDIA_UNMOUNTABLE	média présent, mais ne peut pas être monté
MEDIA_UNMOUNTED	média présent mais non monté

Système de fichiers – stockage sur SD

- Un répertoire est affecté à chaque application
- les fichiers de données sont répartis dans des sous-répertoires : Music, Alarms, ….

– correspond à des constantes de la classe Environment

Répertoire	Contante	Contenu
Music/	DIRECTORY_MUSIC	fichiers de musique
Podcasts/	DIRECTORY_PODCASTS	podcasts
Ringtones/	DIRECTORY_RINGTONES	sonneries
Alarms/	DIRECTORY_ALARMS	sons des alarmes
Notifications/	DIRECTORY_NOTIFICATIONS	sons des notifications
Pictures/	DIRECTORY_PICTURES	photos
Movies/	DIRECTORY_MOVIES	films
Downloads/	DIRECTORY_DOWNLOADS	autres types de fichiers

Système de fichiers – stockage sur SD

- La méthode getExternalFilesDir retourne un objet de type File représentant le sous-répertoire de données
- Toutes les opérations d'écriture sur la carte SD nécessite une permission d'écriture
- dans le fichier *AndroidManifest.xml*

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

Système de fichiers – stockage sur SD

- Les fichiers communs à plusieurs applications sont mis dans une arborescence spécifique
- les fichiers créés dans cette arborescences ne sont pas supprimés même si l'application qui les a créés est désinstallée
- même organisation d'arborescence en fonction du type de données
- objet de type File représentant le sous-répertoire de données est retourné par la méthode : getExternalStoragePublicDirectory

Système de fichiers – stockage sur SD

- Exemple d'écriture sur SD

```
public void ecrireFichierExterne(View v){
  byte[] datas = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
  String state = Environment.getExternalStorageState();
  if(!state.equals(Environment.MEDIA_MOUNTED)){
    Log.e("FichierExterneActivity", "La carte SD n'est pas montée en écriture");
    Toast.makeText(this, "La carte SD n'est pas montée en écriture", Toast.LENGTH_SHORT).show();
    return;
  }
  try {
    File dir = this.getExternalFilesDir(Environment.DIRECTORY_DOWNLOADS);
    File file = new File(dir,"test.data");
    file.createNewFile();
    FileOutputStream fout = new FileOutputStream(file);
    fout.write(datas);
    fout.close();
    Toast.makeText(this, "Ecriture effectuée", Toast.LENGTH_SHORT).show();
  } catch (Exception e) {
    Log.e("FichierInterneActivity", "Erreur sur fichier", e);
  }
}
```

Système de fichiers – stockage sur SD

- Exemple de lecture depuis la carte SD

```
public void lireFichierExterne(View v){
  String state = Environment.getExternalStorageState();
  if(!(state.equals(Environment.MEDIA_MOUNTED_READ_ONLY) || state.equals(Environment.MEDIA_MOUNTED))){
    Log.e("FichierExterneActivity", "La carte SD n'est pas montée");
    Toast.makeText(this, "LLa carte SD n'est pas montée", Toast.LENGTH_SHORT).show();
    return;
  }
  try{
    File dir = this.getExternalFilesDir(Environment.DIRECTORY_DOWNLOADS);
    File file = new File(dir,"test.data");
    FileInputStream fin = new FileInputStream(file);
    int b;
    do {
      b = fin.read();
      Log.i("FichierExterneActivity", "Octet lu : " + b);
    } while (b != -1);
    fin.close();
    Toast.makeText(this, "Lecture effectuée", Toast.LENGTH_SHORT).show();
  }catch(Exception e){
    Log.e("FichierInterneActivity", "Erreur sur fichier", e);
  }
}
```

Système de fichiers – fichier temporaire

- Un emplacement spécifique à chaque application est réservé pour les fichiers temporaires
- en interne pour des fichiers de petite taille
  – public File getCacheDir() de la classe Context
- sur le support SD pour des fichiers de grande taille
  – public File getExternalCacheDir() de la classe Context
- l'application est chargée de supprimer le fichier temporaire
  – doit être supprimé dès que l'application n'en a plus besoin
- les fichiers temporaires sont supprimés si l'application est désinstallée

SQLite

- Une application peut stocker des données dans une ou plusieurs bases de données SQLite
- site officiel : http://www.sqlite.org/
- les bases de données sont privées à l'application qui les a créées
- passer par un fournisseur de contenu pour les rendre publiques
- Android ne dispose pas de framework ORM (Object Relational Mapping)
- il faut programmer l'exécution des requêtes SQL
- L'application doit tenir compte
- de la création de la base
- de sa mise à jour

ENI : Chapitre 11 :
Ch11_BaseDeDonneesExemple

SQLite

- Classes de stockage
  - INTEGER : nombres entiers, signés ou non
    - utilisé pour les clés primaire
      - PRIMARY KEY AUTOINCREMENT
  - REAL : nombres réels
  - TEXT : données textuelles
  - BLOB : stockage de données sous leur forme binaire
  - NULL : si la donnée est nulle

SQLite

- SQlite et Android
  - les données brutes (images, fichiers, …) ne sont généralement pas enregistrées dans les tables (BLOB)
    - les noms de fichiers, chemin sont enregistrés dans la base
  - il est préférable d'ajouter aux tables un index autoincrémenté
    - nommé _id
    - la table pourra ainsi être utilisée par un ContentProvider

SQLite

- Afin de gérer le cycle de vie de la base, Android fournit une classe abstraite SQLiteOpenHelper
  - le développeur doit créer une classe qui la spécialise
  - il faut redéfinir des méthodes callback du cycle de vie
    – une méthode de création
    public void SQLiteDatabase onCreate(SQLiteDatabase db)
    où db est la base de donnée manipulée
    – une méthode de mise à jour
    public void SQLiteDatabase onUpdate(SQLiteDatabase db, int odlVersion, int newVersion)
    où db est la base de donnée manipulée
    oldVersion est la version précédente de la base
    newVersion la nouvelle version de la base

SQLite

- Exemple de code pour le helper

```
public class ApplicationHelper extends SQLiteOpenHelper {
   private static final int version = 1;

   private static final String bdName = « todoBD";

   public ApplicationHelper(Context context) {
      super(context, bdName, null, version);
   }

   @Override
   public void onCreate(SQLiteDatabase db) {
      String sql = "CREATE TABLE todo ( kp INTEGER PRIMARY KEY AUTOINCREMENT, item VARCHAR(100);";
      db.execSQL(sql);
   }
   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
   // TODO Auto-generated method stub
   }
}
```

SQLite

- La classe SQLiteOpenHelper fournit la méthode
  - public synchronized SQLiteDatabase getWritableDatabase()
  - permet de gérer la base via la classe SQLiteDatabase
  - lors du premier appel de cette méthode, la base de données est réellement créée
    - les méthodes suivantes seront successivement invoquées
      - onCreate
      - onUpdate
      - onOpen
  - pour avoir une base de données en lecture seule il faut appeler getReadableDatabase
    - getWriteableDatabase peut aussi retourner une base en lecture seule s'il y un problème (disque plein par exemple)

SQLite

- La classe SQLiteDatabase possède les méthodes d'interrogation et gestion de la base
  - les plus utiles
  - public void execSql(String sql)
    - permet l'exécution de requêtes ne renvoyant pas de résultat
  - sélection d'enregistrements : méthodes query(...)
    - renvoie une instance de type Cursor
  - insertion d'enregistrements : méthodes insert(...)
    - renvoie le nombre d'enregistrements insérés
  - mise à jour d'enregistrements : méthodes update(...)
    - renvoie le nombre d'enregistrements mis à jour
  - suppression d'enregistrements : méthodes delete(...)
    - renvoie le nombre d'enregistrements supprimés

CRUD :

Create : créer
Read : lire
Update : mettre à jour
Delete : supprimer

SQLite

- Interrogation de la base
  - utilisation de la classe SQLiteDatabase
  - les requêtes retournent un Cursor
  - la méthode query accepte les paramètres suivants
    - distinct : booléen facultatif indiquant que le résultat doit conten valeurs uniques
    - table : nom de la table
    - projection : tableau de String contenant les colonnes
    - selection : clause WHERE, peut inclure des paramètres (?)
    - selectionArgs : tableau des paramètre de la clause WHERE
    - groupBy : clause GROUP BY
    - having : clause HAVING
    - orderBy : clause ORDER BY
    - limit : clause LIMIT

SQLite

- Support des transactions
- cf. la documentation de la classe SQLiteDatabase
- La classe Cursor gère le curseur résultat d'une sélection
- récupération des valeurs de champ par des méthodes T getT(int columnIndex)
où T est le type à récupérer : getString, getInt, …
- déplacements avec les méthodes de type move(...)
– moveToFirst(), move(int offset), moveToNext(), …
- la première colonne résultat est numéroté 0
- et bien d'autres
– cf. la documentation de la classe Cursor

SQLite

- L'outil adb permet d'ouvrir un client SQLite
- situé dans le répertoire plateform-tools du SDK
- les commandes du client commencent par le caractère point (.)
  – databases : liste les bases de données
  – help : affiche aide
  – exit : sort de l'application SQLite
- cf. la documentation du client SQLite3
  – lien : www.sqlite.org/sqlite.html

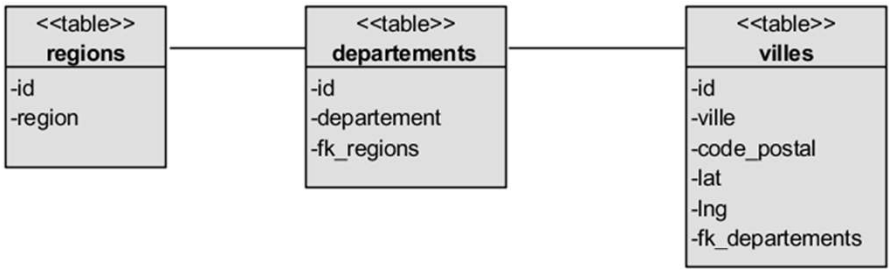
```
$ adb -s emulator-5554 shell
# sqlite3 /data/data/package.appli/databases/maBase.db
sqlite>.databases
```

SQLite

- Nous allons étudier un exemple complet d'une application utilisant une base de données
  - création de la base de données à la première utilisation de l'application
  - interrogation de la base de données
- Exemple dans le ZIP, et autre exemple détaillé ci-après.
- L'application permet de récupérer une ville et ses coordonnées géographique par son code postal
- les données livrées avec l'application sous forme d'un fichier assets/villes.csv

SQLite

- Structure de la base de données



- Extrait du fichier villes.csv

```
"1";"Alsace";"Bas-Rhin";"Auenheim";"67480";"48.81313";"8.00982"
"2";"Alsace";"Bas-Rhin";"Bischwiller";"67240";"48.76712";"7.8604"
"3";"Alsace";"Bas-Rhin";"Dalhunden";"67770";"48.77566";"7.98999"
"4";"Alsace";"Bas-Rhin";"Drusenheim";"67410";"48.76468";"7.95021"
```

SQLite

- Le fichier csv est volumineux
  - il est renommé en csv.mp3 pour qu'il ne soit pas compressé lors de la fabrication de l'APK
    - permet de sauter la barrière du 1 Mo
    - une autre manière de faire aurait été de casser le fich fichiers de maximum 1 Mo
  - ceci n'est pas une bonne pratique
    - import dans SQLite lent
      - environ 5 mn sur l'émulateur
    - APK volumineux

SQLite

- La l'importation dans la base a lieu lors du premier lancement de l'activité FranceActivity
  - l'utilisateur doit être patient
- Une ProgressDialog est affichée pour suivre la progression de l'import
- L'import est effectuée sur un thread secondaire par l'utilisation d'une classe qui étend AsyncTask
  - optimiser au mieux le code
    – éviter de faire des logs lors de l'import
    – ne pas appeler publishProgress à chaque insertion en base

SQLite

- Exemple d'interrogation de la base

```
public List<Ville> getVillesByCP(String cp){
    String[] columns = new String[]{VILLES_KEY_ID,
    VILLES_KEY_VILLE,VILLES_KEY_CP,
    VILLES_KEY_LATITUDE,VILLES_KEY_LONGITUDE};

    String where = VILLES_KEY_CP + " LIKE ? »;
    String[] params = new String[]{cp+ »%"};

    Cursor cursor = db.query(TABLE_VILLES, columns, where, params, null, null, null);
    List<Ville> villes = new ArrayList<Ville>();

    cursor.moveToFirst();
    while(!cursor.isAfterLast()){
        villes.add(new Ville(cursor.getInt(0),
        cursor.getString(1),cursor.getString(2),
        cursor.getDouble(3),cursor.getDouble(3)));
        cursor.moveToNext();
    }

    cursor.close();
    return villes;
}
```

SQLite

- Exemple d'interrogation de la base

```
public List<Ville> getVillesByCP(String cp){
    String[] columns = new String[]{VILLES_KEY_ID,
    VILLES_KEY_VILLE,VILLES_KEY_CP,
    VILLES_KEY_LATITUDE,VILLES_KEY_LONGITUDE};

    String where = VILLES_KEY_CP + " LIKE ? »;
    String[] params = new String[]{cp+ »%"};

    Cursor cursor = db.query(TABLE_VILLES, columns, where, params, null, null, null);
    List<Ville> villes = new ArrayList<Ville>();

    cursor.moveToFirst();
    while(!cursor.isAfterLast()){
        villes.add(new Ville(cursor.getInt(0),
        cursor.getString(1),cursor.getString(2),
        cursor.getDouble(3),cursor.getDouble(3)));
        cursor.moveToNext();
    }

    cursor.close();
    return villes;
}
```

Qu’est-ce qu’un ContentProvider ?

Un contentProvider sert à stocker et récupérer des données et ainsi les rendre accessibles à toutes les applications. C’est le moyen le plus connu de partager des données entres différentes applications. Par exemple, il existe un Content Provider gérant les Contacts d’un téléphone.

ENI : Chapitre 11 : Ch11_BaseDeDonneesExemple

https://developer.android.com/guide/components/loaders.html
https://developer.android.com/reference/android/content/CursorLoader.html

---

https://www.tutorialspoint.com/android/android_content_providers.htm

http://mathias-seguy.developpez.com/tutoriels/android/content-provider/

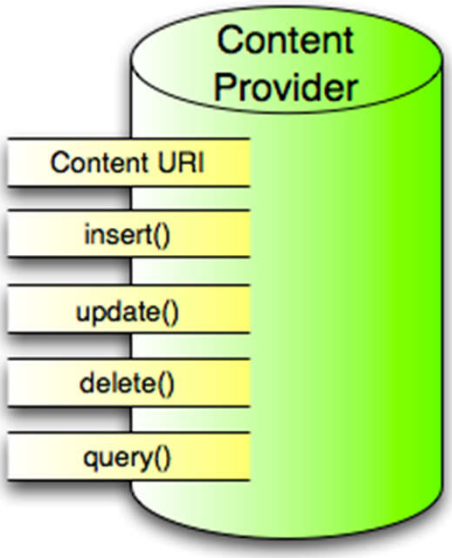
https://developer.android.com/guide/topics/providers/content-providers.html
https://developer.android.com/guide/topics/providers/co

ntent-provider-creating.html

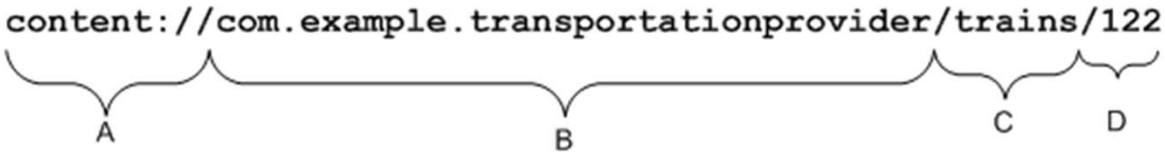
Android propose plusieurs ContentProviders basiques (audio, vidéo, images, informations sur les contacts…). Un contentProvider se compose d’une :

- Uri
- Méthodes (Insert, Update, Delete, Query).

Les interfaces d’accès aux données : Content Providers



Le chemin d’accès vers un ContentProvider se présente toujours sous forme d’une URI. Par exemple :



Les interfaces d'accès aux données : Content Providers

- A : Un préfixe standard, il sert à indiquer que les données sont contrôlées par un ContentProvider.
- B : L'autorité qui contrôle cette URI. Elle identifie le ContentProvider responsable de cette URI.
- C : Permet au ContentProvider de savoir quelle donnée est requêtée par l'url. Ce segment est optionnel. Un content provider peut exposer plusieurs données (ici les trains) mais on pourrait avoir par exemple les voitures, donc ce ContentProvider pourra gérer deux types de données.
- D : L'id de la donnée qu'on souhaite récupérer. (optionnel)

Comment créer un ContentProvider ?

Pour créer un ContentProvider, vous devez :

Mettre en place un systéme pour stocker vos données (les contents providers utilisent généralement le SQLite, la classe SQLiteOpenHelper vous facilite la création de votre base).

Etendre la classe ContentProvider.
Déclarer votre Content Provider dans le manifest (AndroidManifest.xml).

Les interfaces d'accès aux données : Content Providers

Un Content Provider doit surcharger les 6 méthodes suivantes :

- query() : Cette méthode retourne un objet Cursor sur lequel vous pouvez itérer pour récupérer les données.
- insert() : Cette méthode est utilisé pour rajouter des données à notre ContentProvider.
- update() : Cette méthode est utilisé pour mettre à jour une données déjà existante dans notre Content Provider.
- delete() : Cette méthode permet de supprimer une données du Content Provider.
- getType() : Retourne le type MIME des données contenues dans le Content Provider.
- onCreate() : Appeler afin d'initialiser le Content Provider

Cette classe est très simple, elle contient simplement l’identifiant des 3 colonnes. Elle doit absolument implémenter BaseColumns.

Maintenant, nous allons créer une classe qui gérera notre base de données. Elle doit hériter de la classe SQLiteOpenHelper, pour nous faciliter la création de notre base de données.

Démo

Tâche asynchrones et tâche de fond

Réception d'événements avec BroadcastReceiver

Réception d’évènements

- Les événements sont des Intent
- Classe qui permet de réagir à des changements d'état
  - de l'appareil lui même
    - mise en veille, démarrage, boot fini, etc.
  - d'applications entre elles
  - d'un service vers une activité
    - le service ne peut pas interagir directement avec l'IHM de l'activité
- Cette classe réagit à des intentions envoyées par sendBroadcast()

Réception d’évènements

- Deux types principaux de récepteurs
  - BroadcastReceiver
    - à l'écoute de tous les événements
    - appareil ou application
  - LocalBroadcastReceiver
    - à l'écoute des événements dans une même applicatior
    - peut nécessiter android.support.v4
    - les messages envoyés seront locaux
    - via la classe LocalBroadcastManager
    - meilleures sécurité et efficacité
- Une méthode de réception
  - onReceive(Context context, Intent intent)

Réception d’évènements

- Le récepteur peut être déclaré dans le fichier AndroidManifest.xml
  - spécialise un BroadcastReceiver
    - ou une autre sous classe
      - WakefulBroadcastReceiver
      - DeviceAdminReceiver
      - AppWidgetProvider
- souvent plus simple pour réagir aux actions standards
  - par exemple : permet de déclencher un service lorsque l'appareil a complètement démarré
    - ACTION_BOOT_COMPLETED
    - voir aussi la classe WakefulBroadcastReceiver
- Le récepteur peut aussi être enregistré à la demande

Réception d’évènements

- Le récepteur peut aussi être enregistré à la demande
  - permet de faire interagir un service sur une activité
  - l'activité inscrit et désinscrit le récepteur auprès du LocalBroadcastManager
  - pas d'enregistrement du récepteur dans le fichier manifeste

Exemple de filtre de SMS

- Certains SMS sont filtrés par un BroadcastReceiver pour un traitement spécifique
  - dont l'origine est le numéro de téléphone 0123456789
  - la chaîne des récepteurs est alors abandonnée
- Déclaration du récepteur dans le fichier manifeste
  - la priorité est élevée pour que notre récepteur soit le premier de la chaîne

```
<receiver android:name=".SmsFilterBroadcastReceiver" android:exported="true">
    <intent-filter android:priority="100" >
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>
```

Exemple de filtre de SMS

- Code du récepteur

```
public class SmsFilterBroadcastReceiver extends BroadcastReceiver {
    static final String ACTION_FILTREE = "android.provider.Telephony.SMS_RECEIVED";
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equalsIgnoreCase(ACTION_FILTREE)){
            Bundle bundle = intent.getExtras();
            if (bundle != null)
            {
              Object[] pdus = (Object[]) bundle.get("pdus");
              SmsMessage sms = SmsMessage.createFromPdu((byte[])pdus[0]);
              String tel = sms.getOriginatingAddress();
              if(tel.equals("0123456789")){
              Toast.makeText(context,tel+" : "+sms.getMessageBody(),
              Toast.LENGTH_LONG).show();
              this.abortBroadcast();
              }
            }
        }
    }
}
```

Exemple de filtre de SMS

Mise en place des permissions dans le fichier manifeste

```
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.READ_SMS" />
```

Exemple de filtre de SMS

- Un service doit mettre à jour un widget de l'activité
  - le service est dans la même application
  - le service n'a pas accès au widget
  - il envoi un Intent par le LocalBroadcastManager
  - extrait de code

```
public class YahooInterrogationService extends Service {
    private LocalBroadcastManager broadcastManager = LocalBroadcastManager.getInstance(this);
    ...
    @Override
    public int onStartCommand(Intent intent,int flags, int startId){
    ...
    timer.schedule(new TimerTask() {
    @Override
    public void run() {
    ...
    Intent intent = new Intent(MainActivity.FILTER);
    intent.putExtra("value", responseBody);
    broadcastManager.sendBroadcast(intent);
    ...
    } }, 0,10000);
    return Service.START_STICKY;
    }
}
```

Exemple de gestion du récepteur par l'activité

L'activité crée le récepteur et lance le service

- extrait de code

```
public class MainActivity extends Activity {
...
  public static final String FILTER = "org.antislashn.android.FILTRE";
  private BroadcastReceiver receiver;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ...
    receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
      String val = intent.getStringExtra("value");
      ...
    }
  };
  Intent intent = new Intent(this, YahooInterrogationService.class);
  intent.putExtra("name", "GOOG");
  startService(intent);
}
...
```

Exemple de gestion du récepteur par l'activité

- L'activité gère l'enregistrement du récepteur
  - extrait de code

```
...
@Override
protected void onStart(){
  super.onStart();
  LocalBroadcastManager.getInstance(this).registerReceiver(receiver, new
IntentFilter(FILTER));
}
@Override
protected void onStop(){
  LocalBroadcastManager.getInstance(this).unregisterReceiver(receiver);
  super.onStop();
}
...
```

Thread

- Tout traitement long doit donc s'effectuer dans un thread secondaire
  - utilisation de Runnable et Thread
  - utilisation de Handler
  - utilisation de AsyncTask
- AsyncTask simplifie la gestion des threads
  - façon simple d'exécuter un thread et d'afficher un résultat
  - permet de ne pas manipuler directement les threads – gère la création et la communication entre les threads

Créer des tâches asynchrones : AsyncTask, Thread

- Fournit des méthodes de rappels
  - onPreExecute : initialisation du traitement
  – invoquée depuis le thread principal
  - doInBackground : exécution du traitement
  – invoquée depuis le thread secondaire
  - onProgressUpdate : progression du traitement
  – invoquée depuis le thread principal
  - onPostExecute : fin du traitement
  – invoquée depuis le thread principal
  - pour arrêter la tâche : cancel

Créer des tâches asynchrones : AsyncTask, Thread

- Prend trois types génériques
  - Params : type des paramètres fournis à la tâche
  - Progress : type de l'unité de progression du traitement
  - Result : type du résultat du traitement

Créer des tâches asynchrones : AsyncTask, Thread

- Utilisation
  - créer la classe étendant AsyncTask
  – souvent une classe interne du composant applicatif
  - instancier la classe qui étend AsyncTask
  - invoquer la méthode execute sur l'instance de type AsyncTask
- Une tâche n'est traitée qu'une seule fois
  - il faut instancier une nouvelle tâche pour relancer un nouveau traitement

Créer des tâches asynchrones : AsyncTask, Thread

- Exemple de code

```
public class TraitementLong extends AsyncTask<Integer, Integer, Integer>{
    @Override
    protected void onPreExecute(){
        super.onPreExecute();
        Log.d(this.getClass().getName(),"protected void onPreExecute()");
    }

    @Override
    protected Integer doInBackground(Integer... params) {
        int resultat = 0;
        Log.d(this.getClass().getName(),"protected Integer doInBackground(Integer... params)");
        for(int i=0 ; i<100 ; i++){
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {}
            publishProgress(i);
            resultat++;
        }
        return resultat;
    }

    @Override
    protected void onProgressUpdate(Integer... values){
        super.onProgressUpdate(values);
        Log.d(this.getClass().getName(),"protected void onProgressUpdate(Integer... values)");
        String m = values[0]+"%";
        tvAvancement.setText(m);
    }
    @Override
    protected void onPostExecute(Integer result){
        super.onPostExecute(result);
        Log.d(this.getClass().getName(),"protected void onPostExecute(Integer result)");
    }
}
```

invoquée avant l'exécution du thread secondaire

traitement long dans le thread secondaire

appel de `onProgressUpdate`

invoquée par `publishProgress`

invoquée après l'exécution du thread secondaire

Créer des tâches asynchrones : AsyncTask, Thread

- Lancement de la tâche
  - execute(...) reçoit les paramètres qui seront utilisés par doInBackground

```
TraitementLong tl = new TraitementLong();
tl.execute(100);
```

ProgressBar

- Permet de visualiser une progression
  - par défaut un indicateur tournant

```
<ProgressBar
    android:id="@+id/progressBar1"
    style="@android:style/Widget.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:layout_marginTop="20dp" />
<ProgressBar
    android:id="@+id/progressBar2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/progressBar1"
    android:layout_marginLeft="100dp"
    android:layout_marginTop="38dp" />
```

ProgressBar

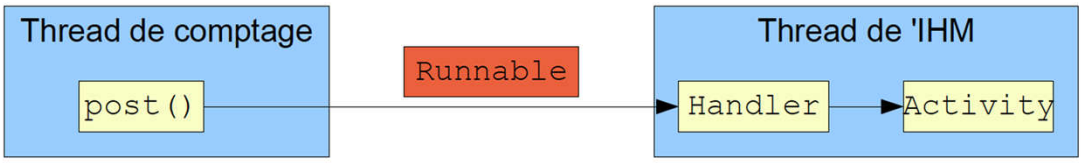
- Code de progression (extrait)
  - dans méthode onCreate()

```
...
new Thread(new Runnable() {
    public void run() {
        while (mProgressStatus < 100) {
            mProgressStatus++;
            try {
                Thread.sleep(50);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            mHandler.post(new Runnable() {
                public void run() {
                    barHorizontal.setProgress(mProgressStatus);
                }
            });
        }
    }
}).start();
...
```

`mHandler` est de type `Handler`

ProgressBar

- Le Handler appartient au thread principal
- thread de l'activité
- Le thread secondaire doit signaler à l'activité de mettre à jour la barre de progression
- le thread secondaire est celui qui compte jusqu'à 100
- Nous utilisons ici une message, le Handler, qui est un Runnable



Service

- Composant applicatif indépendant
  - ne possède pas d'interface graphique
  - fournit une interface (au sens API) de communication avec les autres composants applicatifs
  - hérite de la classe Context
  - l'exécution du service s'effectue dans le thread principal de l'application
- Utilisation d'un service
  - directement
  - par établissement d'une connexion
  - ou un mixte des deux

Intent service :
File/New/Service/Service (Intent Service)
Ensuite implémenter les méthodes avec :

```
try {
    Thread.sleep(3000);
} catch (InterruptedException e) {
    e.printStackTrace();
}
```

Et ajouter un log pour afficher les paramètres.

Bind Service (Voir ci-dessous ou https://developer.android.com/guide/components/bound-services.html)

Service

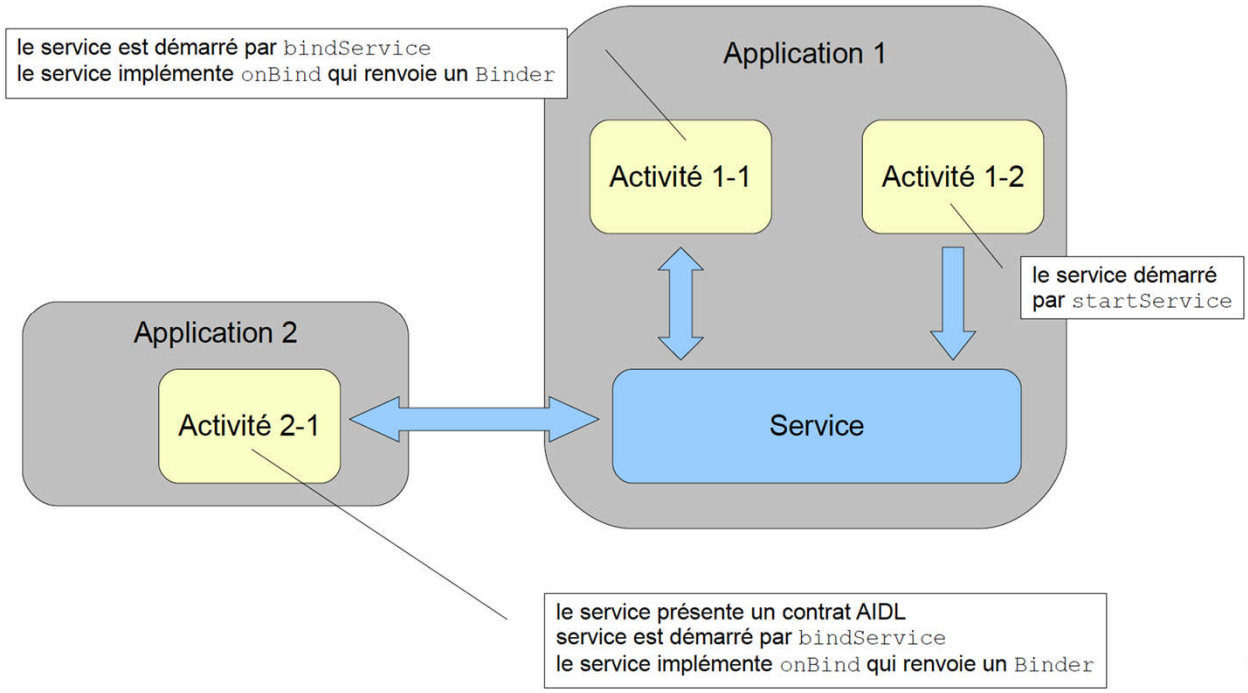
- Le service est déclaré via le fichier manifeste
  - attribut name : classe du service
    – comme une activité, peu commencer par un point si dans le package principal

```
<application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name" >
<activity
android:label="@string/app_name"
android:name=".SimpleServiceActivity" >
<intent-filter >
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<service
android:label="HelloService"
android:name=".HelloService">
<intent-filter>
<action android:name="com.codeattitude.android.service.HELLO_SERVICE"/>
</intent-filter>
</service>
</application>
```

Service

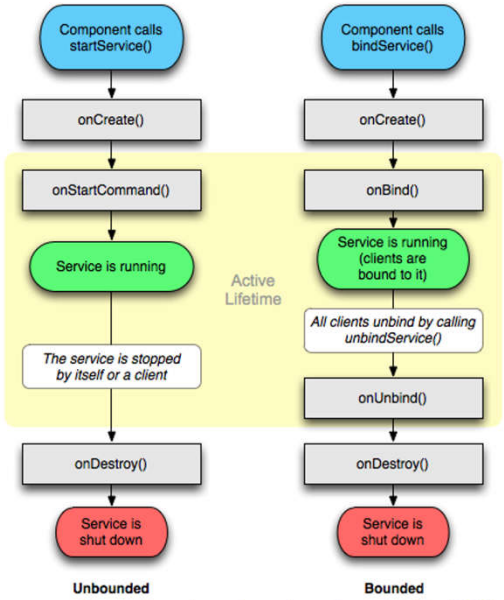
- Un service peut-être local ou distant (remote)
  - le service local n'est utilisable que par les activités de l'application
- Un service peut-être lancé et devenir "autonome"
  - sans interaction avec l'activité qui l'a lancé
    - service de ping régulier par exemple
  - si le service doit inter-agir avec l'activité il faut que l'activité connaisse les méthodes publiques du service
    - par une liaison au service (Bind)
    - par utilisation de messages (Messenger)
    - par un contrat AIDL (Android Interface Definition Language)
      - permet de faire de la communication inter-processus (IPC) en

Service


le service est démarré par bindService
le service implémente onBind qui renvoie un Binder
Application 1
Activité 1-1
Activité 1-2
le service démarré par startService
Application 2
Activité 2-1
Service
le service présente un contrat AIDL
service est démarré par bindService
le service implémente onBind qui renvoie un Binder

Service

- Le cycle de vie est différent si le service est lié ou non



Service

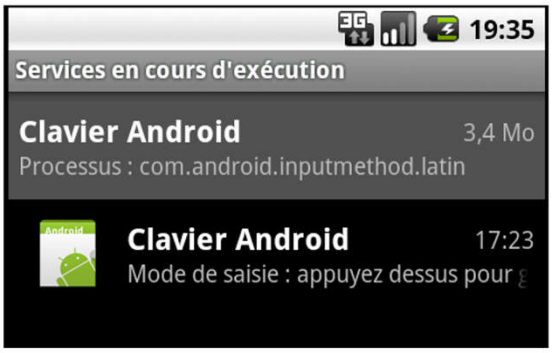
- Démarrer le service depuis l'activité
  - appel de la méthode startService()
    - invoque onCreate(), puis onStartCommand()
    - le service est alors démarré
  - appel de la méthode bindService()
    - invoque uniquement la méthode onCreate()
    - le service est alors démarré et lié avec l'activité
      - l'activité peut interagir avec lui à travers la connexion (Bind)

Service

- Arrêt d'un service démarré par un startService
  - le système n'arrête pas un service sauf s'il a besoin de mémoire
  - le service peut s'arrêter lui-même par un stopSelf
  - un autre composant peut arrêter un service par un appel à stopService
  – le service n'est pas stoppé si d'autres composants sont connectés au service
- Si le service a été démarré par un bindService
le service est arrêté par un unbindService
  - le service est réellement détruit si plus aucun client n'est lié au service

Service

- Voir les services démarrés
  - dans le menu du téléphone
    - `Menu → Settings → Applications → Running Services`



Service

- Démarrer le service avec startService()
  - méthode de la classe Activity
  - prend une intention en paramètre
    – une intention explicite

```
Intent intent = new Intent(this,PingService.class);
startService(intent);
```

– une intention implicite

```
Intent intent = new Intent(".PingService.ACTION");
startService(intent);
```

– startService renvoie null si le service n'est pas démarré, sinon un ComponentName correspondant à un identifiant du composant

Service

- Démarrer le service avec startService()
- le service est démarré de manière autonome
- exemple de code de service

```java
public class PingService extends Service {
    private Timer timer;

    @Override
    public void onCreate(){
        super.onCreate();
        timer = new Timer();
        Log.d(this.getClass().getName(), "onCreate");
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        Log.d(this.getClass().getName(), "onStartCommand");
        timer.schedule(new TimerTask() {
            @Override
            public void run() {
                Log.d(this.getClass().getName(), "FAIRE QUELQUE CHOSE D'UTILE ICI");
            }
        }, 0,10000);
        return START_NOT_STICKY;
    }
...
```

mode de redémarrage du service

Service

- Démarrer le service avec `startService()`
  - suite de l'exemple de la classe `Service`

```
...

    @Override
    public void onDestroy(){
        Log.d(this.getClass().getName(), "onDestroy");
        this.timer.cancel();
    }

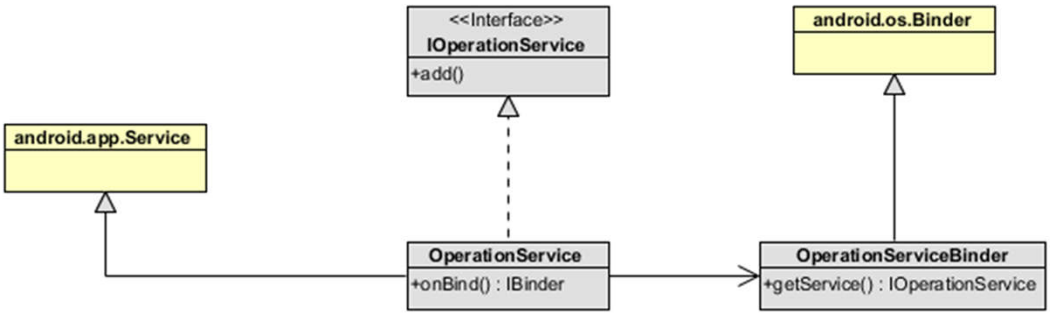
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

pas d'utilisation de connexion

```
<service
    android:label="PingService"
    android:name=".PingService">
    <intent-filter>
        <action android:name=".PingService.ACTION"/>
    </intent-filter>
</service>
```

Service

- Pour créé un service connecté
  - le service doit implémenter la méthode onBind et retourner un IBinder
    - le IBinder défini l'interface de communication avec le service
    - la classe qui implémente IBinder possède une méthode renvoyant une interface vers le service



Service

- Code du service (extrait)

```
public class OperationsService extends Service implements IOperationService {
    private final OperationServiceBinder osBinder = new OperationServiceBinder();

    public class OperationServiceBinder extends Binder{
        IOperationService getService(){
            return OperationsService.this;
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        Log.d(this.getClass().getName(), ">> onBind()");
        return osBinder;
    }
...
```

Service

- Pour démarrer et utiliser le service connecté
- implémenter l'interface ServiceConnection
  – possède des méthodes callback invoquées lors des connexions et déconnexions au service
  – onServiceConnected permettra de récupérer une référence vers le service via la connexion
- l'appel de la méthode de déconnexion est effectuée si Android service
- invoquer la méthode bindService
  – en lui passant une instance de l'implémentation de ServiceConnection

Service

- Code de l'activité se connectant au service (extrait)

```
public class ServiceBindingActivity extends Activity {
   private boolean connecte = false;
   private IOperationService service;
   private ServiceConnection serviceConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder binder) {
       service = ((OperationServiceBinder)binder).getService();
       connecte = true;
       long r = service.add(1,2,3,4);
       Log.d(this.getClass().getName(), ">> onServiceConnected("+name+")");
       Toast.makeText(ServiceBindingActivity.this, "Appel du service : "+r,
       Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onServiceDisconnected(ComponentName name) {
     Log.d(this.getClass().getName(), ">> onServiceDisconnected("+name+")");
     connecte = false;
    }
}
```

Service

- Code de l'activité se connectant au service (extrait)

```
...
public void bindService(View v){
Toast.makeText(this, "Bind service", Toast.LENGTH_SHORT).show();
Intent intent = new Intent("org.antislashn.android.service.OPERATIONS_SERVICE");
bindService(intent,serviceConnection,Context.BIND_AUTO_CREATE);
}
public void unbindService(View v){
Toast.makeText(this, "Unbind service", Toast.LENGTH_SHORT).show();
if(connecte)
unbindService(serviceConnection);
}
...
```

Service

- En résumé
  - Côté service
  - – créer l'interface du service
  - – créer le service
    - implémenter l'interface
    - étendre la classe Service
  - – ajouter une classe étendant de Binder
    - souvent une classe interne du service
    - avec une méthode retournant l'implémentation de l'interface du service
  - – implémenter la méthode onBind()
    - retourne une instance de Binder
  - – déclarer le service dans le manifeste

Service

- En résumé
  - côté client (l'activité)
  – créer une implémentation du listener ServiceConnecti
  – sur la méthode onServiceConnected
    - récupérer le service via le Binder
  – appeler la méthode bindService(...) pour se connecter au service
  – appeler la méthode unbindService(...) pour se déconnecter du service

Service

Démo

Accès au matériel

- Capteurs
  - Accéléromètre
  - Gyroscope
  - Capteur magnétique
  - …

Annexes :

1. Historique des versions d'Android - https://fr.wikipedia.org/wiki/Historique_des_versions_d%27Android
2. Android History - https://www.android.com/intl/fr_fr/history
3. Android 4.0 APIs - https://developer.android.com/about/versions/android-4.0.html
4. Market share http://uk.businessinsider.com/apple-iphone-market-share-in-the-us-v-android-2015-12
5. L'architecture android - https://openclassrooms.com/courses/creez-des-applications-pour-android/l-architecture-d-android
6. Android studio - https://developer.android.com/studio/index.html
7. Android studio features - https://developer.android.com/studio/features.html
8. Android Runtime - https://en.wikipedia.org/wiki/Android_Runtime
9. BackStack - https://developer.android.com/guide/components/tasks-and-back-stack.html
10. Touch - https://developer.android.com/training/gestures/detector.html
11. Source de certains slides - antislashn.org
12. Content Provider - http://www.tutos-android.com/contentprovider-android

Level Up
http://blog.stablekernel.com/level-up-with-android-studio-shortcuts-and-live-templates/

Annexe

- Débug en WiFi :
- adb tcpip 5555
  adb connect 192.168.xx.xx
  adb devices –l

Débug WiFi :
adb tcpip 5555
adb connect 192.168.xx.xx
adb devices –l

Obtenir l'adresse IP du téléphone :
adb shell ip -f inet addr show wlan0

Revenir au mode USB :
adb usb

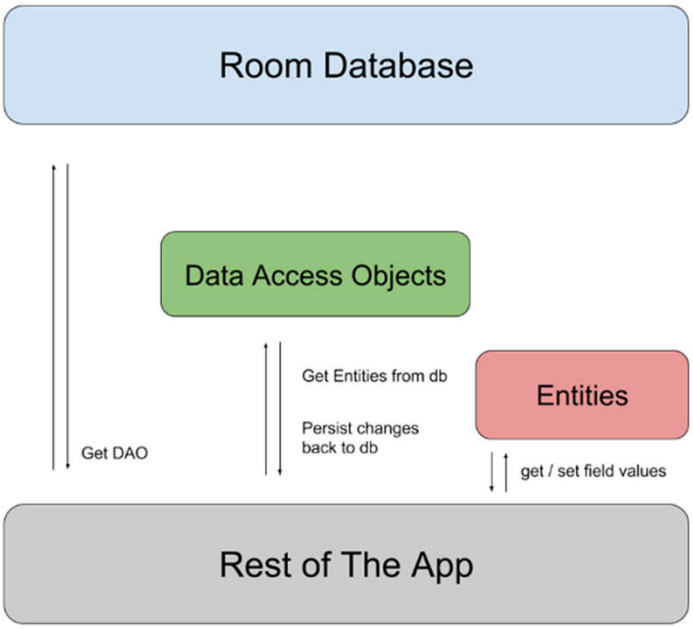
Annexe

- Kotlin

Kotlin

Histoire
http://www.journaldunet.com/developpeur/expert/67258/kotlin--le-renouveau-de-java.shtml
Kotlin officiel
https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/
Kotlin Android
https://developer.android.com/kotlin/index.html

JetPack : Room



https://developer.android.com/jetpack/

Pas à pas :
https://codelabs.developers.google.com/codelabs/android-room-with-a-view/#0

Room

```
@Entity
public class User {
    @PrimaryKey
    public int uid;

    @ColumnInfo(name = "first_name")
    public String firstName;

    @ColumnInfo(name = "last_name")
    public String lastName;
}
```

Room

```
@Dao
public interface UserDao {
    @Query("SELECT * FROM user")
    List<User> getAll();

    @Query("SELECT * FROM user WHERE uid IN (:userIds)")
    List<User> loadAllByIds(int[] userIds);

    @Query("SELECT * FROM user WHERE first_name LIKE :first AND " +
           "last_name LIKE :last LIMIT 1")
    User findByName(String first, String last);

    @Insert
    void insertAll(User... users);

    @Delete
    void delete(User user);
}
```

Room

- ```
  @Database(entities = {User.class}, version = 1)
  public abstract class AppDatabase extends
  RoomDatabase {
      public abstract UserDao userDao();
  }
  ```
  ```

Room

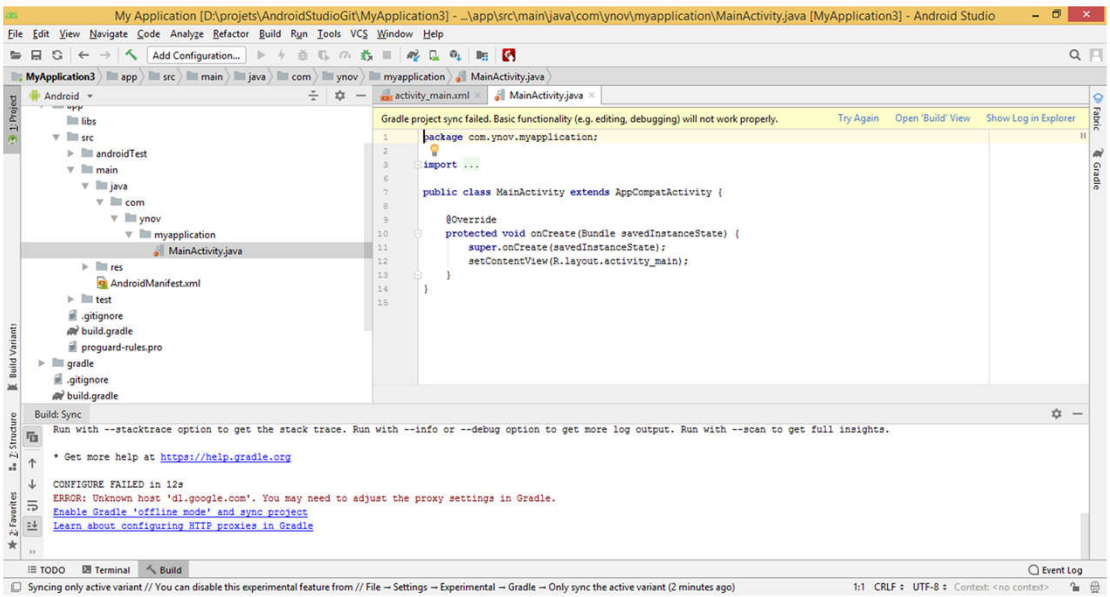
- AppDatabase db = Room.databaseBuilder(getApplicationContext(), AppDatabase.class, "database-name").build();

Dans l'Activity/Application

Ignorer un champ :
@Ignore
Bitmap picture;

https://medium.com/androiddevelopers/7-pro-tips-for-room-fbadea4bfbd1

Erreurs courantes



Pas de connexion internet.