



Android



Jean-Claude Tarby
Laboratoire CRISTAL
Université Lille 1





Android

- But : découvrir la programmation Android
- Aucune connaissance préalable sauf Java
- Quel est votre niveau Android ?

- Evaluation
 - Un TP individuel : 50%
 - Tirage au sort des sujets (cf. slide suivant)
 - Explication des TPs juste après ça
 - Une petite application à réaliser dans la semaine qui suit le dernier cours : 50%



Tirage au sort des TPs

- Un n° au hasard de 1 à 24 (FA) / 25 (FI)
- Votre nom
- ➔ votre sujet (avec son niveau de difficulté) et votre ordre de passage

Liste des sujets

Début :

- le 12 octobre (FI), un TP en fin de séance
- le 20 octobre (FA), 4h de TPs



Le TP

- Pensez que vous parlez à des novices en Android !
- Des fichiers sur un git (gitlab de la fac ou équivalent → accès facile)
 - Code source initial
 - Code source final
- Un "support" de quelques pages (en pdf)
- Faire le TP à toute la promo (et moi :-))
 - en 30mns on comprend le sujet traité, à quoi ça sert, comment on crée et gère les choses (de façon simple)
Conseil : faites au plus simple pour le TP, 30 mns = très court !
Exemple : data en dur pour ListView ou JSON...
- N'hésitez pas à donner (dans le support par exemple) des pointeurs pour des compléments d'infos.



Android, Google, San Francisco...





Avant tout...

- On charge les téléphones pour pouvoir les utiliser tout à l'heure...
 - ~~Fiches de prêts impérativement à chaque séance~~
 - Dorénavant pensez à aller chercher votre téléphone/tablette avant la séance !!!
- Laisser les téléphones éteints pour le moment
 - Charge + rapide
 - On testera d'abord sur l'émulateur ☺ ☹



Vous connaissez déjà Android ?

- Qu'en pensez-vous ?
 - Facile, simple, puissant... OU difficile, compliqué... ?
 - Pourquoi cet avis ?
 - Que pensez-vous par exemple :
 - des « fragments »,
 - de la gestion des BDD Sqlite
 - des layouts ?



Avis personnel

- Android est très puissant
 - On peut faire tout ce qu'on veut (y compris attaquer le noyau)
 - Fonctionne sur un vaste ensemble de périphériques
 - de la montre connectée à la TV connectée, en passant par la voiture connectée, etc.
- Mais...
 - Très long à apprendre
 - Parfois très complexe (tordu ?)
 - En perpétuel changement
 - Chaque année une nouvelle version au moins...
 - Outils encore trop basiques.



Nouveautés depuis 4.4/L

- New Android Runtime (ART)
 - Avant Dalvik, compilation à la volée
 - Maintenant ART avec optimisation du garbage collector, compilation en code machine natif lors de l'installation, support de débogage
- Nouvelles notifications
- Nouveaux composants graphiques
 - RecyclerView (mieux que ListView), CardView (vs. Fragment ?), touch feedback animations,...
- Modifications autour du multimédia
- Modifications autour du réseau
- ...
- Détails à <http://developer.android.com/preview/api-overview.html>



Nouveautés avec les versions 5...

- Nouvelles *nouvelles* notifications
- Nouvelles permissions (gestion par demande à utilisateur)
- Nouveau design (look and feel)
- Nouveau SDK et API, bien sûr...

- Android Studio 1.3 est sorti
- Android 6 arrive...



Android 6

- Permissions en live, etc.
 - <https://github.com/googlesamples/android-RuntimePermissions>
 - <https://developer.android.com/preview/features/runtime-permissions.html>
- Autres nouveautés...
 - Géoloc
 - <http://developer.android.com/training/location/receive-location-updates.html>
 - <http://developer.android.com/training/location/index.html>
 - Floating Action Button, SnackBar, CoordinatorLayout...
 - Cf. <http://android-developers.blogspot.fr/2015/05/android-design-support-library.html>
 - ...
- Google APIs
 - <https://developers.google.com/android/>



Android 7...

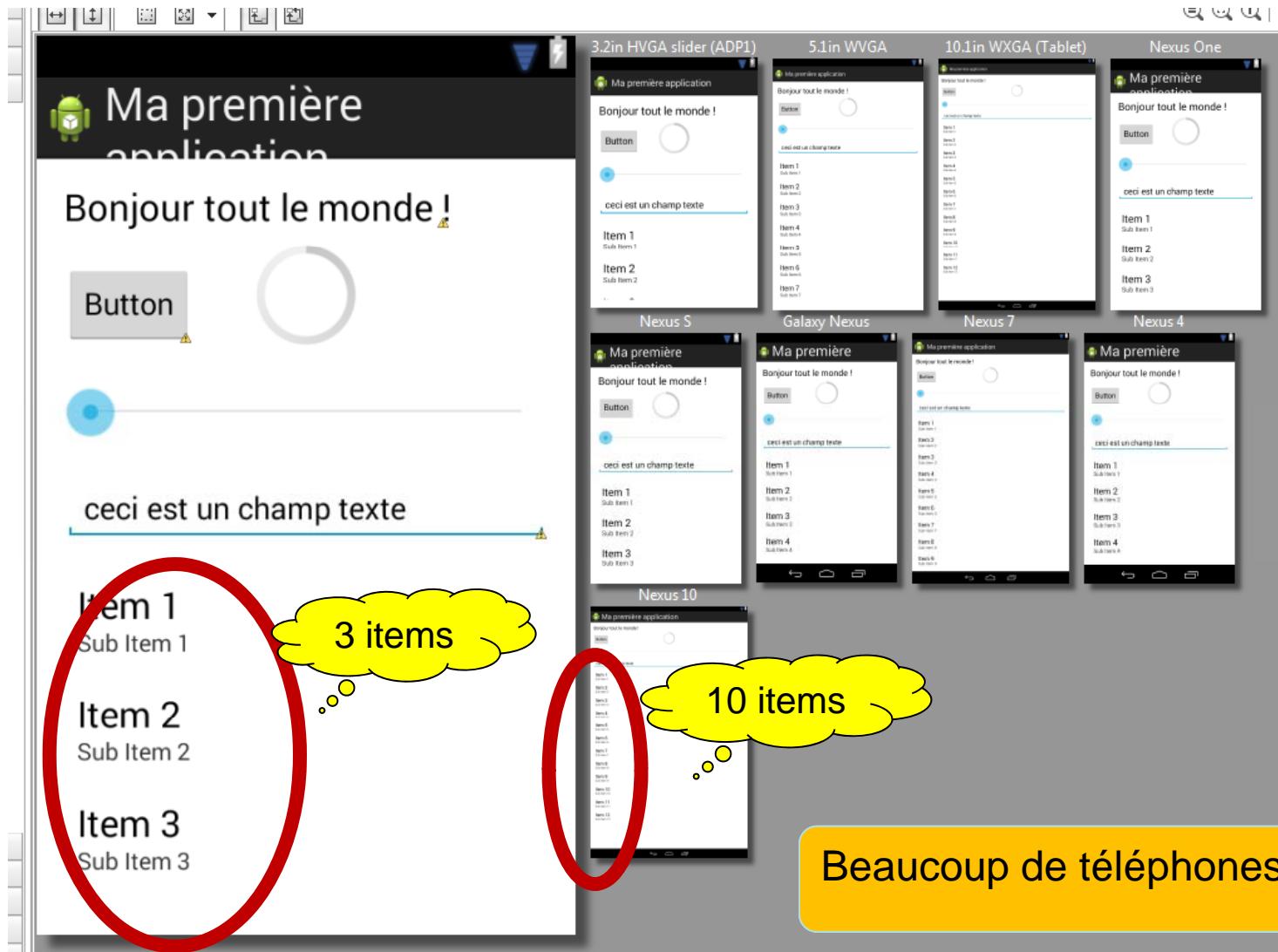
- Allez voir par vous-mêmes ☺
 - <https://developer.android.com/about/versions/nougat/index.html>



Ergonomie, UI Patterns, ...



Attention aux tailles des écrans !





Attention aux tailles des écrans !



Beaucoup de téléphones,
tablettes, ordinateurs...





Attention aux tailles des écrans !





Des montres et des bagues connectées





Des télévisions connectées





Des voitures connectées



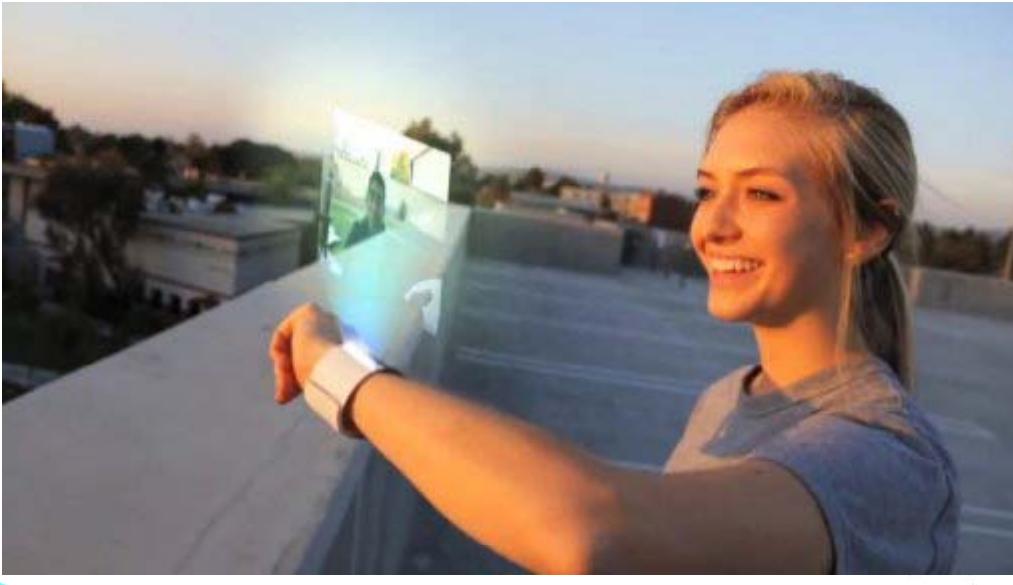


Des très grands écrans connectés





Des hologrammes connectés demain ?



Exemple à
<http://www.infohightech.com/un-casque-de-chantier-a-realite-augmentee/>





Des hologrammes connectés demain ?

Ca devient très compliqué !

de la « card » Google Glass à
900” !!!





Ressources Google pour le design !

- De + en + d'informations (good/bad news)
 - <https://design.google.com/> (site complet à part)
 - <http://developer.android.com/design/index.html>
- Des cours !
 - <http://developer.android.com/training/index.html>
- Material Design
 - <http://developer.android.com/design/material/index.html>
 - <http://www.google.com/design/spec/material-design/introduction.html>
- Icônes
 - <https://www.google.com/design/icons/index.html>
- Layouts
- Palettes de couleurs
- Polices de caractères
- ...
- <https://www.google.com/design/spec/resources>
- Cf. aussi <http://android-developers.blogspot.fr/2015/05/android-design-support-library.html>
- Voir aussi ce très bon site : <http://www.materialup.com/>
- Et aussi [bonnes pratiques de Google pour Android.pdf](#)



Design / Training / Samples ...

The screenshot shows a web browser window displaying the Android developer samples website at developer.android.com/samples/index.html. A red circle highlights the top navigation bar, which includes links for Developers, Design, Develop (which is underlined in green), Distribute, Training, API Guides, Reference, Tools, Google Services, and Samples.

About the Samples

What's New

- Admin
- Background
- Connectivity
- Content
- Input
- Media
- Notification
- RenderScript
- Security
- Sensors
- System
- Testing
- UI

Samples

Welcome to code samples for Android developers. Here you can browse sample code and learn how to build different components for your applications. Use the categories on the left to browse the available samples.

Each sample is a fully functioning Android app. You can browse the resources, source files and see the overall project structure. You can copy and paste the code you need, and if you want to share a link to a specific line you can double-click it to get the URL.

Import Samples from GitHub

Android Studio provides easy access to import Android code samples from GitHub and is the recommended method to retrieve Android code samples.

To import a code sample into Android Studio:

- In the Android Studio menu, select **File > Import Sample** to open the Import Sample wizard.
- Select a sample to import and click **Next**.
- Specify the application name and project location if different from the displayed settings.
- Click **Finish**.

The sample project opens in a new Android Studio project.

Note: When starting Android Studio, you can also select **Import an Android code sample** in the Welcome to Android Studio wizard to import a sample project from GitHub as a new project.

FR 18:14 02/11/2015



Ressources Google (layout)

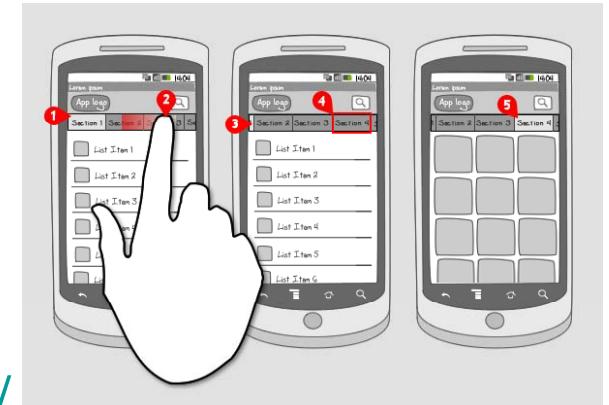
The screenshot shows a PDF document titled "Material Design: Mobile Keylines and Spacing" viewed in Adobe Acrobat Pro. The document is page 1 of 14, at 75.2% zoom. The title page features a blue background with white text. To the right, there is a white panel with text and diagrams explaining keyline and spacing guides. The text includes:

- "Use the layers palette to toggle keyline and spacing guides"
- "Fixed increments determine widths of structural UI elements. For mobile we use a 56dp increment."
- "Vertical keylines determine placement of type and icons from edges of elements."
- "Vertical sizing determines the height of content blocks."
- "Horizontal margins determine spacing."



Qualité de vos applications

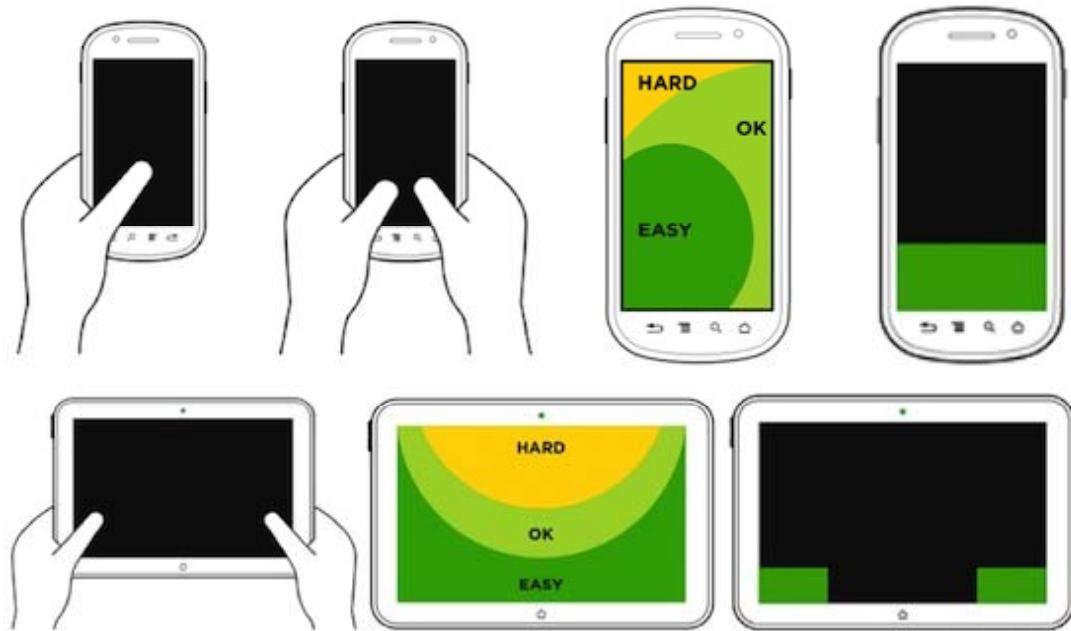
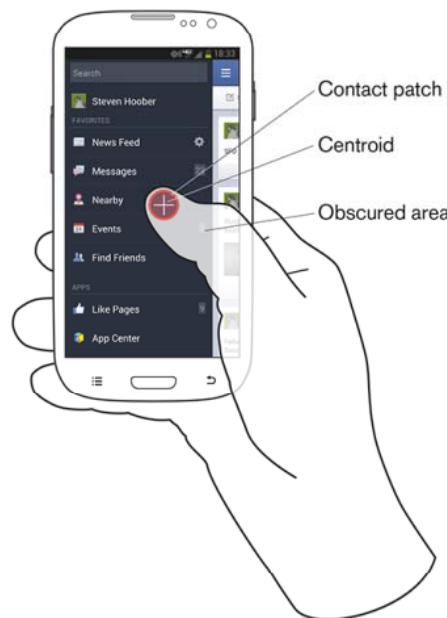
- De plus en plus de guidelines à respecter !
 - <http://developer.android.com/distribute/googleplay/quality/core.html>
 - À quand un outil pour le faire automatiquement ?
- Sources d'inspiration
 - <http://www.mobile-patterns.com/>
 - <http://www.androidpatterns.com/>
 - <http://androidpttrns.com/>
 - <http://pttrns.com/>
 - <http://www.android-app-patterns.com/>
 - <http://mycolorscreen.com/>





Quelques articles sur l'IHM mobile

- <http://www.simpleweb.fr/tag/tablette/> et <http://www.simpleweb.fr/> (mine d'infos)
 - <http://www.simpleweb.fr/2012/04/24/pour-bien-demarrer-dans-la-conception-dinterfaces-mobiles/>
 - <http://www.simpleweb.fr/2011/06/27/des-bibliotheques-de-composants-dinterfaces-mobiles/>
- <http://uxmag.com/articles/framework-for-designing-for-multiple-devices>





Aperçu de ce qu'on pourrait voir... ...si on avait 200h !

- Installation de l'environnement + 1ère application automatique
- Activités (StartActivity, Intents)
- Permissions
- IHM
 - layouts
 - widgets de base
 - Listview et Adapters
 - Spinner
 - Boite de dialogue...
 - Styles
 - Menus et ActionBar
- Data
 - BDD SQLite
 - Persistance des données (sauf SQLite)
 - Passage de données entre applications (hors fournisseurs de contenu/content provider)
 - préférences d'une application (sharedPreferences)
 - Stockage de texte simple dans un fichier
 - Fournisseur de contenus
- QRCode (lire et générer)
- Préférences (utilisateur et téléphone)
- C2DM (cloud to device messaging);
- Capteurs
 - GPS
 - Géoloc
 - Calcul itinéraire
 - Accéléromètre
 - Réseau
 - 3G, Wifi, Bluetooth...
 - Autres capteurs...
- AsyncTask
- Service
- Notification
- Threads
- Filtres
- Notification
- Appel/réception d'appel et de SMS
- Multimédia
 - son
 - vidéo
 - synthèse vocale
 - text-to-speech
- JSON et XML (parsing)
- Web services, connexion PHP /MySQL
- Publication sur le market + stats
- ...et encore plein d'autres choses...



Sommaire général

- Ce qu'on verra en 12 h de cours/TD/TP
 - Installation d'Android Studio
 - Architecture générale et fonctionnement d'une application
 - Layout
 - Activité
 - Ressources...
 - (Mise en ligne dans Google Play)
 - « Et le reste ? », me direz-vous...



Le reste = 24 TP individuels

- Affectation par tirage au sort d'un sujet de TP
- Préparation du TP
 - Documentez-vous sur le sujet
 - Documents à écrire et à mettre à disposition sur GitLab (celui de Lille 1, ou autre...)
 - Code source, documents PDF, images, sons...
 - Pas de plagiat ! Sinon je ne serai « pas cool » !
- Réalisation du TP avec la promotion et moi-même
 - Durée : 30 mns maxi !
 - Pensez à tester à l'avance sinon je vous arrêterai à 30 mns.



Quelques outils pour maquetter...

- Indigo Studio :
 - description succincte à <http://www.developpez.com/actu/51009/Infragistics-lance-Indigo-Studio-l-outil-gratuit-pour-la-conception-de-prototypes-d-IU-interactifs-pour-applications-Web-Desktop-et-mobile/>
 - outil dispo à <http://www.infragistics.com/products/indigo-studio>.
 - V2 payante (\$400 !), mais 30 jours gratuits
 - V1 gratuite : me contacter si nécessaire.
- Balsamiq (30 jours gratuits) : <http://balsamiq.com>
- Pencil (gratuit) : <http://pencil.evolus.vn/>
- un super petit outil pour montrer vos maquettes en contexte : <http://placeit.breezi.com/>
 - devenu payant ! ☺ ... mais « print screen » possible ☺
 - Quelques images encore gratuites
- <https://code.google.com/p/android-ui-utils/> et <http://www.journaldugeek.com/2011/11/28/android-design-preview-consulter-vos-maquettes-mobile/> : pour connecter Photoshop à vos mobiles
- <http://wireframesketcher.com/>
- <http://www.justinmind.com/prototyper/free-edition> <http://www.justinmind.com/> (a priori Très Bien)
- <http://www.build.me/> (a priori Très Bien)
- <http://fluidui.com/android/> (**adopté en 2013-2014 par vos prédecesseurs en e-services**)
- <http://www.invisionapp.com/> : gratuit pour un projet
- <http://www.justinmind.com/> : gratuit 30 jours
- <http://www.axure.com/>
- <http://www.pixate.com/> : prototypage cross-platform (nouveau, pas encore testé)



(Très) Bonnes références (tutoriels, vidéos, forums)

- <http://developer.android.com/index.html>: LA référence
- <http://developer.android.com/training/index.html> : tutoriaux (mais des erreurs ! et pas toujours à jour !)
- <http://developer.android.com/guide/practices/index.html> : « best practices » officielles
- <http://www.google.com/design/spec/material-design/introduction.html> : la référence pour le Design Android
- <http://www.siteduzero.com> ==> <http://fr.openclassrooms.com/>: TB tutoriaux pour Android et iPhone
- <http://www.vogella.com/articles/Android/article.html> et <http://www.vogella.com/android.html>
 - dont ActionBar et Navigation Drawer (menu glissant) : <http://www.vogella.com/articles/ActionBar/article.html>
- <http://stackoverflow.com/>: forum pour Android et iPhone
- <http://android.developpez.com/cours/>
- <http://www.tutos-android.com/>
- <http://forum.frandroid.com>
 - <http://forum.frandroid.com/topic/16808-serie-de-tutoriauxdevelopper-sous-android/>
- <http://android-coding.blogspot.fr> : tutoriels moins courants
- <http://codentrick.com/> : tutoriels Android, Web
- <http://www.mysamplecode.com> : tutoriels Android et iOS
- <http://blog.restomaniak.com/restomaniak-sur-votre-mobile/> : tutoriels
- <http://www.tutomobile.fr/> : tutoriels pour Android et iPhone
- <http://www.univ-orleans.fr/lifo/Members/Jean-Francois.Lalande/enseignement/android/presentation-android.html> : cours pour se faire une idée générale d'Android
- Vidéos
 - <https://egghead.io/>
 - <http://thenewboston.org>
 - <http://www.mybringback.com/tutorials/>



Des outils pour compléter le développement

- Pour générer les icônes et autres graphiques à la bonne taille
 - <https://romannurik.github.io/AndroidAssetStudio/>
- Développer en natif Android et IOS avec... Javascript !
 - <https://www.nativescript.org/>
 - Source : <http://i-programmer.info/news/167/8367.html>
 - <http://www.appcelerator.com/>
- Développer en natif Android et IOS avec C#
 - <http://xamarin.com/platform>
 - <https://xamarin.com/getting-started/android>
 - <http://developer.xamarin.com/guides/android/>
 - http://developer.xamarin.com/guides/android/getting_started/



Quelques outils utilisés par les E-Services en Platine ou en stage

- Robotium
 - <https://code.google.com/p/robotium/>
 - Robotium is an Android test automation framework that has full support for native and hybrid applications...
 - Hébergé maintenant à <https://bintray.com/robotium/generic/releases/view>
- Spoon (pas celui de l'INRIA)
 - <http://square.github.io/spoon/>
 - Distributing instrumentation tests to all your Androids
- Push Notification
 - <http://urbanairship.com/>
 - Payant mais avec une base gratuite qui suffit pour des tests en Platine
 - <https://parse.com>
 - Propose aussi d'autres solutions autour du mobile dont le push <https://parse.com/products/push>
- Butter Knife
 - <http://jakewharton.github.io/butterknife/>
 - View "injection" library for Android
 - Permet d'écrire moins de lignes de code et + lisibles
- CrashLytics
 - <https://try.crashlytics.com/>
 - Analyse de crash
- Picasso
 - <https://github.com/square/picasso>
 - Pour télécharger des images et gérer un cache sans pb
- AndroidAsync
 - <http://koush.com/AndroidAsync>
 - Pour faire de l'asynchrone sans pb
 - Voir sur le même site:
 - <http://koush.com/ion> : Android Asynchronous Networking and Image Loading
- CouchBase
 - Accès à BDD NoSQL externes très facilement et très rapidement (???)
 - A priori, stage → bcp + compliqué que ça!
 - <http://www.couchbase.com/fr>
- PouchDB :
 - BDD local et distane synchronisée
 - <https://pouchdb.com/>



Quelques outils utilisés par les E-Services en Platine ou en stage

- Asciidoctor
 - <http://asciidoctor.org/>
 - Pour générer des docs toujours à jours, et très jolies (et incluant des diagrammes)
- Swagger
 - <http://swagger.io/>
 - permet de spécifier, visualiser, et consommer des web-services REST
 - permet à un développeur frontend de s'abstraire d'une connaissance précise du web-service
- JSON-Generator
 - <http://www.json-generator.com/>
 - générer des données au format JSON
 - très bien pour travail séparé entre équipe Front et équipe Back
- EventBus
 - <https://github.com/greenrobot/EventBus/blob/master/README.md>



Autres références

- PhoneGap : <http://phonegap.com> pour produire sur 7 plateformes mobiles avec un seul code + développement hybride (natif + webview...)
- Ionic !!! <http://ionicframework.com/>
 - <http://ionicmaterial.com/>
 - <https://github.com/loicknuchel> (cf. Ch'ti JUG), <http://loic.knuchel.org/blog/>
 - <http://mcgivery.com/>, <http://devgirl.org/>, <http://www.raymondcamden.com/>
 - <https://material.angularjs.org/>
- Bibliothèque SQLite : <http://ormlite.com>
 - Il y en a d'autres :
 - <https://github.com/pardom/ActiveAndroid>
 - <https://code.google.com/p/droidpersistence/> (bon ?)
 - <http://androrm.the-pixelpla.net/> (bon ?)
- Bibliothèque pour les webservices + cache
 - <http://www.technotalkative.com/android-volley-library-example/>
- Autres bibliothèques:
 - <http://www.androidviews.net/category/libraries/> : plein de refs de bibliothèques dont :
 - <http://androidannotations.org/> pour écrire beaucoup moins de code grâce aux annotations !
 - <https://github.com/amigold/FunDapter> pour simplifier les Adapter dans les ListViews



Evaluation

- TP individuel : 50 %
- Application à réaliser après les cours : 50 %

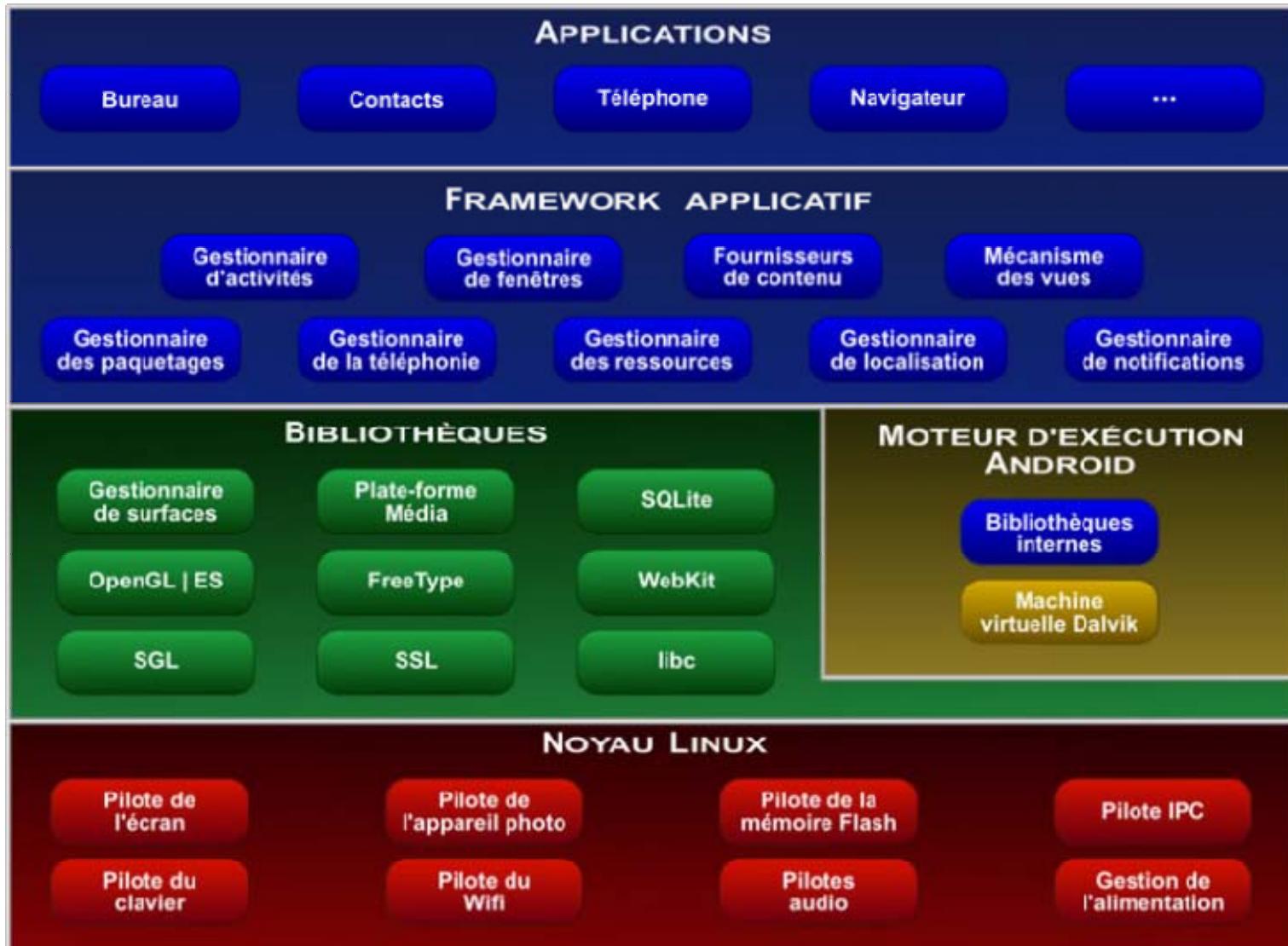


Préambule

- Android est né en septembre 2008 (V 1.0). Aujourd’hui V 7.
 - Comparatif TB fait des versions :
<http://socialcompare.com/fr/comparison/android-versions-comparison>
- Sachez que les SDK Android contiennent beaucoup d’exemples de code (dans « samples »), pensez à les consulter).
 - Vous pouvez aussi les utiliser directement dans Android Studio comme support de nouveaux projets



Préambule





Remarque pour votre salle de cours

- Vos postes sont configurés pour reconnaître nos téléphones ACER Liquid et Stream + quelques tablettes.

```
/etc/udev/rules.d/51-android.rules :  
SUBSYSTEM=="usb", ATTRS{idVendor}=="0502", ATTRS{idProduct}=="3202", MODE="0666"  
SUBSYSTEM=="usb", ATTRS{idVendor}=="0502", ATTRS{idProduct}=="3317", MODE="0666"  
SUBSYSTEM=="usb", ATTRS{idVendor}=="04e8", ATTRS{idProduct}=="681c", MODE="0666"
```

- Si vous voulez utiliser votre propre téléphone Android, ou un autre téléphone/tablette, il faudra faire ajouter de nouvelles règles par l'administrateur du M5 :
 - lancer la commande « `lsusb` » qui vous affichera un message du style
Bus 001 Device 013: ID 22b8:708b Motorola PCS
 - Cela signifie que `idVendor=22b8` et `idProduct=708b`. La ligne sera donc :
SUBSYSTEM=="usb", ATTRS{idVendor}=="22b8", ATTRS{idProduct}=="708b", MODE="0666"
 - Envoyez **TOUS** les « `Bus xxx Device yyy: ID zzzz:zzzz`
`xxxx` » **dans un seul mail groupé** à mickael.carlier@univ-lille1.fr

+ d'infos à <http://doc.ubuntu-fr.org/android>



Introduction rapide à Android

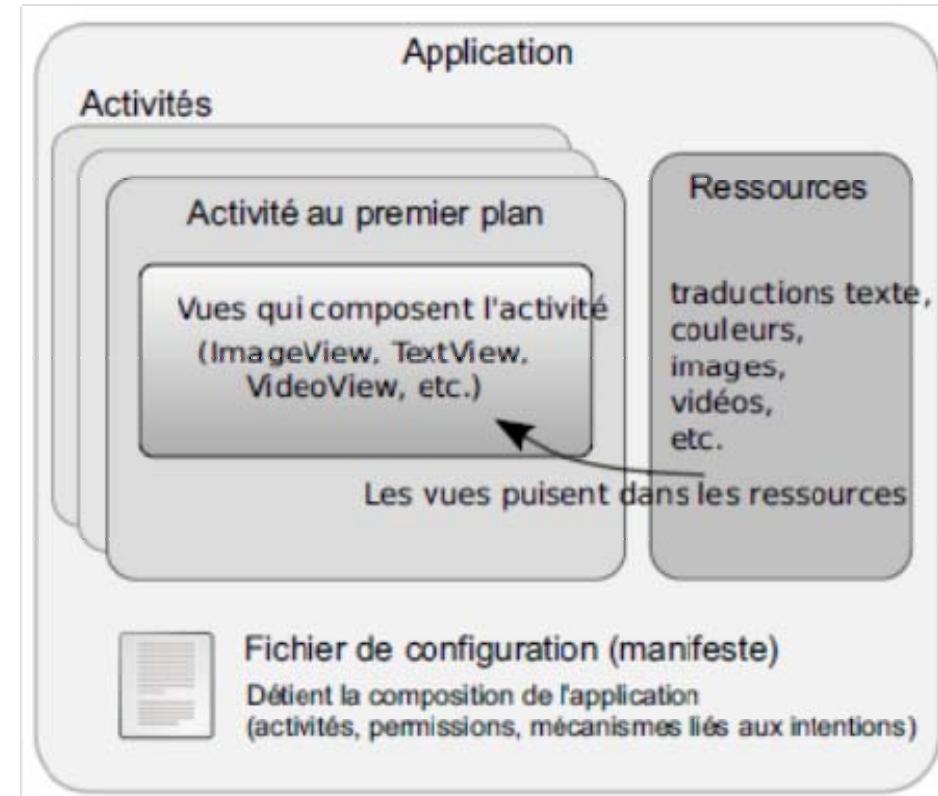
Quelques concepts de base

- Une application est décomposée en *activités* (et *fragments*).
- Une activité peut appeler d'autres activités (de l'application ou d'autres applications).
 - Si Activité 1 appelle Activité 2, on empile A2 sur A1
 - Android gère la mémoire et ces piles d'activités en conséquence
 - Bouton « back » = dépiler la dernière activité
- Les appels d'activités se font par des *Intents*
 - Intent, Activité, Vue
 - « qui sait faire X ? », « qui sait faire Y ? »
 - « métaphore de personnes avec habits »



Quelques concepts de base

- Activité
- Vues et contrôles (+ leur mise en page) ;
- Ressources
- le fichier Manifest





Activité

- Une Activité
 - Est un écran contenant des vues et des contrôles composant l'IHM de façon logique
 - Elle contient une hiérarchie de vues et de sous-vues.
 - Exemple : un formulaire d'ajout de contacts.
 - Est décomposée en :
 - logique + gestion du cycle de vie dans le **code java** de sa classe (héritant de Activity)
 - Il existe des classes d'activité qui facilitent (parfois) le travail, par exemple ListActivity
 - **IHM** : soit définie dans le code Java ci-dessus soit dans un fichier XML
- En général, si une application a « N » écrans, elle aura « N » d'activités.
 - Ce n'est plus vraiment le cas avec les Fragments...



Vues et Contrôles

- **Vues**
 - Ce sont les parties composant l'IHM
 - Elles contiennent des composants organisés suivant des mises en page (layout)
- **Contrôles** = widgets (boutons, checkbox...)
 - sous-ensemble des vues
 - accèdent aux textes et aux images qu'ils affichent en puisant dans les fichiers « ressources » de l'application



Ressources

- Stockées dans le dossier « res ».
- Contiennent les textes, images, sons, ... mais aussi les layouts, etc.

Type de ressource	Répertoire associé	Description
Valeurs simples	res/values	Fichiers XML convertis en différents types de ressources. Ce répertoire contient des fichiers dont le nom reflète le type de ressources contenues : 1. <code>arrays.xml</code> définit des tableaux ; 2. <code>string.xml</code> définit des chaînes de caractères ; 3. ...
Drawables	res/drawable	Fichiers <code>.png</code> , <code>.jpeg</code> qui sont convertis en <code>bitmap</code> ou <code>.9.png</code> qui sont convertis en "9-patches" c'est-à-dire en images ajustables. Note : à la construction, ces images peuvent être optimisées automatiquement. Si vous projetez de lire une image bit à bit pour réaliser des opérations dessus, placez-la plutôt dans les ressources brutes.
Layouts	res/layout	Fichiers XML convertis en <code>mises en page d'écrans</code> (ou de parties d'écrans), que l'on appelle aussi gabarits.
Animations	res/anim	Fichiers XML convertis en objets animation.
Ressources XML	res/xml	Fichiers XML qui peuvent être lus et convertis à l'exécution par la méthode <code>resources.getXML</code> .
Ressources brutes	res/raw	Fichiers à ajouter directement à l'application compressée créée. Ils ne seront pas convertis.



Multi-lingue...facile !

- res/values
 - strings.xml
- res/values-**fr**
 - strings.xml
- res/values-**es**
 - strings.xml



Mode portrait et paysage...facile !

- res/layout
- res/layout-land



Disposition sur l'écran...facile !(?)

- res/layout
- res/layout-large
- res/layout-xlarge



Disposition sur l'écran...facile ! (?)

- res/layout
- res/layout-large
- res/layout-xlarge

« deprecated » depuis Android 3.2.

Utilisez plutôt les « largeurs » comme ci-après.



Résolution et Style... facile ?

- res/values
 - dimens.xml
 - styles.xml
- res/values-v11
- res/values-v14
- res/values-w820dp
- res/values-sw820dp
- ...
- Solution : les fragments (?)



Ressources

- Toutes ces ressources seront placées dans l'APK.
- Android crée une **classe statique R** qui sera utilisée pour accéder aux ressources depuis le code.
 - Dans « /gen » ou « /app/build »

```
package com.eyrolles.android.exemples;
public final class R {
    public static final class string { ①
        public static final int bienvenue-0x7f040000; ②
        public static final int texte_bouton_quitter-0x7f040001;
        public static final int texte_titre_ecran-0x7f040002;
    };
    public static final class Layout {
        public static final int ecran_de_demarrage-0x7f030001;
        public static final int ecran_principal-0x7f030000;
    };
    public static final class drawable {
        public static final int image_android-0x7f020000;
    };
}
```



Fichier Manifest

- Fichier indispensable à chaque application qui décrit entre autres :
 - le *point d'entrée* de votre application (quel code doit être exécuté au démarrage de l'application) ;
 - quels *composants* constituent ce programme ;
 - les *permissions* nécessaires à l'exécution du programme (accès à Internet, accès à l'appareil photo...).
 - dans les dernières versions, les « tools: »
 - <http://tools.android.com/tech-docs/tools-attributes>
 - Exemple : tools:listItem="@+id/simple_list_item_2" />



Fichier Manifest

- Une application « complexe » Android peut faire appel à :
 - des **activités** (composant applicatif)
 - des **services** (composant applicatif)
 - des **fournisseurs de contenus** (composant applicatif), par exemple pour accéder à des données en BDD
 - des **gadgets** (composant applicatif)
 - des **Intents**
 - des **Récepteurs d'Intents**
 - des **Notifications**.



Intents

- Les objets **Intents** permettent de diffuser des messages demandant la réalisation d'une action → permettent de fournir ou de demander des services. C'est Android qui « choisit » QUI fera l'action demandée.
- **Récepteurs d'Intents** : permettent à l'application d'être à l'écoute pour répondre aux objets Intents qui lui demanderaient quelque chose.
- **Notification** : signale une information à l'utilisateur sans interrompre ses actions en cours.
- **Filtres d'Intents** : un objet Intent peut mentionner explicitement un composant cible. Dans le cas contraire, Android choisit le meilleur composant grâce aux filtres (cf. <intent-filter> du fichier de config Manifest).



Permissions

- Pour accéder à certaines fonctionnalités, on doit déclarer des permissions dans le Manifest
 - par exemple pour envoyer des SMS, accéder au compte Google, etc.
- Permet à l'utilisateur d'en avoir conscience et de choisir.
- Permet aussi au Play Store de filtrer les applications pour ne proposer que celles qui peuvent fonctionner sur le périphérique connecté.
 - Peut être contourné en utilisant intelligemment `<uses-permission>` et `<uses-features android:required = "false">`
 - À vous de faire attention dans votre code !



Permissions

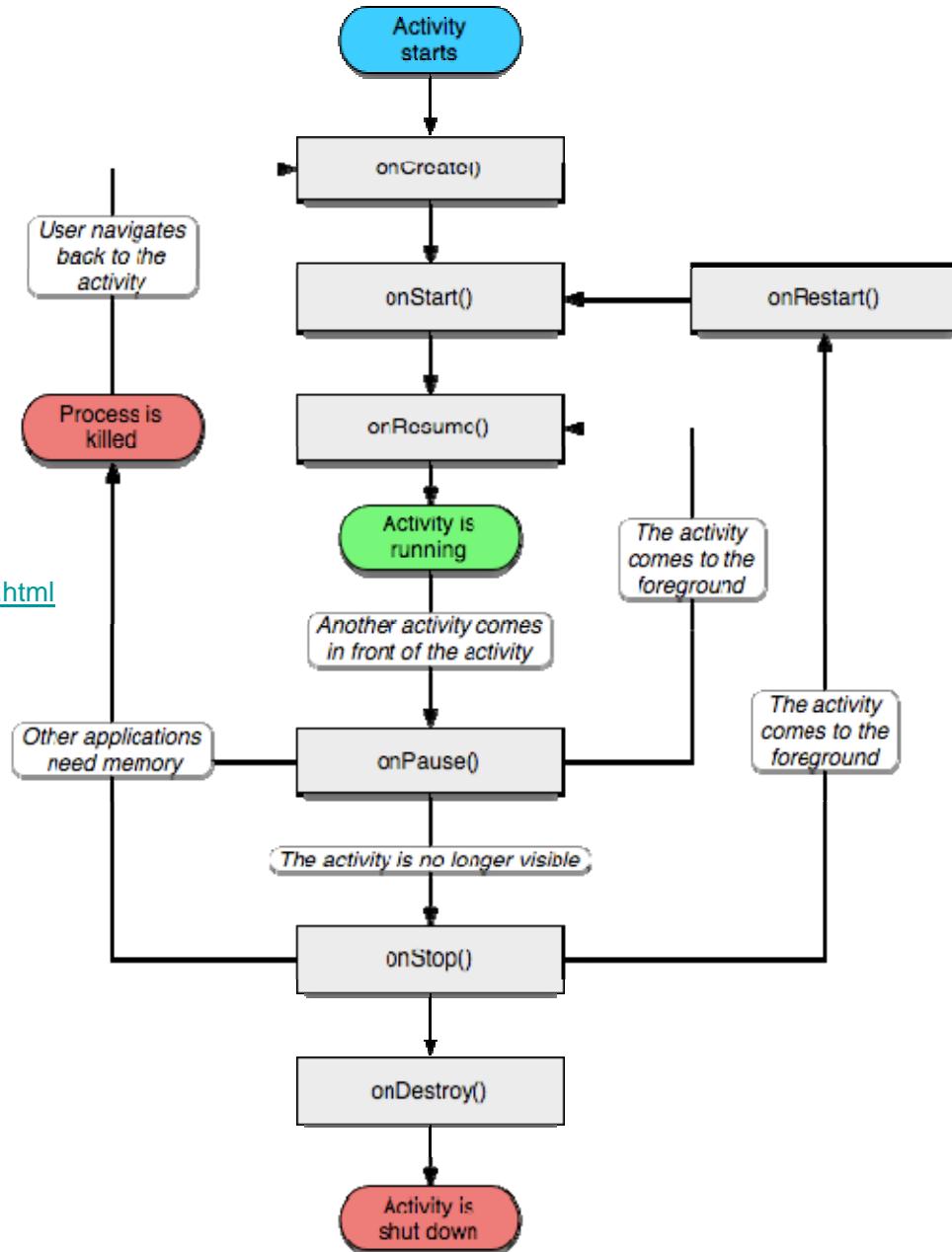
- Pour accéder à certaines fonctionnalités, on doit déclarer des permissions dans le Manifest
 - par exemple pour envoyer des SMS, accéder au compte Google, etc.
- Permet à l'utilisateur de faire son choix et de choisir.
- Permet aussi de faire attention aux changements avec Android 6+... ! Les permissions se font maintenant « à la volée » !
 - Peut être contourné avec `<uses-permission>` et `<uses-permission android:required = "false">`
 - À vous de faire attention dans votre code !



Cycle de vie d'une application

source : <http://developer.android.com/reference/android/app/Activity.html>

On verra un exemple plus tard...





Outils de développement (1/2)

- **ADT / Eclipse + plug-in Android**
 - <http://developer.android.com/sdk/adt.html>
 - Bug dans la version 23.0.2 (juillet 2014) !
 - Importation de projets
 - Détails à <http://developer.android.com/sdk/adt/knownissues.html#libraryfeature>
 - » Détails : <http://developer.android.com/sdk/support-repository.html>
 - **Android Studio**
 - Basé sur IntelliJ IDEA
 - <http://developer.android.com/sdk/installing/studio.html>
 - Futur bon outil ? (utilisé partout à SF)
 - + d'options que dans ADT, mais encore plein de « soucis »

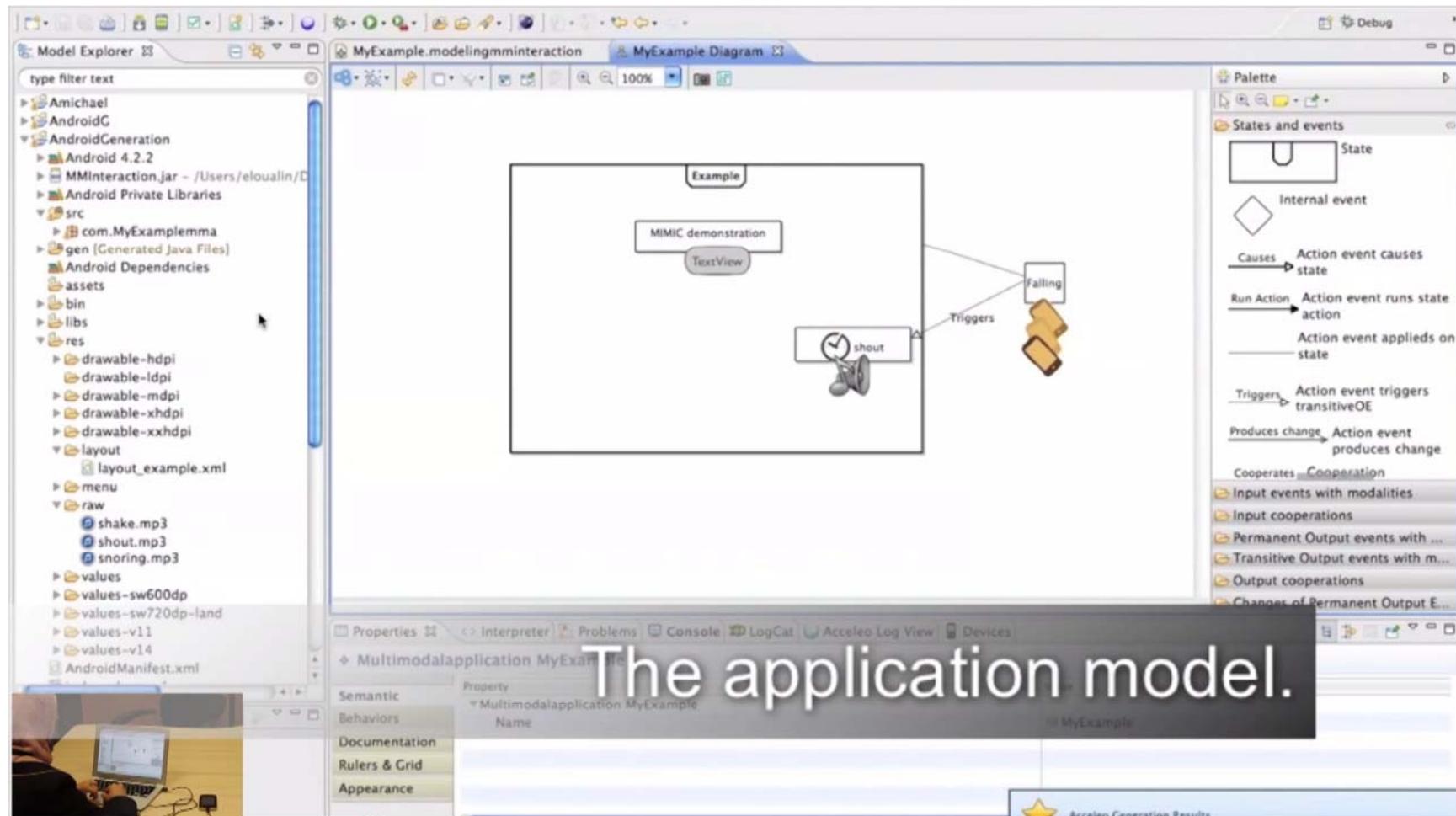


Outils de développement (2/2)

- **AppInventor** (MIT)
 - Basé sur Scratch du MIT : <http://scratch.mit.edu/>
 - <http://appinventor.mit.edu>
- **Intel XDK**, pour du cross-platform
 - <https://software.intel.com/en-us/intel-mobile-development-kit-for-android>
 - <http://xdk-software.intel.com/index.html>
 - <https://software.intel.com/en-us/html5/tools>
- Notepad++ et autres éditeurs + compilateur en ligne de commande... (si vous êtes courageux !)



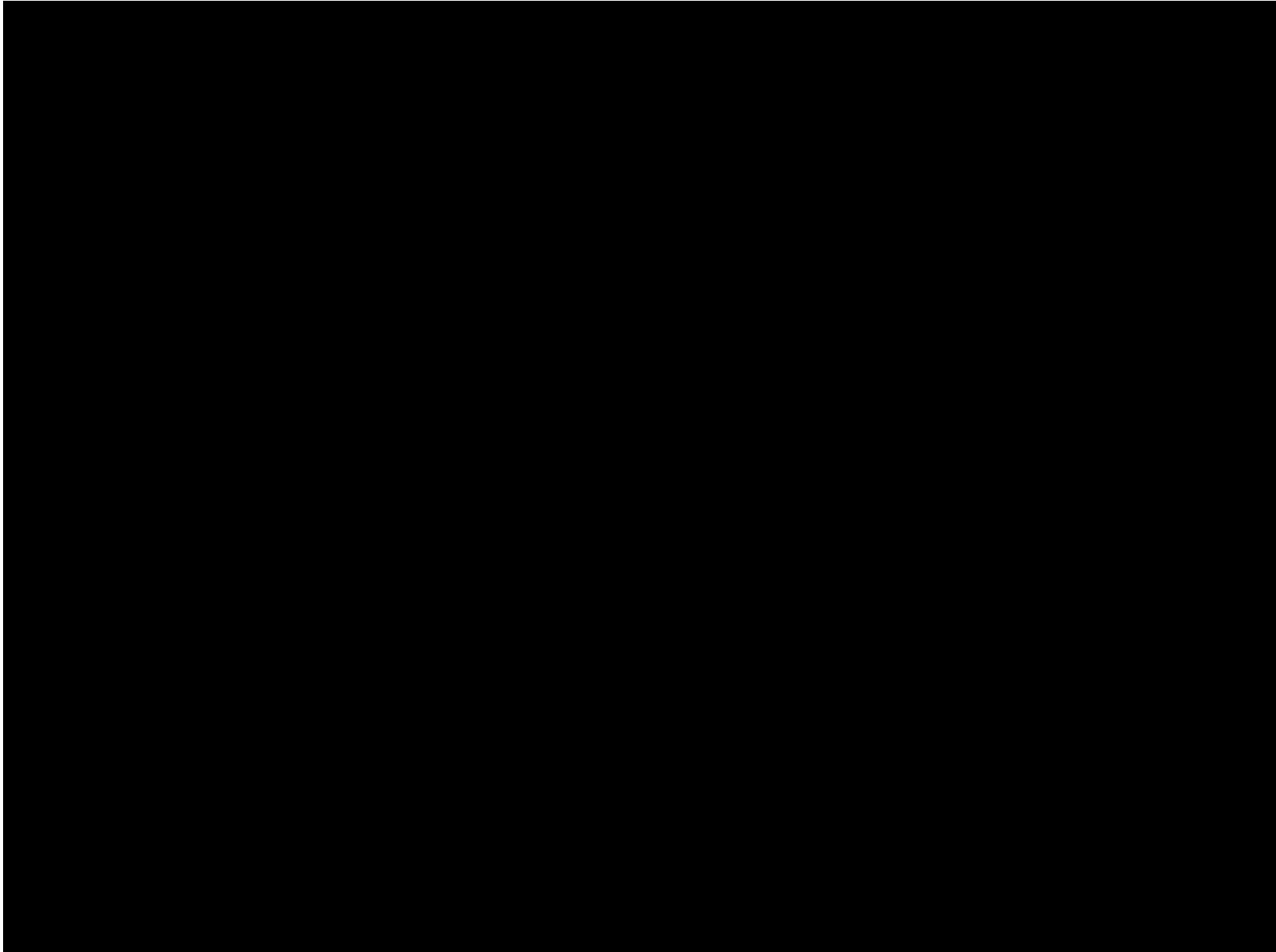
Futurs outils ?



<http://youtu.be/g0Ekioe1CQ0?list=UUofGGFqq9t1kfKOAzn5HJGQ>

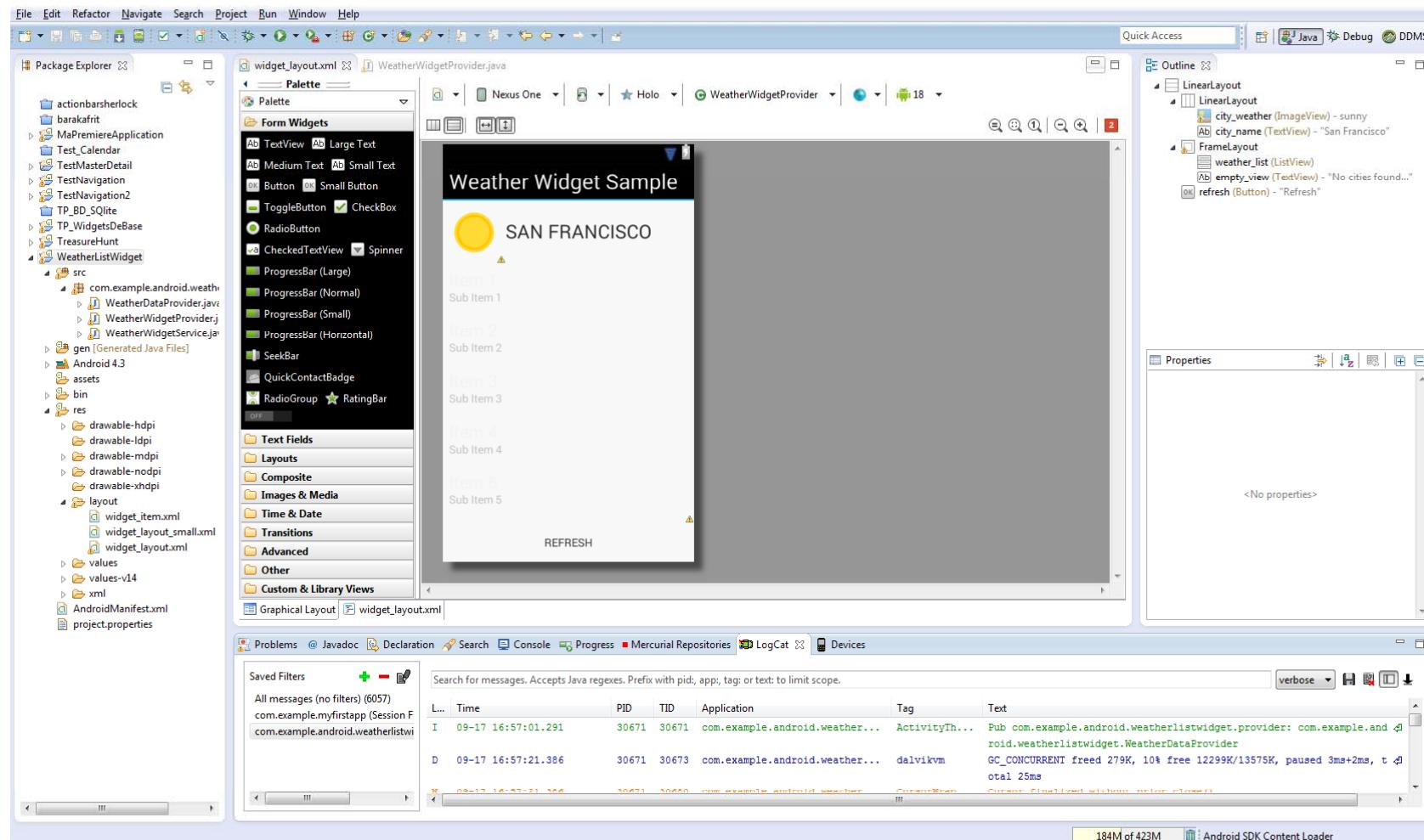


Futurs outils ?





ADT (ou Eclipse + plug-in Android)





Android Studio

Arborescence
différente de ADT

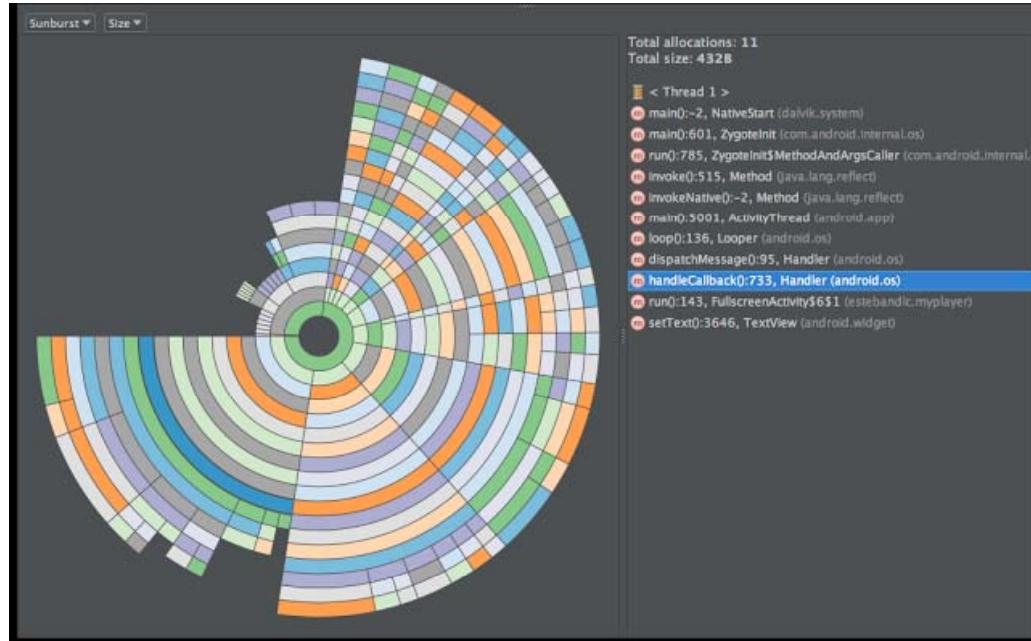
The screenshot shows the Android Studio interface with the following components:

- Project Structure View:** On the left, a red circle highlights the "MyApplication" project structure. It includes the "gradle", "src", and "res" directories, along with files like "build.gradle", "MyApplication.java", and "AndroidManifest.xml".
- Design View:** The main area shows the "activity_main.xml" layout. The "Layouts" palette lists various layout types, and the "Widgets" palette lists various UI components. A preview window shows a simple application with the title "My Application" and the text "Hello world!".
- Component Tree:** A sidebar on the right shows the component tree for the current screen, starting with a "Device Screen" containing a "RelativeLayout" which contains a "TextView" with the string "@string/hello_world".
- Logcat:** At the bottom, the Logcat window displays log messages from a Samsung GT-P1000 device running Android 2.3.3 (API 10). The log includes entries related to battery updates and debugger acceptance.

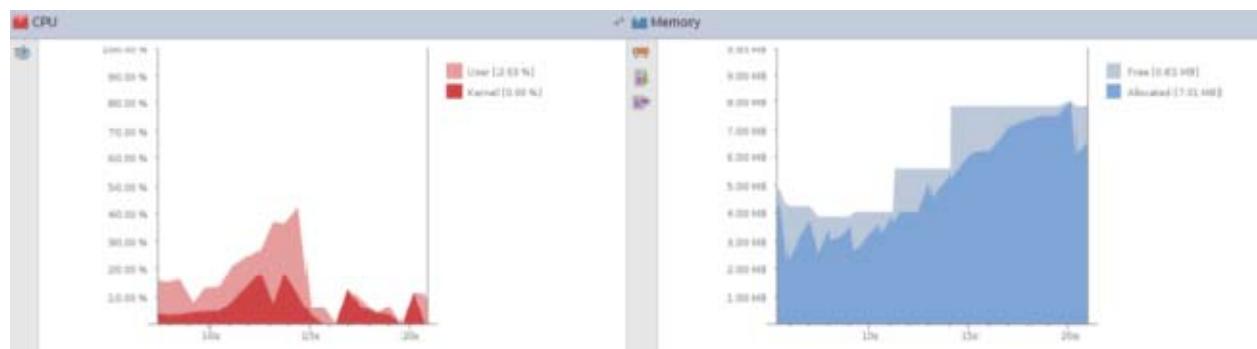
Plus d'infos à
<http://developer.android.com/tools/studio/index.html>



Android Studio : outil puissant... ?



CVS,
Git,
Mercurial...





Android Studio : outil puissant... ?

VCS

```
C:\Users\Jean-Claude\AndroidStudioProjects\MyFirstApplication\app\src\main\java\fr\univ_lille\ieea\tarby\myfirstapplication\MyActivity.java

Ignore whitespace: Do not ignore | Highlight: By word | Current | 3 differences | Deleted | Changed | Inserted

07/09/2015 17:15 - MyActivity.java (Read-only)
package fr.univ_lille.ieea.tarby.myfirstapplication;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MyActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar
        getMenuInflater().inflate(R.menu.menu_my, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }
    }
}

public class MyActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);
    }

    @Override
    public boolean onKeyLongPress(int keyCode, KeyEvent event) {
        return super.onKeyLongPress(keyCode, event);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }
    }
}
```



Android Studio : outil puissant... ?

VCS

C:\Users\Jean-Claude\AndroidStudioProjects\MyFirstApplication\app\src\main\java\fr\univ_lille\ieea\tarby\myfirstapplication\MainActivity.java

Ignore whitespace: Do not ignore | Highlight: By word | Current | 1 package | 2 import | 3 .support.v7.app.AppCompatActivity; | 4 import | 5 android.os.Bundle; | 6 import | 7 android.view.Menu; | 8 import | 9 android.view.MenuItem;

07/09/2015 17:15 - MainActivity.java (Read-only)

```
package fr.univ_lille.ieea.tarby.myfirstapplication;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is available.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }
    }
}
```

3 differences Deleted Changed Inserted

Voir aussi
l'inspecteur de
code (Analyse)
pour vos projets ☺



Android Studio : outil puissant... ?

- Sur la bonne voie en tous cas...
- Exemple :

```
<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">
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />
</LinearLayout>
```



Android Studio : outil puissant... ?

- Sur la bonne voie en tous cas...
- Exemple

The screenshot shows the Android Studio interface with the layout editor open. The code editor displays an XML layout file named `activity_my.xml`. A specific line of code is highlighted:

```
<EditText android:id="@+id/edit_message"  
    android:layout_weight="1"  
    android:layout_width="wrap_content"
```

A context menu is open over the highlighted line, listing several inspection options:

- Replace size attribute with 0dp
- Disable inspection
- Edit 'Inefficient layout weight' inspection settings
- SUPPRESS: Add tools:ignore="InefficientWeight" attribute
- Override Resource in Other Configuration...
- Set Namespace Prefix to Empty

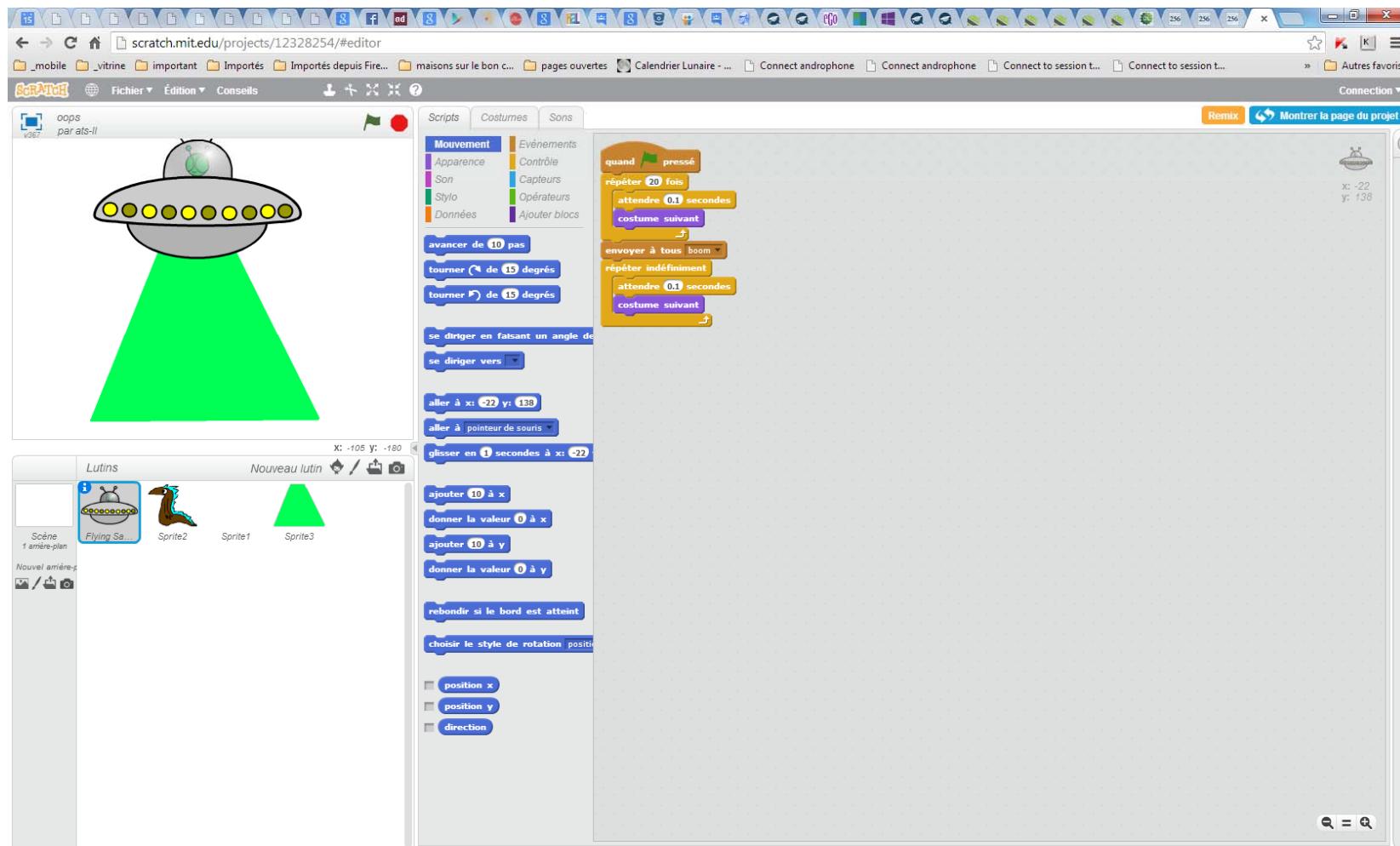


Android Studio : outil puissant... ?

- Emulateur toujours (très) long à démarrer
- Solution 1 : émulateur de **GenyMotion**
 - www.genymotion.com
 - (sous forme de plugin pour Android Studio)
- Solution 2 : voir le téléphone en live à l'écran
 - plugin **Vysor** sur chrome
 - <https://play.google.com/store/apps/details?id=com.koushikdutta.vysor&hl=fr>
 - et aussi AirDroid
 - <http://web.airdroid.com> et <https://play.google.com/store/apps/details?id=com.sand.airdroid&hl=fr>

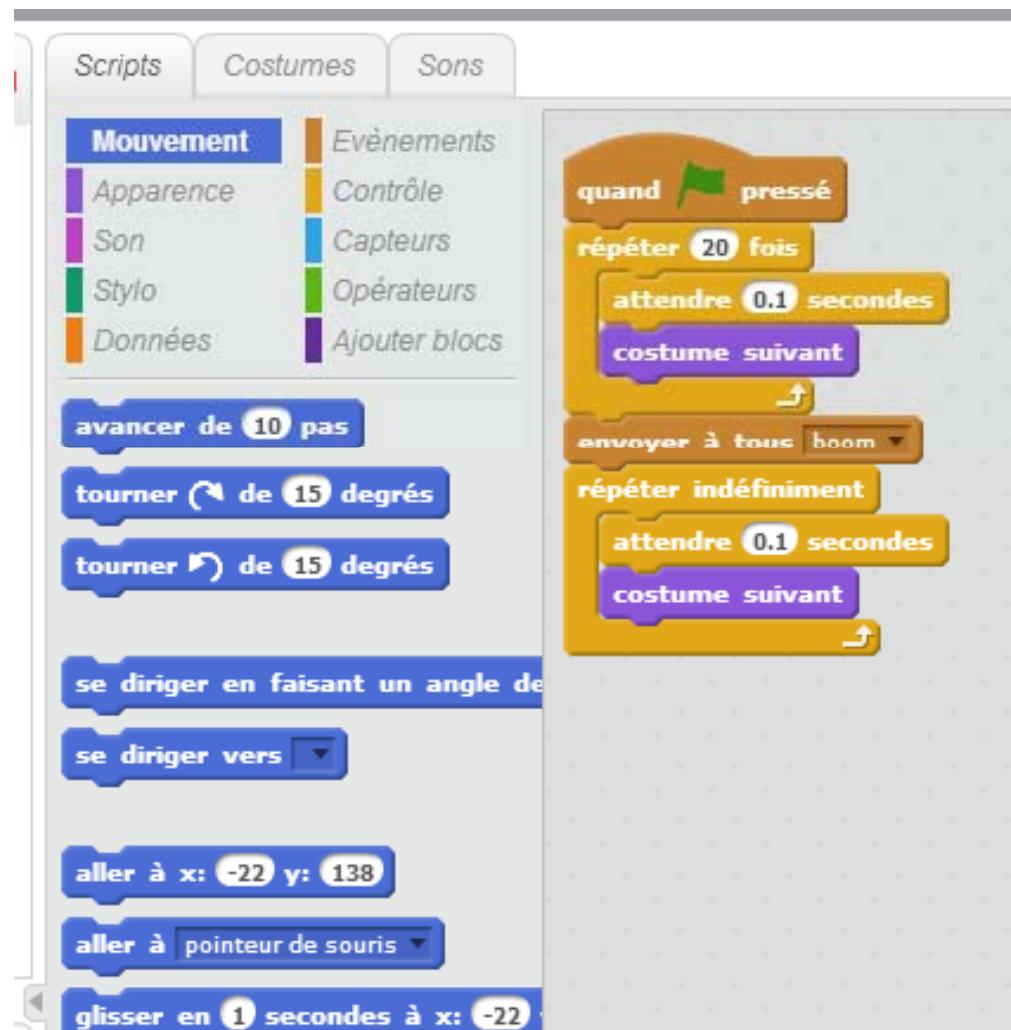


Scratch (à partir de 8 ans !)



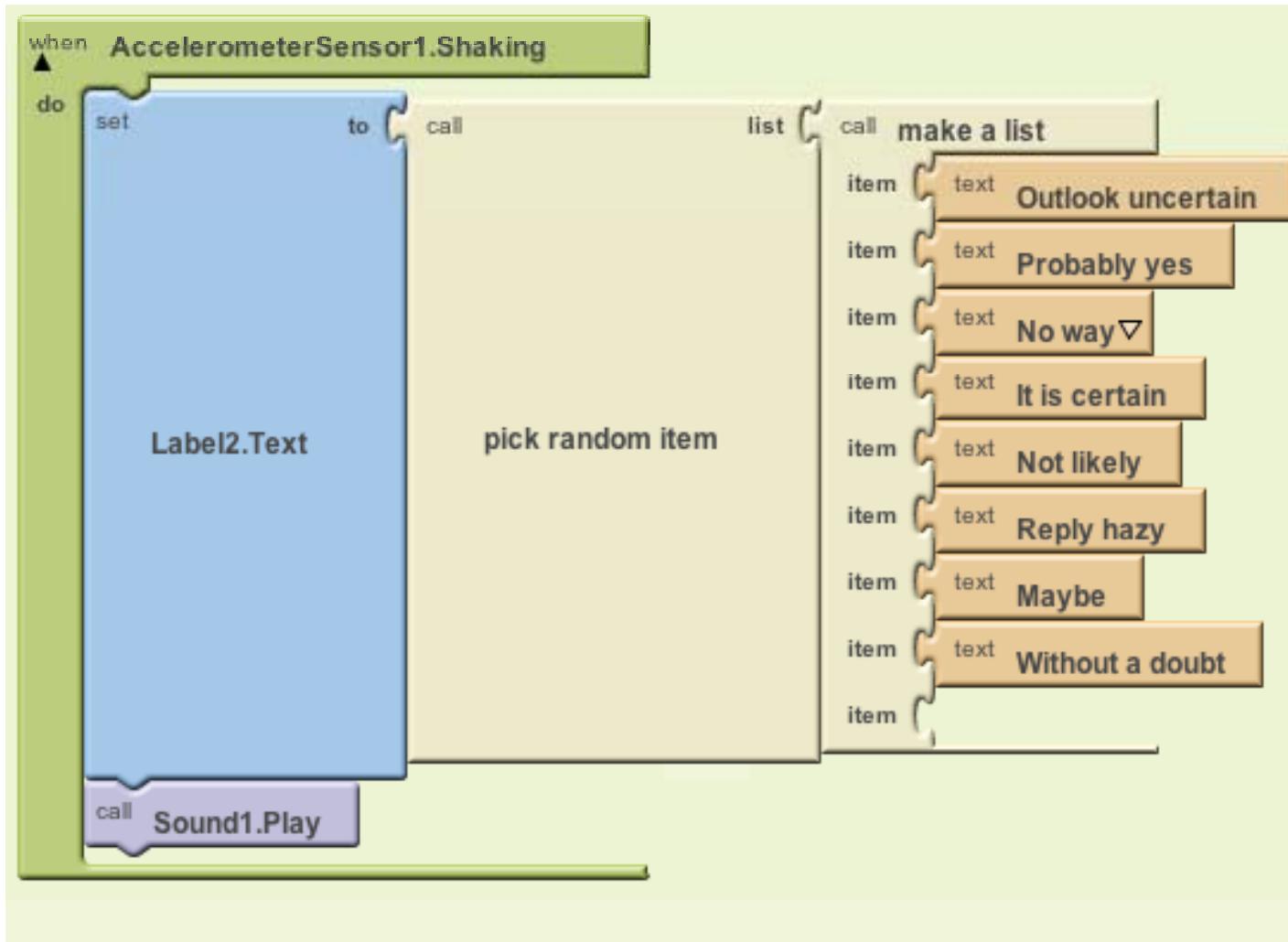


Scratch (à partir de 8 ans !)





App Inventor (à partir de 8 ans ???)





Page de référence

The screenshot shows the Android Developers website at <https://developer.android.com/index.html>. The main feature is the announcement for "Android 7.0 Nougat!" featuring a green Android robot standing on a pink and white textured base. Below the announcement, a red circle highlights the "Get Android Studio" button in the navigation bar.

Android 7.0 Nougat!

Android 7.0 Nougat is here! Get your apps ready for the latest version of Android, with new system behaviors to save battery and memory. Extend your apps with multi-window UI, direct reply notifications and more.

> Learn more

Get Android Studio

Browse sample code

Watch stories

Build Beautiful Apps

Resources to get you started with designing and developing for Android.

<https://developer.android.com/about/versions/nougat/index.html>

<http://developer.android.com/sdk/index.html>



Quelques raccourcis...

- Go to class CTRL + N
- Go to file CTRL + Shift + N
- Navigate open tabs ALT + Left-Arrow; ALT + Right-Arrow
- Look up recent files CTRL + E
- Go to line CTRL + G
- Navigate to last edit location CTRL + SHIFT + BACKSPACE
- **Go to declaration CTRL + B**
- **Go to implementation CTRL + ALT + B**
- Go to source F4
- Go to super Class CTRL + U
- Show Call hierarchy CTRL + ALT + H
- Search in path/project CTRL + SHIFT + F
- **Reformat code CTRL + ALT + L**
- **Optimize imports CTRL + ALT + O**
- Code Completion CTRL + SPACE
- Issue quick fix ALT + ENTER
- **Surround code block CTRL + ALT + T**
- Rename and Refactor Shift + F6
- Line Comment or Uncomment CTRL + /
- **Block Comment or Uncomment CTRL + SHIFT + /**
- Go to previous/next method ALT + UP/DOWN
- Show parameters for method CTRL + P
- Quick documentation lookup CTRL + Q
- Delete a line CTRL + Y
- View declaration in layout CTRL + B

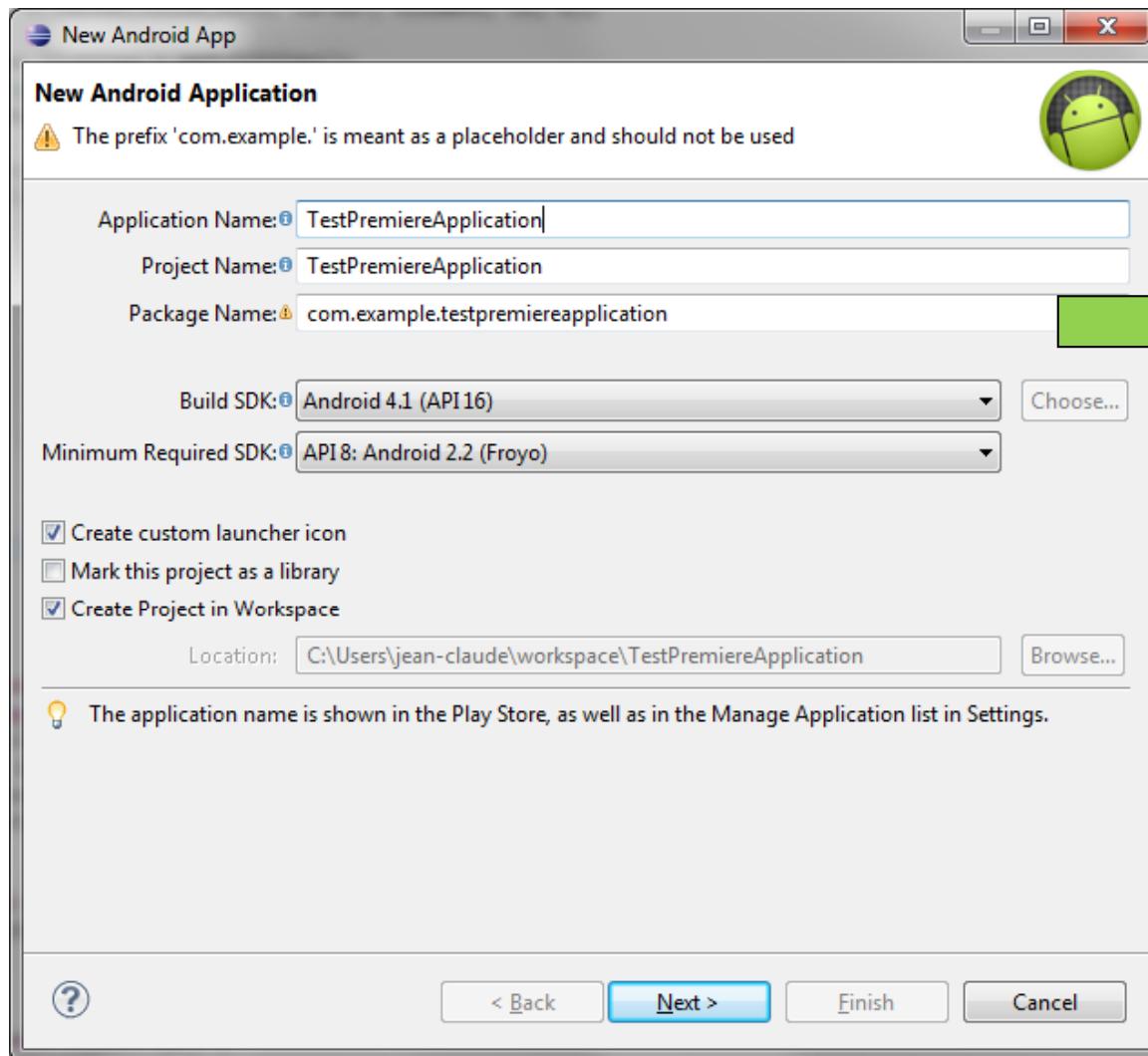


Prêt !

- A présent, le poste est configuré pour créer une première application Android et la tester sur :
 1. l'émulateur
 2. le smartphone
- Avant de passer à la suite, allumez et **réinitialisez** les téléphones !
 - Puis autoriser « sources inconnues »
 - Et « Débogage USB » actif



Première application

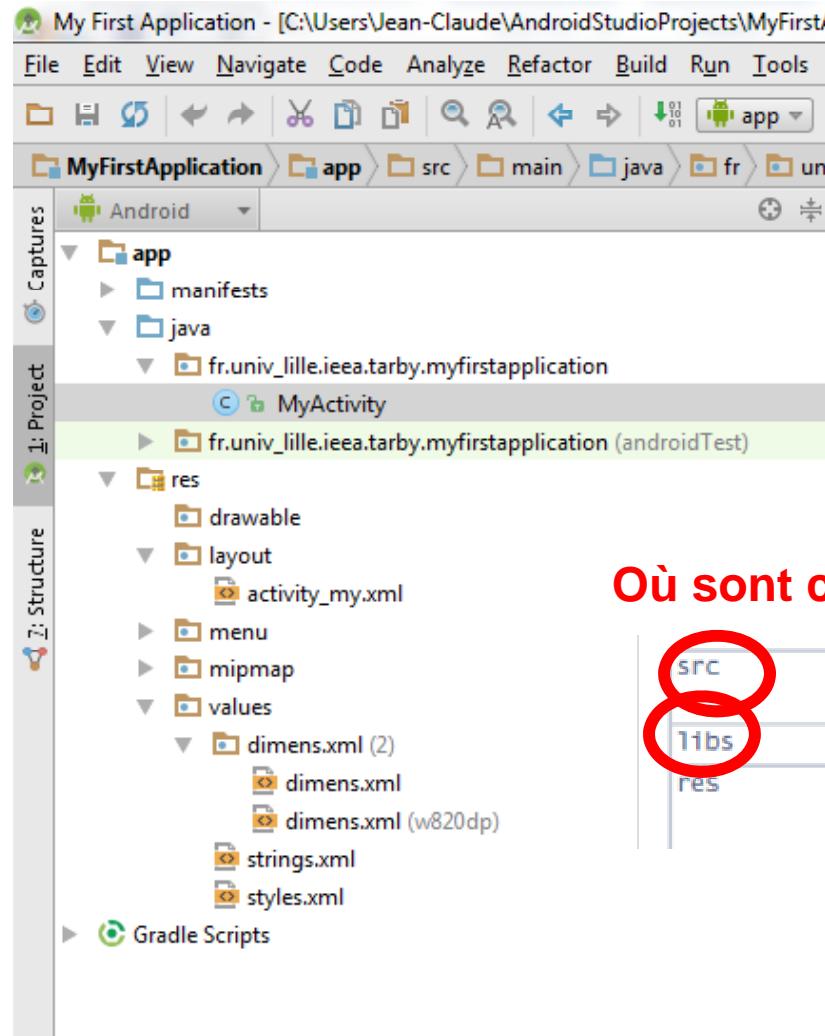


Anciennement
avec ADT

Testons avec
Android Studio...



Première application (Android Studio)



Où sont ces dossiers ?

src	Le répertoire de l'ensemble des sources du projet. C'est dans ce répertoire que vous allez ajouter et modifier le code source de l'application.
libs	Contient les bibliothèques tierces qui serviront à votre application.
res	Contient toutes les ressources telles que les images, les dispositions de l'interface graphique, etc. nécessaires à l'application. Ces ressources seront accessibles grâce à la classe R décrrite plus bas.



Première application (Android Studio)

My First Application - [C:\Users\Jean-Claude\AndroidStudioProjects\MyFirstApplication] - [app] - ...app\src\main\java\fr\univ_lille\ieea\tarby\myfirstapplication\MyActivity.java - Android Studio 1.3

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

MyFirstApplication app src main java fr univ_lille ieea tarby myfirstapplication MyActivity

Android Captures Project Structure

app manifests java fr.univ_lille.ieea.tarby.myfirstapplication MyActivity fr.univ_lille.ieea.tarby.myfirstapplication (androidTest) res drawable layout activity_my.xml menu mipmap values dimens.xml (2) strings.xml styles.xml Gradle Scripts

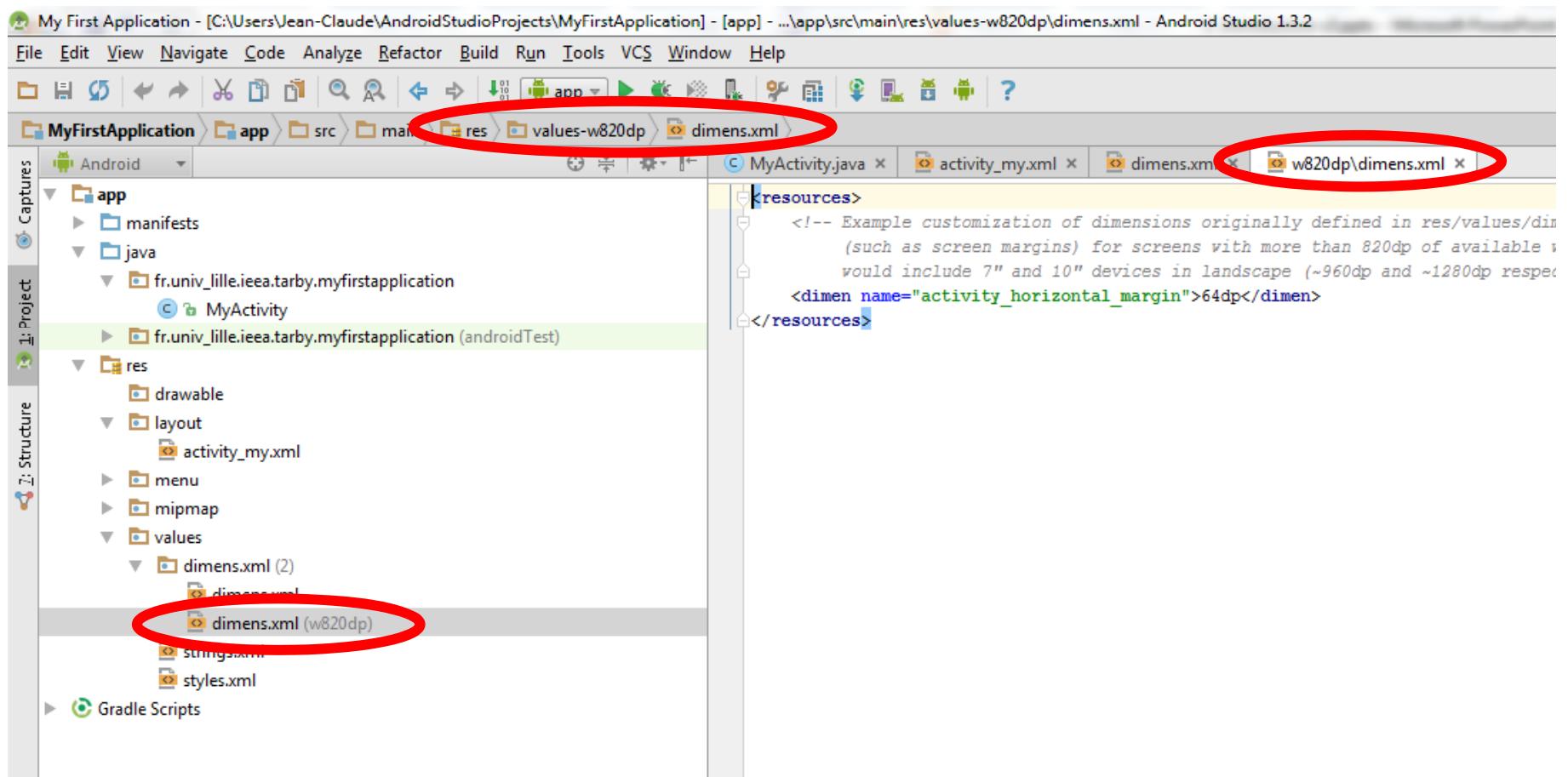
MyActivity.java x activity_my.xml x dimens.xml x w820dp\dimens.xml x

```
package fr.univ_lille.ieea.tarby.myfirstapplication;  
import ...  
  
public class MyActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_my);  
  
        public boolean onCreateOptionsMenu(Menu menu) {  
            // Inflate the menu; this adds items to the action bar if it is present.  
            getMenuInflater().inflate(R.menu.menu_my, menu);  
            return true;  
        }  
  
        @Override  
        public boolean onOptionsItemSelected(MenuItem item) {  
            // Handle action bar item clicks here. The action bar will  
            // automatically handle clicks on the Home/Up button, so long  
            // as you specify a parent activity in AndroidManifest.xml.  
            int id = item.getItemId();
```

Mais pas de dossier SRC visible ici...

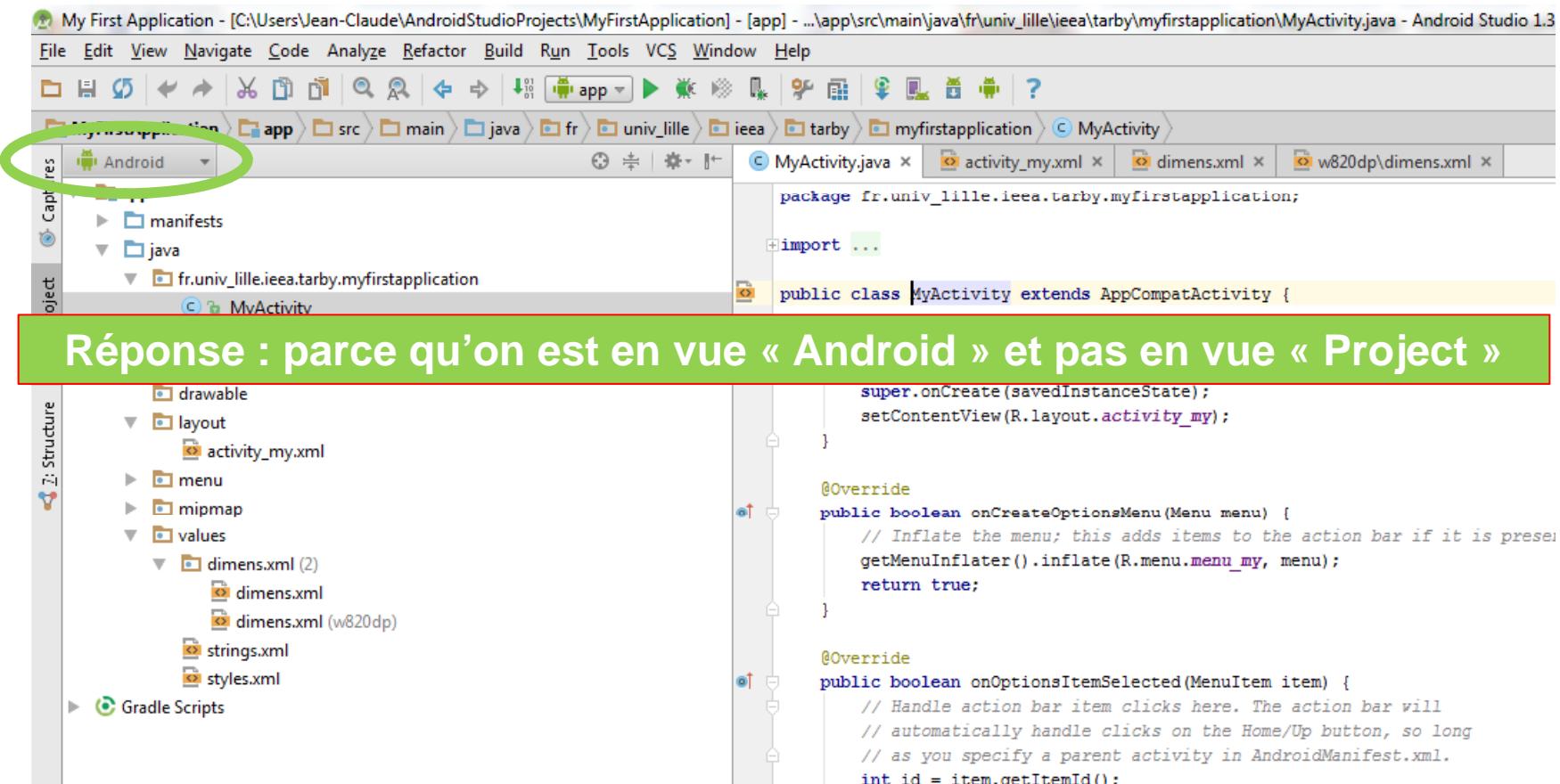


Première application (Android Studio)





Première application (Android Studio)



My First Application - [C:\Users\Jean-Claude\AndroidStudioProjects\MyFirstApplication] - [app] - ...app\src\main\java\fr\univ_lille\ieea\tarby\myfirstapplication\MyActivity.java - Android Studio 1.3

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

My First Application app src main java fr univ_lille ieea tarby myfirstapplication MyActivity

res Android manifests java fr.univ_lille.ieea.tarby.myfirstapplication MyActivity

MyActivity.java activity_my.xml dimens.xml w820dp\dimens.xml

```
package fr.univ_lille.ieea.tarby.myfirstapplication;  
  
import ...  
  
public class MyActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_my);  
    }  
  
    @Override  
    public boolean onCreateOptionsMenu(Menu menu) {  
        // Inflate the menu; this adds items to the action bar if it is present.  
        getMenuInflater().inflate(R.menu.menu_my, menu);  
        return true;  
    }  
  
    @Override  
    public boolean onOptionsItemSelected(MenuItem item) {  
        // Handle action bar item clicks here. The action bar will  
        // automatically handle clicks on the Home/Up button, so long  
        // as you specify a parent activity in AndroidManifest.xml.  
        int id = item.getItemId();  
        ...  
    }  
}
```

Réponse : parce qu'on est en vue « Android » et pas en vue « Project »

Structure

- drawable
- layout
 - activity_my.xml
- menu
- mipmap
- values
 - dimens.xml (2)
 - dimens.xml
 - dimens.xml (w820dp)
 - strings.xml
 - styles.xml
- Gradle Scripts



Première application

- Le répertoire **res** est composé de différents sous-répertoires dont les suivants :
 - **res/drawableXXX** : contient les ressources de type image (PNG, JPEG et GIF) ;
 - **res/layout** : contient les descriptions des interfaces utilisateur ;
 - **res/values** : contient les chaînes de caractères, les dimensions, etc. ;
 - **res/xml** : contient les fichiers XML supplémentaires (préférences, etc.) ;
 - **res/menu** : contient la description des menus ;
 - **res/raw** : contient les ressources autres que celles décrites ci-dessus qui seront empaquetées sans aucun traitement

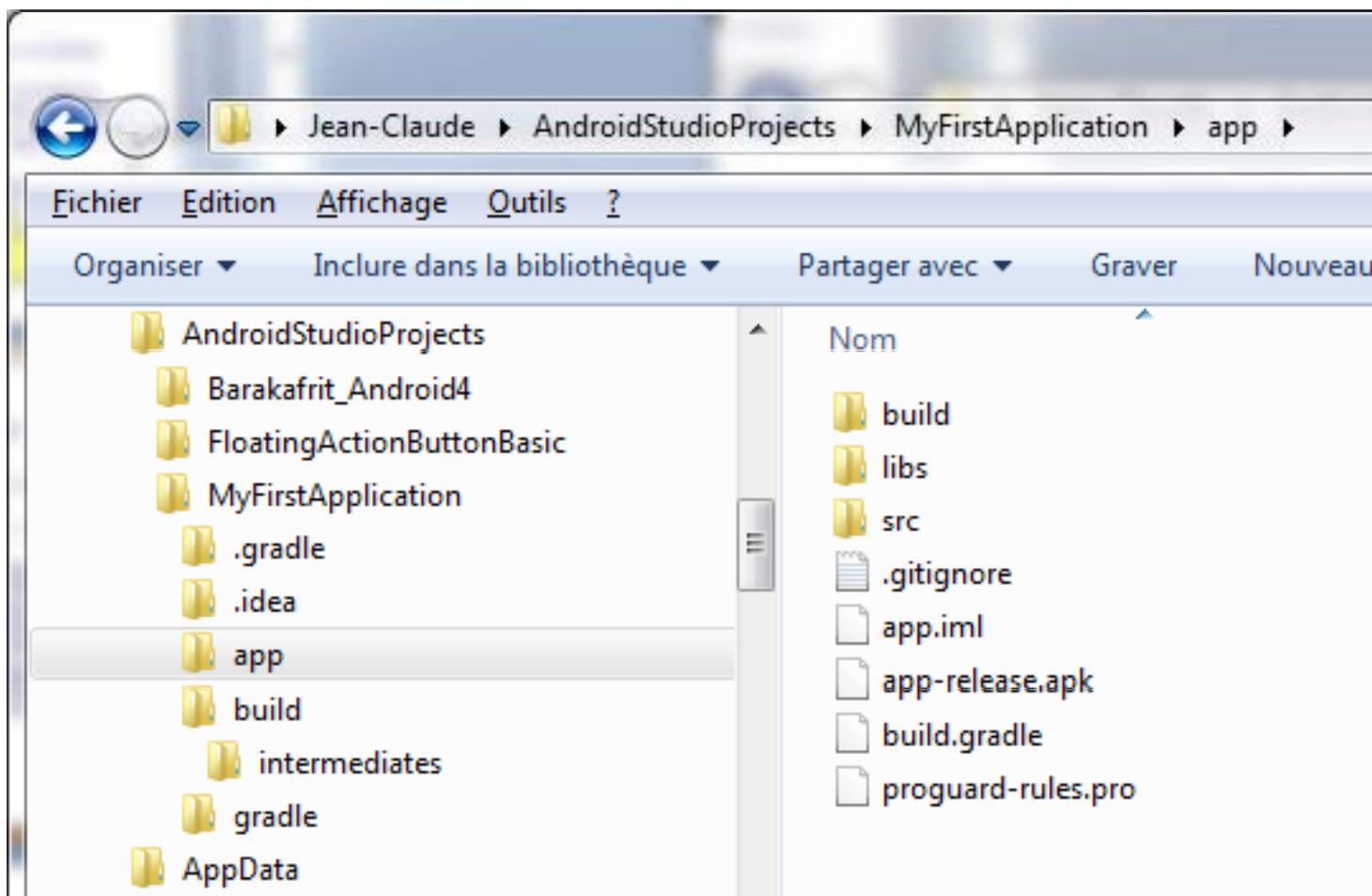


Première application

- Quand vous compilez une application, le résultat sera placé selon l'arborescence suivante :
 - **app/build/intermediates/classes/(votre package)**: les classes compilées en '.class' ;
 - **app/build/outputs/apk** : il s'agit de votre application compilée (mais pas signé !) et empaquetée pour être déployée dans le système Android. Ce fichier contient le code java compilé, les ressources compilées et non compilées (celles contenues dans le répertoire /raw) et enfin le fichier de configuration de l'application.
 - C'est l'apk qu'on donne à quelqu'un pour tester l'application sur un autre téléphone, mais il faut encore le signer numériquement pour le déposer réellement.
 - Signature de l'apk → apk dans dossier des projets Android Studio/MonAppli/app
 - Exemple dans C:\Users\Jean-Claude\AndroidStudioProjects\MyFirstApplication\app (cf. slide suivant)



Première application

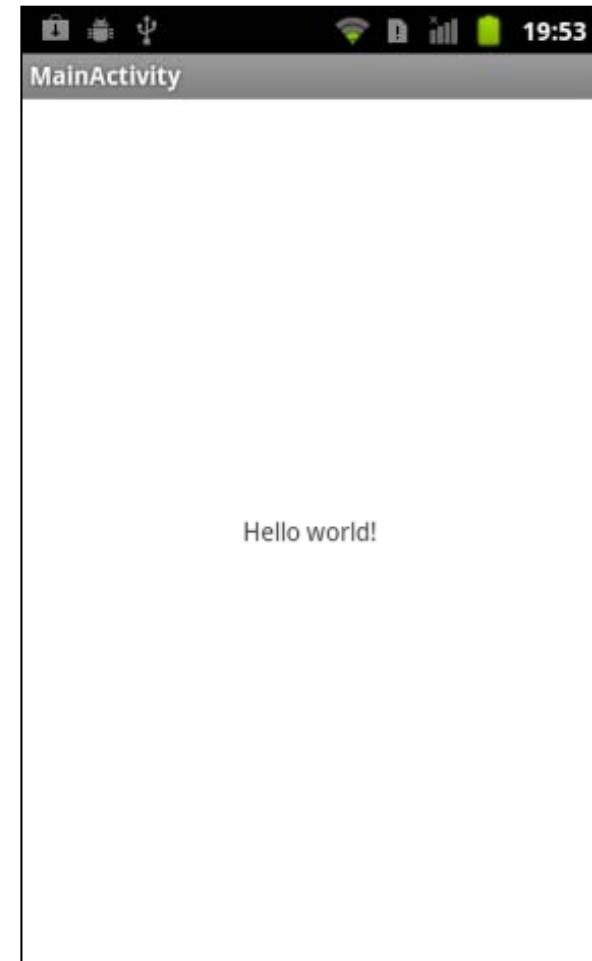




Première application

1. Test de l'application sur l'émulateur

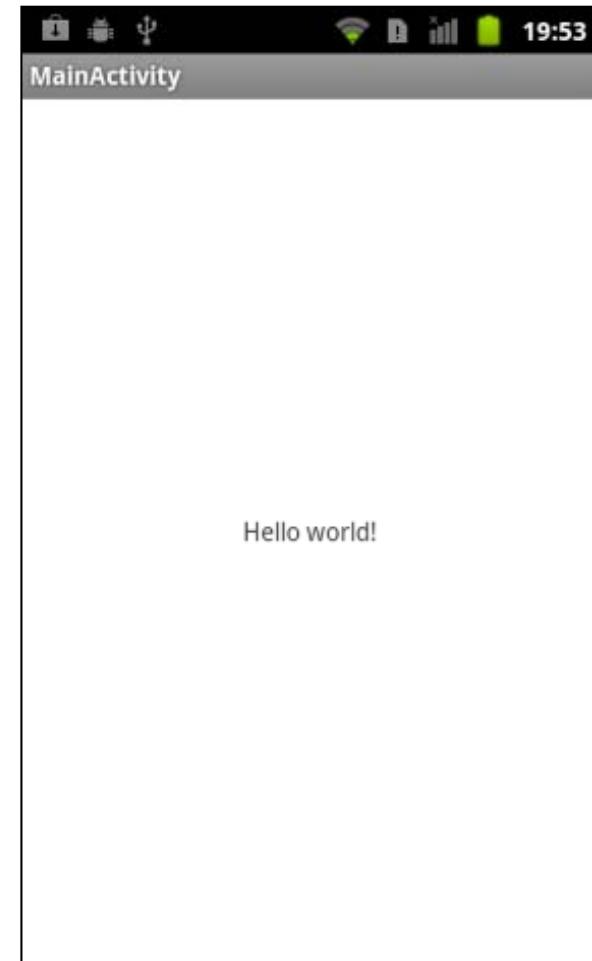
- On doit avoir un device virtuel (AVD) existant (le créer la première fois)
- Long à démarrer la 1^{ère} fois, donc ne pas le fermer chaque fois
- Peut-être des soucis pour l'arrêter ?





Première application

1. Test de l'application sur l'émulateur
 - On doit avoir un device virtuel (AVD) existant (le créer la première fois)
 - Long à démarrer la 1ère fois, donc ne pas le fermer chaque fois
 - Peut-être des soucis pour l'arrêter ?
2. **Test de l'application sur le téléphone connecté**
 - Connecter un téléphone sur le PC (et vérifier que tout est OK) ; vérifier dans Android Studio (onglet Android en bas à gauche) que le téléphone est bien reconnu
 - Lancer l'application (flèche verte dans la barre d'outils)
3. Si tout est OK :
 - (regarder rapidement le dossier generated)
 - l'icône créé sur le téléphone





Première application

- Jetez aussi un œil sur les fichiers « gradle »
 - Dependencies
 - Version du sdk
 - ...
 - Pensez à « mettre votre nez dedans » + tard si problème de compatibilité (par exemple)



Publier sur le Play Store

- Plus facile et rapide que pour iOS ! ☺



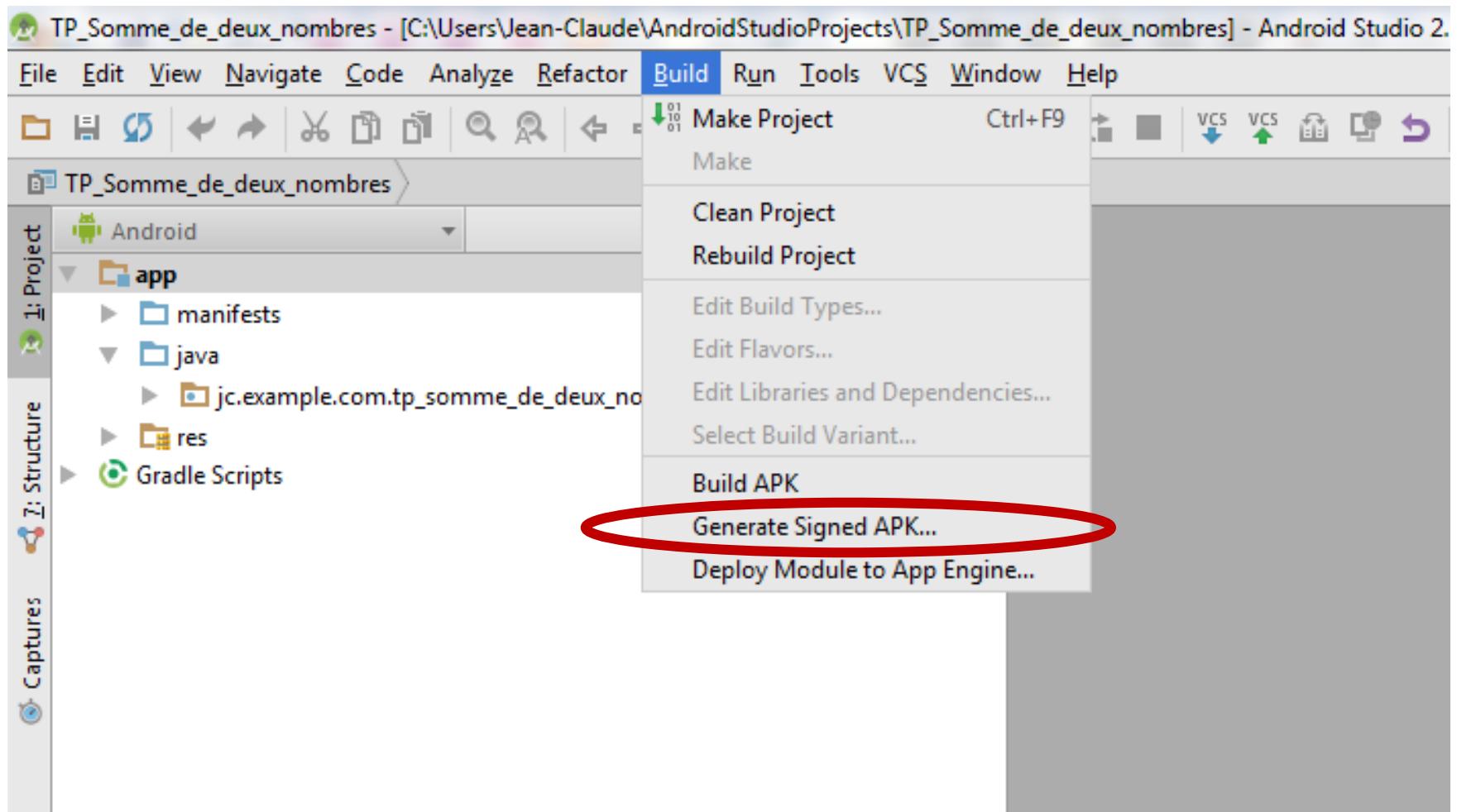


Publier sur le Play Store

1. Créer le projet et le **tester à fond** ! ☺
2. Générer l'APK signé
 - Utiliser (par exemple) l'assistant dans l'onglet du Manifest
 - Ou bien « Export... »
3. Si nécessaire, créer un compte développeur (payant, une seule fois !)
4. Publier...

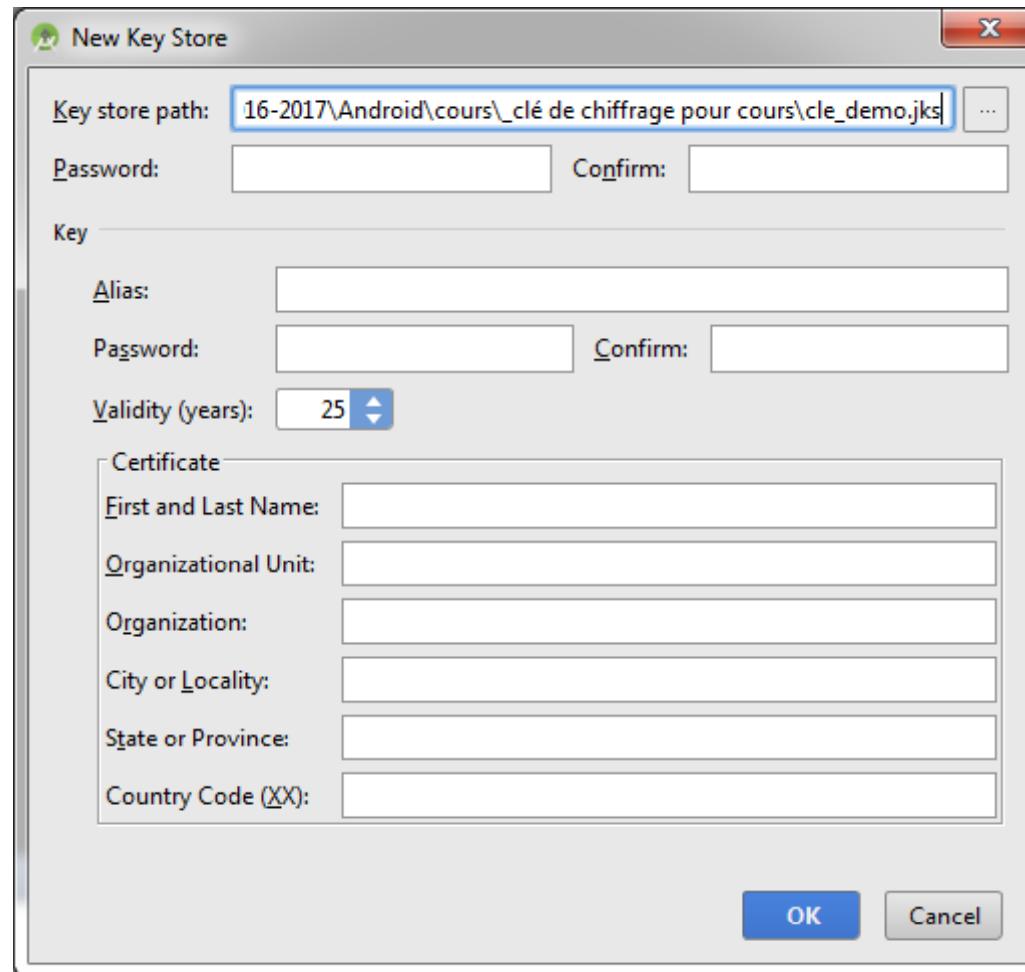


Générer un APK signé





Créer une clé de signature





Créer une clé de signature

- **NE LA PERDEZ SURTOUT PAS ENSUITE**
sinon vous ne pourrez plus jamais mettre à jour votre application !
 - Google ne les stocke pas, vous ne pouvez donc pas la redemander.



APK et fiche

- Déposez ensuite votre APK sur le Play Store et remplissez la fiche associée
 - Ceci peut être fait dans n'importe quel ordre, mais la fiche est obligatoire
 - La fiche doit être remplie correctement
 - N'oubliez pas de spécifier les pays pour votre application !
 - Ensuite, on peut demander à publier l'application
 - Peut être publiée en version alpha, beta, ou en production
- Google ajoute régulièrement de nouveaux services pour le Play Store



Les applications publiées

The screenshot shows the 'Toutes les applications' (All Applications) page in the Google Play Developer Console. The URL in the browser is https://play.google.com/apps/publish/?dev_acc=04946778639877766815&pli=1#AppListPlace. The page displays two published applications:

NOM DE L'APPLICATION	PRIX	INSTALLATIONS ACTUELLES/TOTAL	NOTE MOY./NOMBRE TOTAL	PLANTAGES ET ANR	DERNIÈRE MISE À JOUR	ÉTAT
Barakafrit 2.4.1	Gratuite	715 / 4 442	★ 4,19 / 52	4	1 oct. 2013	Publiée
Boîte à coucou 1.2	Gratuite	—	—	—	10 nov. 2014	Publiée



Fiche d'une application

The screenshot shows the Google Play Developer Console interface. On the left, a sidebar lists various sections: Applications, _mobile, vitrine, important, maisons sur le bon c..., pages ouvertes, Calendrier Lunaire, Connect androphone, Connect androphone, and Autres favoris. The main area displays the details for the app 'BARAKAFRIT' (com.andrilex.barakafrit.activities). The title bar shows the app's icon, name, developer information (Jean-Claude Tarby, Adrien Gooris, Alexandre Wambre), and email (jeanclaude.tarby@gmail.com). A 'Publiée' (Published) status indicator is present. The left sidebar highlights the 'Fiche Google Play Store' section. The main content area is titled 'FICHE GOOGLE PLAY STORE' and shows the 'Enregistré' status. It includes tabs for 'Français – fr-FR' and 'Ajouter des traductions'. The 'INFORMATIONS SUR LE PRODUIT' section contains fields for 'Titre*' (Title), 'Description courte*' (Short Description), and 'Description complète*' (Full Description). The 'Titre*' field contains 'Barakafrit' (10 characters out of 30). The 'Description courte*' field contains 'Trouvez, Notez, Ajoutez, les friteries autour de vous en un clic !' (66 characters out of 80). The 'Description complète*' field contains a detailed description of the app's features, such as finding the nearest barbecue and fries, localizing them on a map, giving notes and comments, and sharing favorite fries with others. The full description is 318 characters long out of 4000. A note at the bottom encourages reading guidelines for creating app descriptions.



Les APK

Fichiers APK - Barakafrit -

https://play.google.com/apps/publish/?dev_acc=0494677863987766815&pli=1#ApkPlace:p=com.andrilex.barakafrit.activities

Applications _mobile _vitrine important maisons sur le bon c... pages ouvertes Calendrier Lunaire - ... Connect androphone Connect androphone Autres favoris

Google play | Developer Console Jean-Claude Tarby, Adrien Gooris, Alexandre Wambre
jeanclaude.tarby@gmail.com Déconnexion ?

BARAKAFRIT - com.andrilex.barakafrit.activities [Afficher sur le Google Play Store](#) ✓ Publié

Statistiques Notes et commentaires Plantages et ANR Conseils d'optimisation

Fichiers APK

PRODUCTION Version **10**

TESTS BÊTA Configurez des tests bêta pour votre application.

TESTS ALPHA Configurez des tests alpha pour votre application.

CONFIGURATION DE LA VERSION EN PRODUCTION Importer un nouveau fichier APK en version production

FICHIER APK ACTUEL date de publication : 14 févr. 2012 04:38:35

Appareils compatibles **5327** **Appareils exclus** **0**

Afficher la liste Gérer les appareils exclus

▼ VERSION	IMPORTÉ LE	ÉTAT	ACTIONS
10 (2.4.1)	-	en production	

AUTRES FICHIERS APK [Masquer](#)

Master Services



Les commentaires

The screenshot shows the Google Play Developer Console interface. The left sidebar has a navigation menu with icons for Applications, Mobile, Vitrine, etc., and the current section is 'Notes et commentaires'. The main content area displays the app's details: BARAKAFRIT (com.andrilex.barakafrit.activities). It shows a green Android icon with a 'NEW' badge, a download link, and a note about CSV export. Below this is a chart of note counts by rating (5 stars: 28, 4 stars: 14, 3 stars: 4, 2 stars: 4, 1 star: 2) and a mean rating of 4,19. The 'AVIS' section shows two reviews: one from Christophe Buyse (Galaxy Note3 Neo) dated 14 oct. 2014 at 16:10, and another from an anonymous user (Version 2.4.1, Galaxy S4) dated 1 juil. 2014 at 14:16. Both reviews mention issues with the app not functioning on certain devices.

Note	Nombre de notes	Nombre d'utilisateurs
5 étoiles	52	28
4 étoiles		14
3 étoiles		4
2 étoiles		4
1 étoile		2



Statistiques de téléchargement

Screenshot of the Google Play Developer Console showing download statistics for the application BARAKAFRIT.

The URL in the browser is https://play.google.com/apps/publish/?dev_acc=04946778639877766815&pli=1#StatsPlace:p=com.andrilex.barakafrit.activities&statm=1.

The sidebar menu shows the following sections:

- Toutes les applications
- Services de jeux
- Rapports financiers
- Paramètres
- Alertes
- Annonces

The main content area displays the following information:

BARAKAFRIT - com.andrilex.barakafrit.activities | [Afficher sur le Google Play Store](#)

STATISTIQUES | **Installations actuelles (appareils)** pour la période 8 oct. 2014 - 8 nov. 2014 | [Exporter au format CSV](#)

Nombre d'appareils actifs sur lesquels l'application est installée actuellement [En savoir plus](#)

Graphique des installations actuelles (appareils) par date:

Version d'Android | Appareil | Tablettes | Pays | Langue | Version | Opérateur

INSTALLATIONS ACTUELLES (APPAREILS) PAR VERSION D'ANDROID

INSTALLATIONS ACTUELLES (APPAREILS) LE 8 NOV. 2014

VOTRE APPLICATION

Version d'Android	Nombre d'appareils	Pourcentage
Android 4.4	198	27,69 %
Android 4.1	136	19,02 %
Android 2.3.3 - 2.3.7	124	17,34 %
Android 4.3	91	12,73 %
Android 4.0.3 - 4.0.4	66	9,23 %
Android 4.2	52	7,27 %
Android 2.2	44	6,15 %
Android 2.1	4	0,56 %

TOUTES LES APPLICATIONS DANS LA CATÉGORIE "VOYAGE ET INFO LOCALE"

Version d'Android	Pourcentage
Android 4.4	28,10 %
Android 2.3.3 - 2.3.7	18,79 %
Android 4.1	17,94 %
Android 4.0.3 - 4.0.4	13,82 %
Android 4.2	10,76 %
Android 4.3	6,36 %
Android 2.2	3,19 %
Android 3.2	0,46 %
Android 2.1	0,34 %
Android 3.1	0,13 %

TOP 10 DES VERSIONS D'ANDROID POUR LA CATÉGORIE "VOYAGE ET INFO LOCALE"

Version d'Android	Pourcentage
Android 4.4	28,10 %
Android 2.3.3 - 2.3.7	18,79 %
Android 4.1	17,94 %
Android 4.0.3 - 4.0.4	13,82 %
Android 4.2	10,76 %
Android 4.3	6,36 %
Android 2.2	3,19 %
Android 3.2	0,46 %
Android 2.1	0,34 %
Android 3.1	0,13 %

DÉCOUVREZ DAVANTAGE DE STATISTIQUES AVEC GOOGLE ANALYTICS POUR LES APPLICATIONS MOBILES



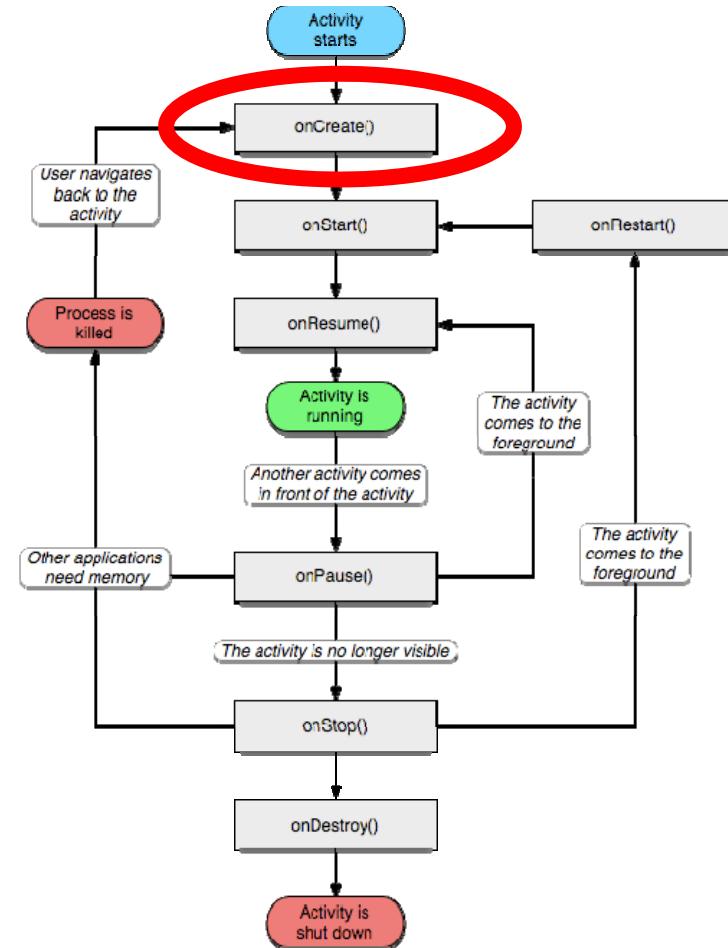
Attention à vos périphériques !





Squelette d'une activité 1/5

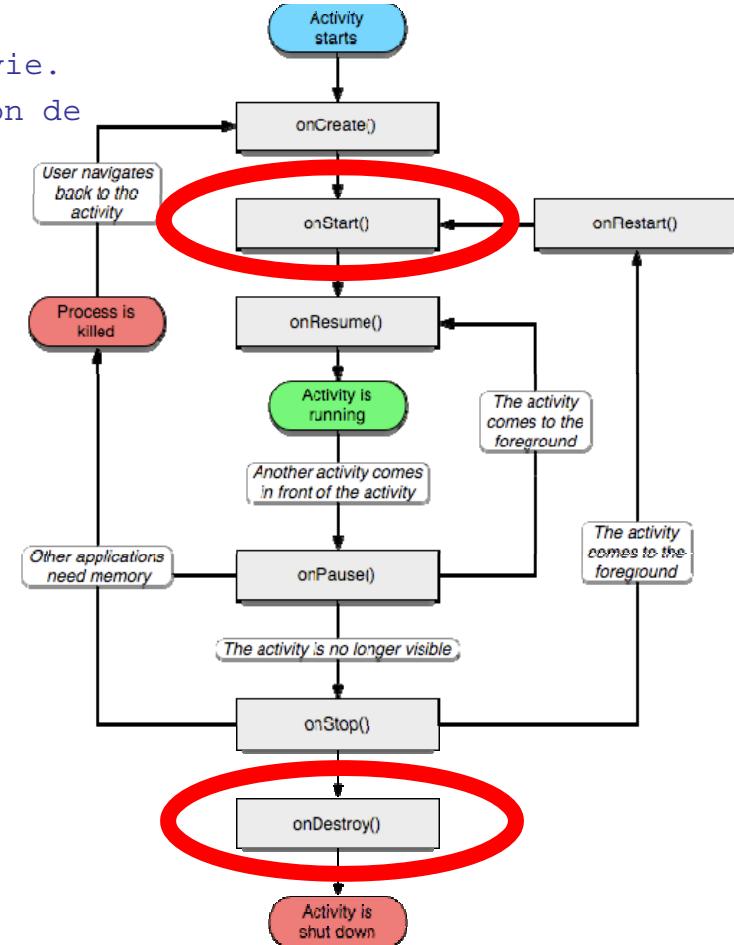
```
package com.eyrolles.android.activity;  
import android.app.Activity;  
import android.os.Bundle;  
  
public final class TemplateActivity extends Activity {  
  
    /**  
     * Appelée lorsque l'activité est créée.  
     * Permet de restaurer l'état de l'interface  
     * utilisateur grâce au paramètre savedInstanceState.  
     */  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        // Placez votre code ici  
    }  
}
```





Squelette d'une activité 2/5

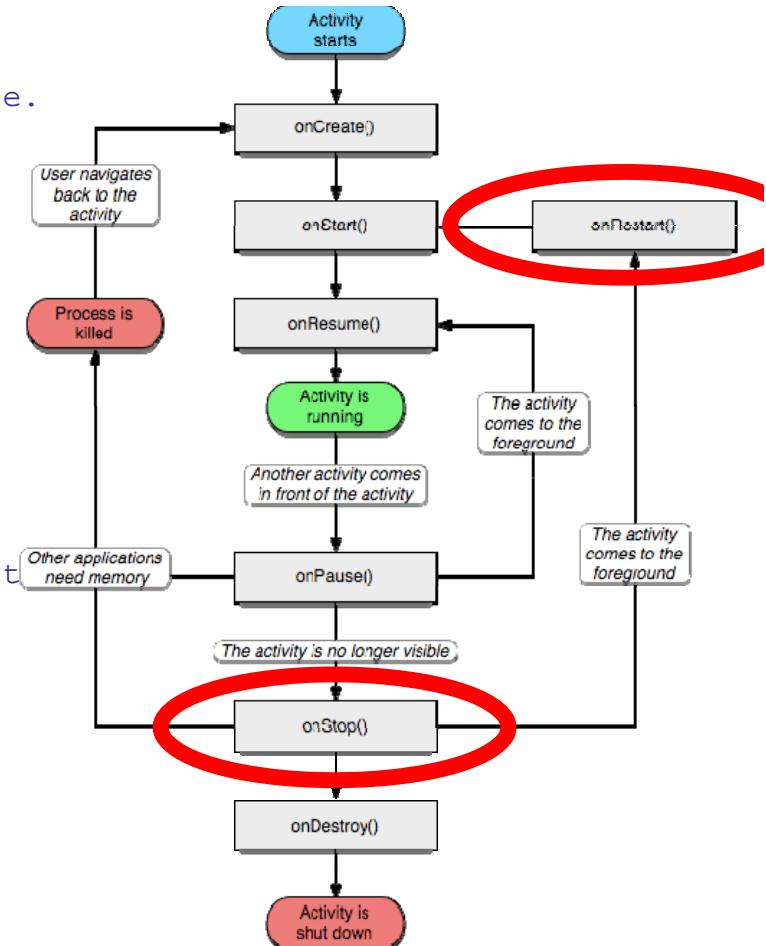
```
/**  
 * Appelée lorsque que l'activité a fini son cycle de vie.  
 * C'est ici que nous placerons notre code de libération de  
 * mémoire, fermeture de fichiers et autres opérations  
 * de "nettoyage".  
 */  
  
@Override  
  
public void onDestroy(){  
    // Placez votre code ici  
    super.onDestroy();  
}  
  
/**  
 * Appelée lorsque l'activité démarre.  
 * Permet d'initialiser les contrôles.  
 */  
  
@Override  
  
public void onStart(){  
    super.onStart();  
    // Placez votre code ici  
}
```





Squelette d'une activité 3/5

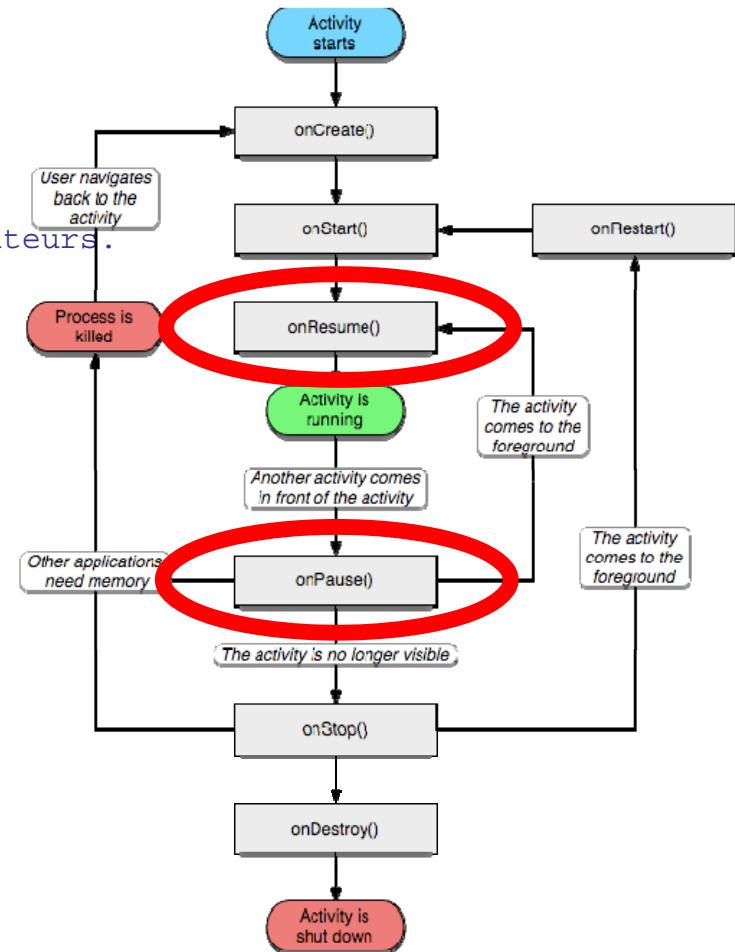
```
/**  
 * Appelée lorsque l'activité sort de son état de veille.  
 */  
  
@Override  
  
public void onRestart(){  
    super.onRestart();  
    //Placez votre code ici  
}  
  
/**  
 * Appelée lorsque l'activité passe en arrière plan.  
 * Libérez les écouteurs, arrêtez les threads, votre act  
 * peut disparaître de la mémoire.  
 */  
  
@Override  
  
public void onStop    // Placez votre code ici  
    super.onStop();  
}
```





Squelette d'une activité 4/5

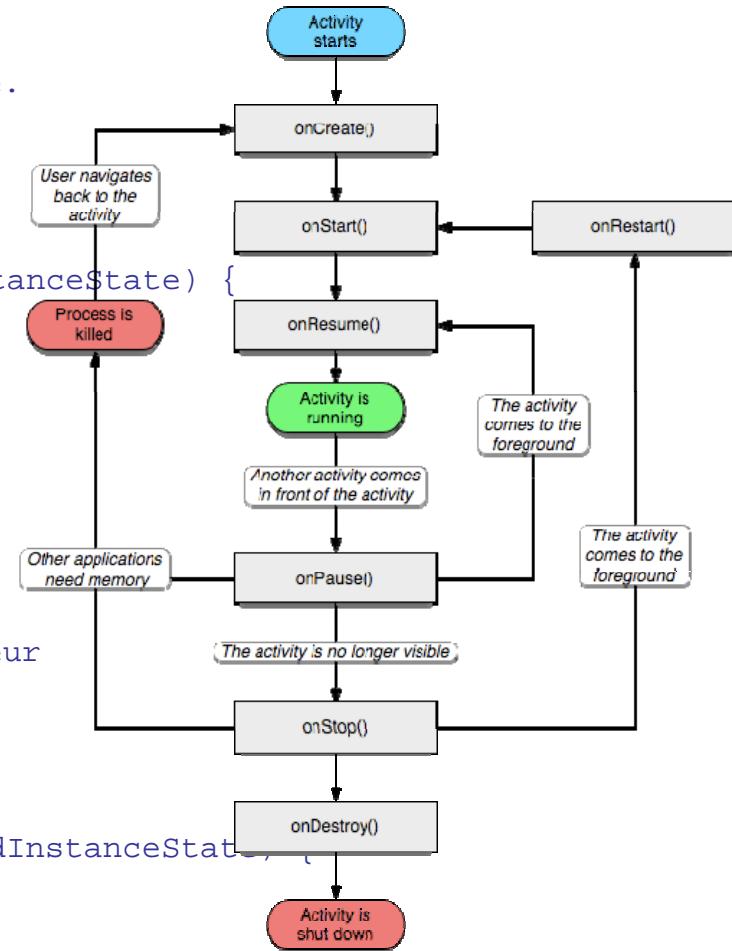
```
/**  
 * Appelée après le démarrage ou une pause.  
 * Relancez les opérations arrêtées (threads).  
 * Mettez à jour votre application et vérifiez vos écouteurs.  
 */  
  
@Override  
  
public void onResume(){  
super.onResume();  
// Placez votre code ici  
}  
  
/**  
 * Appelée lorsque que l'activité est suspendue.  
 * Stoppez les actions qui consomment des ressources.  
 * L'activité va passer en arrière-plan.  
 */  
  
@Override  
  
public void onPause(){  
//Placez votre code ici  
super.onPause();  
}
```





Squelette d'une activité 5/5

```
/**  
 * Appelée lorsque l'activité termine son cycle visible.  
 * Sauvez les données importantes.  
 */  
  
@Override  
  
public void onSaveInstanceState(Bundle savedInstanceState)  
// Placez votre code ici  
// sans quoi l'activité aura perdu son état  
// lors de son réveil  
super.onSaveInstanceState(savedInstanceState);  
}  
/**  
 * Appelée après onCreate.  
 * Les données sont rechargées et l'interface utilisateur  
 * est restaurée dans le bon état.  
 */  
  
@Override  
  
public void onRestoreInstanceState(Bundle savedInstanceState)  
super.onRestoreInstanceState(savedInstanceState);  
//Placez votre code ici  
}
```





Testez !

- Cf. projet Demo_cycledevie_Activity
 - https://gitlab.com/m2eservices/Demo_CycleDeVie_Activity.git
- Lancez l'appli et une fois lancée, tournez l'écran !



Ressources

- Pour accéder aux ressources, il suffit de connaître leur type et leur identifiant
([android.]R.type_de_ressource.nom_ressource).
- Exemple **OK** :

```
// Fixe la mise en page d'une activité
setContentView(R.layout.ecran_de_demarrage);
```
- Exemple **pas OK** car **on renvoie toujours un ID** et pas la string attendue !

```
// Création par copie d'une chaîne de caractères
String titre = new String(R.string.texte_titre_ecran);
```
- **Solution** :

```
Resources resources = getResources();
String nom = resources.getString(R.string.texte_titre_ecran);
```



Ressources référencées par d'autres ressources

- Vous pouvez également utiliser vos ressources comme valeurs d'attributs dans d'autres ressources sous forme XML.
- Cette possibilité est très utilisée dans les mises en page par exemple.
 - Texte affiché
 - Style utilisé
 - Dimension utilisée
 - Couleur utilisée...
- La notation pour faire référence à une autre ressource est la suivante :
`attribute="@[package_name:]resource_type/resource_identifier"`
- Exemple :
`<TextView android:text="@string/table_contenu_cellule_gauche" />`
- Vous pouvez aussi écrire vos propres ressources
 - Par exemple pour définir des configurations différentes de mise en page



Utilisation de ressources systèmes

- Il suffit d'utiliser la classe **android.R**.
 - Exemple : `android.R.drawable.ic_dialog_alert`
- Pour accéder à ces ressources dans un fichier XML, il faut spécifier « android » comme espace de nommage.
 - Exemple : `<... android:text="@android:string/unknownName"/>`



Créer des ressources

- Par convention, on sépare les types de ressources, par exemple res/values/**strings.xml** pour les **strings** etc.
- Exemple (strings.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="nom_application">Suivi des SMS</string>
    <string name="auteur_application">JCT</string>
</resources>
```
- Rappel : Internationalisation « facile »
 - Créer des dossiers pour chaque langue et Android se charge du reste



Définir des couleurs

- Une couleur définit une valeur RVB (rouge, vert et bleu) et une transparence.
- Il existe différents formats dont la syntaxe globale est la suivante :
`<color name=nom_couleur>valeur_de_la_couleur</color>`
- Les différents formats d'une couleur sont les suivants :
 - #RGB (Rouge-Vert-Bleu/valeur hexadécimale sur 1 caractère [0,16])
 - #ARGB (Alpha (transparence)-Rouge-Vert-Bleu)
 - #RRGGBB (Rouge-Vert-Bleu/valeur hexadécimale sur 2 caractères [0,255])
 - #AARRGGBB

```
<resources>
    <color name="bleu_transparent">#50FF00FF</color>
</resources>
```

- Exemple d'utilisation des couleurs :
 - En Java : R.color.bleu_transparent
 - En XML : @[package:]color/bleu_transparent



Définir des chaînes de caractères

- Syntaxe :

```
<string name=nom_chaine>valeur_de_la_chaine</string>
```

- Ce format permet d'utiliser trois balises HTML standard ****, **<i>** et **<u>**

```
<string name="ex1">Un texte <b>mis en forme</b></string>
```
- Note : si vous utilisez des guillemets ou des apostrophes, vous devez les 'échapper' en les faisant précéder du caractère slash ('\').

```
<resources>
    <string name="app_name">Exemple Android</string>
    <string name="menu_principal">Menu Principal</string>
</resources>
```

- Utilisation des chaînes de caractères :
 - Java : R.string.le_nom
 - XML : @[package:]string/le_nom

Pensez à convertir vos chaines de caractères en ressources au fur et à mesure de votre développement



Définir des unités de mesure

- Les unités prises en charge par Android sont:
 - px (pixels), in (pouces), mm (millimètres), pt (points), dp («density-independant» pixel), sp («scale-independant pixel»).

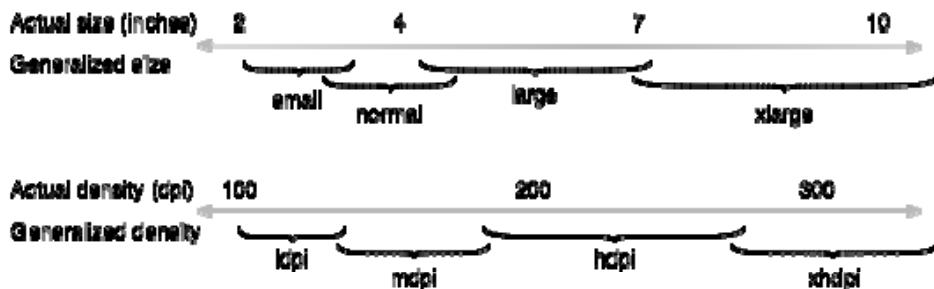
```
<resources>
    <dimen name="taille_texte">5sp</dimen>
</resources>
```

- Utilisation des dimensions :
 - Java : R.dimen.un_nom
Exemple : Resources.getDimen(R.**dimen.taille_texte**);
 - XML : @[package:]dimen/un_nom
`<TextView android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:textSize="@dimen/taille_texte" />`



Autres ressources

- Ressources images : PNG, JPG et GIF
 - Java : R.drawable.fichier_bitmap
 - XML : @[package:]drawable/fichier_bitmap
- Balise <supports-screens> qui grâce aux attributs android:smallScreens, android:normalScreens et android:largeScreens permet de spécifier quelle(s) taille(s) d'écran votre application supporte.



- *xlarge* screens are at least 960dp x 720dp
- *large* screens are at least 640dp x 480dp
- *normal* screens are at least 470dp x 320dp
- *small* screens are at least 426dp x 320dp

« deprecated »...
Pensez aux « swxxx »...

Cf.
http://developer.android.com/guide/practices/screens_support.html pour gérer des tailles d'écrans différentes de façon efficace

- Autres ressources : menu (définis en XML), layouts (en XML)



Autres ressources

- Ressources images : PNG, JPG et GIF
 - Java : R.drawable.fichier_bitmap
 - XML : @[package:]drawable/fichier_bitmap
- **Note:** Place all your launcher icons in the **res/mipmap-[density]/ folders**, rather than the **res/drawable-[density]/ folders**. The Android system retains the resources in these density-specific folders, such as mipmap-xxxhdpi, regardless of the screen resolution of the device where your app is installed. This behavior allows launcher apps to pick the best resolution icon for your app to display on the home screen. For more information about using the mipmap folders, see [Managing Projects Overview](#).

For example, the following application resource directories provide different layout designs for different screen sizes and different drawables. Use the `mipmap/` folders for launcher icons.

```
res/layout/my_layout.xml           // layout for normal screen size ("default")
res/layout-large/my_layout.xml     // layout for large screen size
res/layout-xlarge/my_layout.xml   // layout for extra-large screen size
res/layout-xlarge-land/my_layout.xml // layout for extra-large in landscape orientation

res/drawable-mdpi/graphic.png     // bitmap for medium-density
res/drawable-hdpi/graphic.png    // bitmap for high-density
res/drawable-xhdpi/graphic.png   // bitmap for extra-high-density
res/drawable-xxhdpi/graphic.png  // bitmap for extra-extra-high-density

res/mipmap-mdpi/my_icon.png       // launcher icon for medium-density
res/mipmap-hdpi/my_icon.png      // launcher icon for high-density
res/mipmap-xhdpi/my_icon.png     // launcher icon for extra-high-density
res/mipmap-xxhdpi/my_icon.png    // launcher icon for extra-extra-high-density
res/mipmap-xxxhdpi/my_icon.png   // launcher icon for extra-extra-extra-high-density
```



Autres ressources

- On peut définir à la volée des attributs (**très très utilisé**)
 - exemple : un « id » auquel on fera référence + tard.

```
<TextView android:id="@+id/monText" />
```

Utilisation : **R.id.monText**

```
TextView monTexte =  
    (TextView) findViewById(R.id.monText);  
monTexte.setText("Coucou c'est moi !");
```



Manifest.xml

- ~~ADT : Fichier placé dans le répertoire de base du projet, à sa racine.~~
- Android Studio : Fichier placé dans app/src/main du projet.
 - Note : pas toujours vrai avec l'importation de projets ADT dans Android Studio !
 - Plusieurs Manifest ou un seul « mergé » (dépend des paramètres d'importation)
- Il décrit le contexte de l'application, les activités, les services, les récepteurs d'Intents (Broadcast receivers), les fournisseurs de contenu et les permissions.
- ~~Peut être écrit à la main ou par éditeur dans ADT.~~



Structure de Manifest

Détails à
<http://developer.android.com/guide/topics/manifest/manifest-intro.html>

Cf. slide suivant

```
<?xml version="1.0" encoding="utf-8"?>
<manifest>
    <uses-permission />_
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <application>_
        <activity>_
            <intent-filter>
                <action />
                <category />
                <data />
            </intent-filter>
            <meta-data />
        </activity>
        <activity-alias>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </activity-alias>
        <service>_
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </service>
        <receiver>_
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </receiver>
        <provider>_
            <grant-uri-permission />
            <path-permission />
            <meta-data />
        </provider>
        <uses-library />
    </application>
</manifest>
```



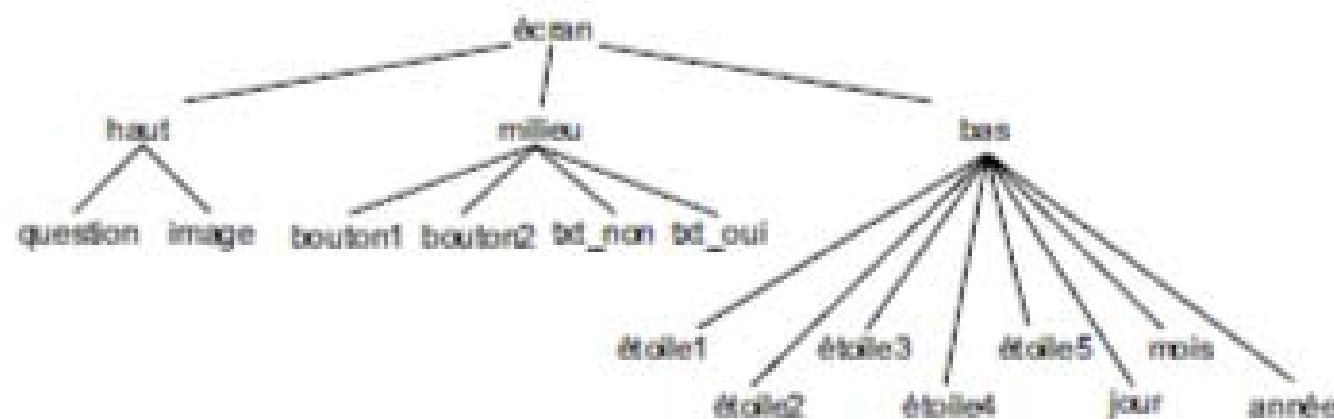
Structure de Manifest

- <uses-permission>
 - Les permissions qui seront déclarées ici seront un prérequis pour l'application. À l'installation, l'utilisateur se verra demander l'autorisation d'utiliser l'ensemble des fonctions liées à ces permissions comme la connexion réseau, la localisation de l'appareil, les droits d'écriture sur la carte mémoire...
- <application>
 - **Un manifeste contient un seul et unique nœud application qui en revanche contient des noeuds** concernant la définition d'activités, de services...
- <activity>
 - Déclare une activité présentée à l'utilisateur. **Si vous oubliez ces lignes de configuration, vos éléments ne pourront pas être utilisés.**
- <service>
 - Déclare un composant de l'application en tant que service.
- <receiver>
 - **Déclare un récepteur d'objets Intent.** Cet élément permet à l'application de recevoir ces objets alors qu'ils sont diffusés par d'autres applications ou par le système.
- <provider>
 - **Déclare un fournisseur de contenu qui permettra d'accéder aux données gérées par l'application.**

Cf. MAJ Android 6 !



IHM





Layout

- Une application utilise le layout créé soit en XML soit en Java :

```
//setContentView(R.layout.main);  
LinearLayout layout = new LinearLayout(this);  
TextView text = new TextView(this);  
text.setText(R.string.hello);  
layout.addView(text);  
setContentView(layout);
```

- mais XML + facile à gérer, à réutiliser, et permet le multilingue



Layout

- Les vues (c'est-à-dire tous les composants graphiques) héritent de View,
- Les vues peuvent être regroupées dans des ViewGroup.
 - De fait, les layouts héritent aussi de ViewGroup
- Les layout sont définis en XML (en général) dans res/layout.
 - Noms uniquement avec des minuscules et des lettres !
- ViewGroup (quelques uns...)
 - **LinearLayout**
 - les éléments sont alignés de gauche à droite ou de haut en bas (propriété orientation);
 - **RelativeLayout**
 - les enfants sont positionnés les uns par rapport aux autres, le premier enfant servant de référence aux autres ; tous les éléments doivent avoir un id.
 - **FrameLayout**
 - le plus basique des gabarits. Chaque enfant est positionné dans le coin en haut à gauche de l'écran et affiché par-dessus les enfants précédents, les cachant en partie ou complètement. Ce gabarit est principalement utilisé pour l'affichage d'un élément (par exemple, un cadre dans lequel on veut charger des images) ;
 - **TableLayout**
 - permet de positionner en lignes et colonnes à l'instar d'un tableau.
 - ...
- On peut aussi imbriquer des ViewGroup, faire des include, etc.
- Les layouts possèdent des attributs (height/width...) mais aussi **fill_parent** (remplacé par **match_parent** depuis API 8=Android 2.3) et **wrap_content**.
 - Fill/Match = remplit toute la place disponible,
 - Wrap = ne prend que ce qui est nécessaire en hauteur/largeur.



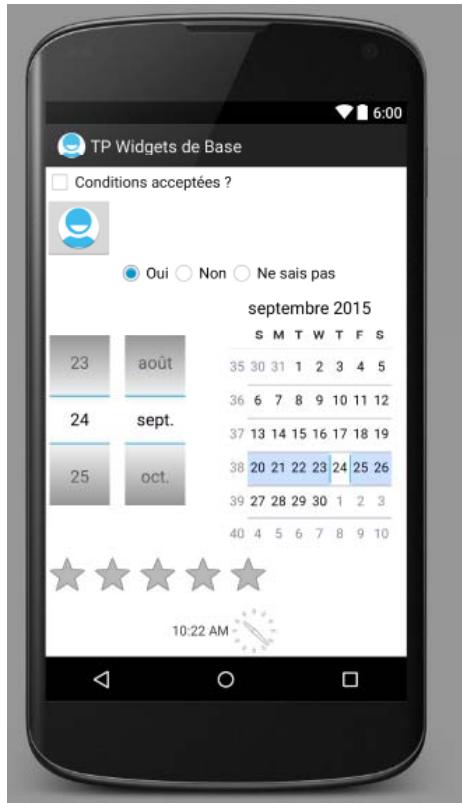
Unités de mesure

- Unités prises en charge par Android :
 - **pixel (px)** : correspond à un pixel de l'écran ;
 - **pouce (in)** : unité de longueur, correspondant à 2,54 cm. Basé sur la taille physique de l'écran ;
 - **millimètre (mm)** : basé sur la taille physique de l'écran ;
 - **point (pt)** : 1/72 d'un pouce ;
 - **pixel à densité indépendante (dp ou dip)** : une unité relative se basant sur une taille physique de l'écran de 160 dpi.
 - Avec cette unité, 1 dp est égal à 1 pixel sur un écran de 160 pixels.
 - Si la taille de l'écran est différente de 160 pixels, les vues s'adapteront selon le ratio entre la taille en pixels de l'écran de l'utilisateur et la référence des 160 pixels ;
 - **pixel à taille indépendante (sp)** : fonctionne de la même manière que les pixels à densité indépendante à l'exception qu'ils sont aussi fonction de la taille de polices spécifiée par l'utilisateur. **Il est recommandé d'utiliser cette unité lorsque vous spécifiez les tailles des polices.**
 - **Ces deux dernières unités sont à privilégier car elles permettent de s'adapter plus aisément à différentes tailles d'écran et rendent ainsi vos applications plus portables.** Notez que ces unités sont basées sur la taille physique de l'écran : l'écran ne pouvant afficher une longueur plus petite que le pixel, ces unités seront toujours rapportées aux pixels lors de l'affichage (1 cm peut ne pas faire 1 cm sur l'écran selon la définition de ce dernier).
- Si vous avez besoin d'intégrer une image avec une taille précise, préférez les valeurs en dip à celles en px.

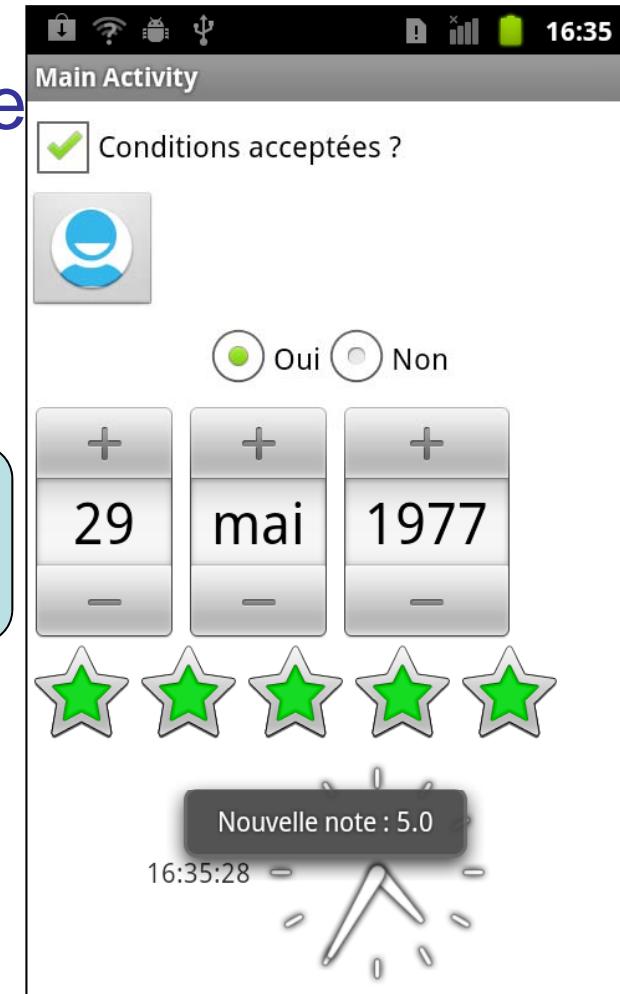


Exemple de widgets

- Cf. projet TP_Widgets_de_base
 - Nous verrons le code + tard



Affichage sur Galaxy
Tab 1



Affichage dans
Android Studio



Quelques exemples de widgets

<TextView propriétés />

= label

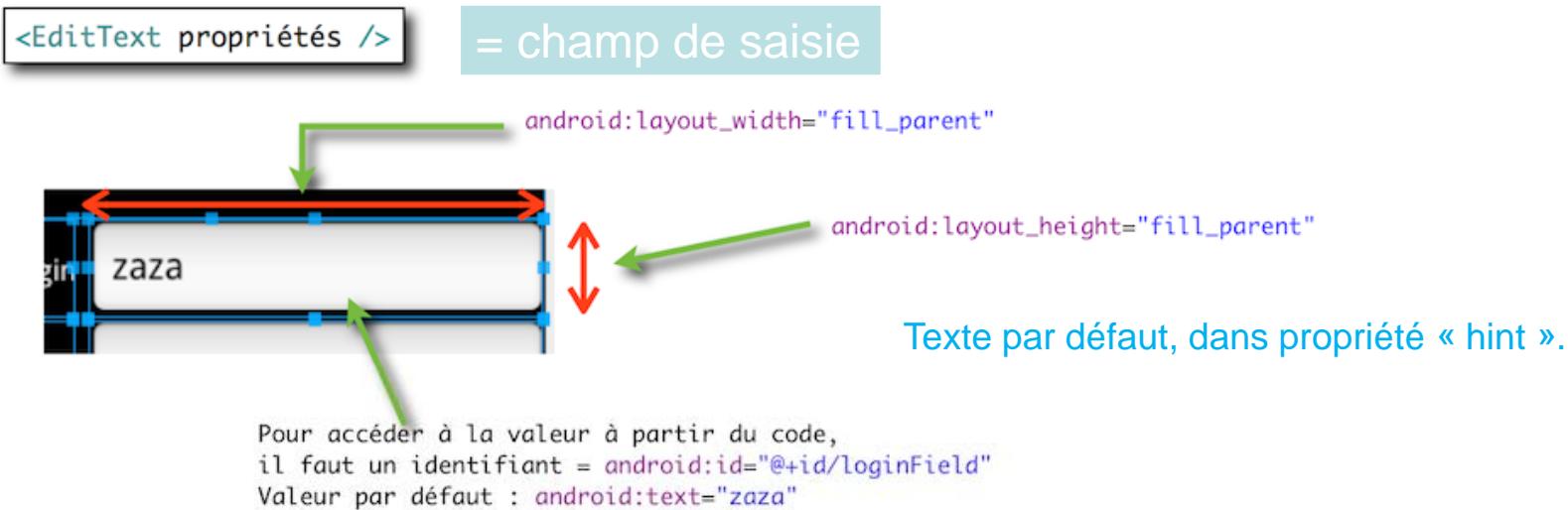
android:text="Login"
centré et justifié à droite = android:gravity="center_vertical|right"
texte en blanc = android:textColor="#FFFFFF"



```
final TextView titre = (TextView)findViewById(R.id.id_vue);
titre.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        ...
    });
}
```



Quelques exemples de widgets



// Récupérer

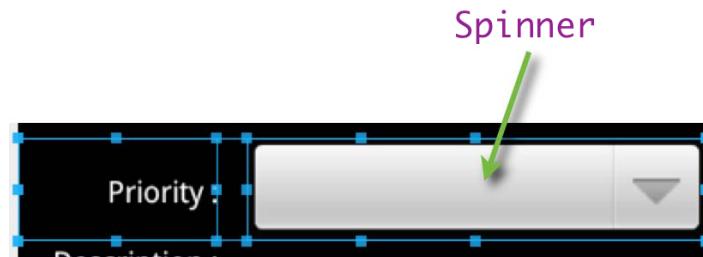
```
String texte=editText.getText()+"";
// comme le texte des vues est de type CharSequence
// il faut le convertir en String
```

// Fixer

```
editText.setText("bonjour");
```



Quelques exemples de widgets



Définir la liste de choix possibles en créant un tableau de String dans strings.xml ou dans arrays.xml :

```
<string-array name="mesChoix">
    <item>Choix 1</item>
    <item>Choix 2</item>
    <item>Choix 3</item>
</string-array>
```

et faire référence à ce tableau via la propriété **android:entries**

Ex : `android:entries="@array/mesChoix"`

Vous pouvez aussi fixer le prompt du spinner via sa propriété **android:prompt**.

Pour accéder en Java à la position correspondant au choix de l'utilisation > **getSelection()** qui renvoie un entier.

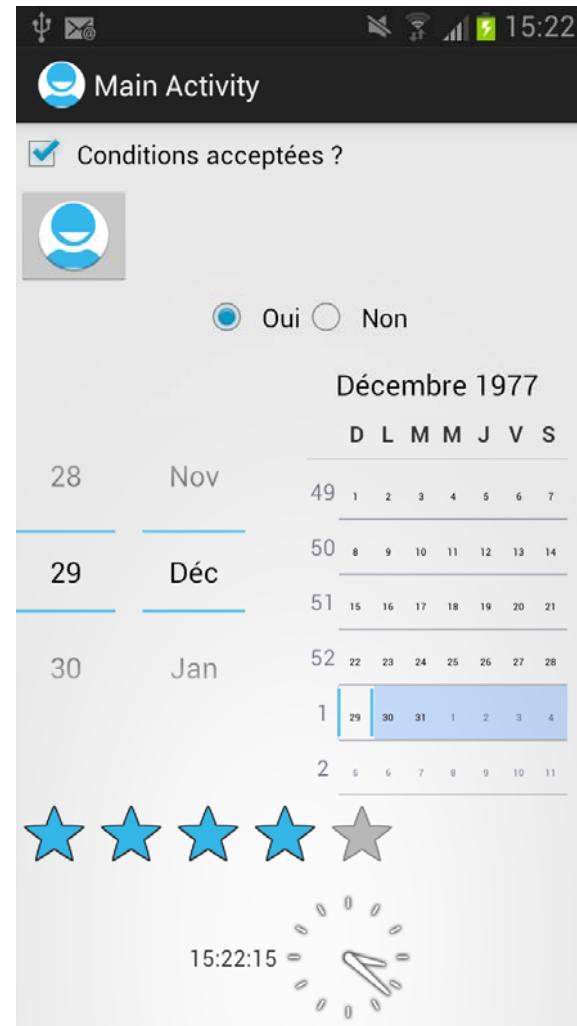
Pour positionner la liste sur un choix/item particulier > **setSelection (position-entier)**.



Attention aux nouveaux widgets !



Certains widgets ne sont pas compatibles
(par ex. le TextClock dans « other »), ou ne
fonctionnent plus de la même façon (par
exemple le DatePicker)





Layout

- Faites des essais par vous-mêmes maintenant
 - Faites des mises en page pour jouer avec les widgets
 - Faites des mises en page que vous aurez faites sur papier avant
 - Dans tous les cas, essayez ensuite de modifier ces mises en page...



Clic sur un bouton

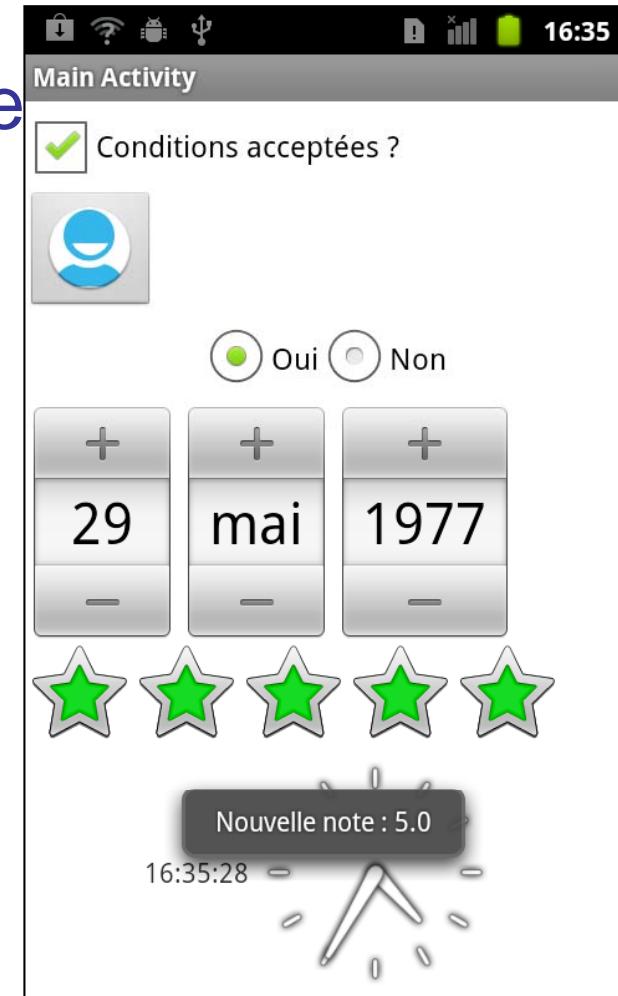
- onClick dans XML
- Listener dans Java



Exemple de widgets

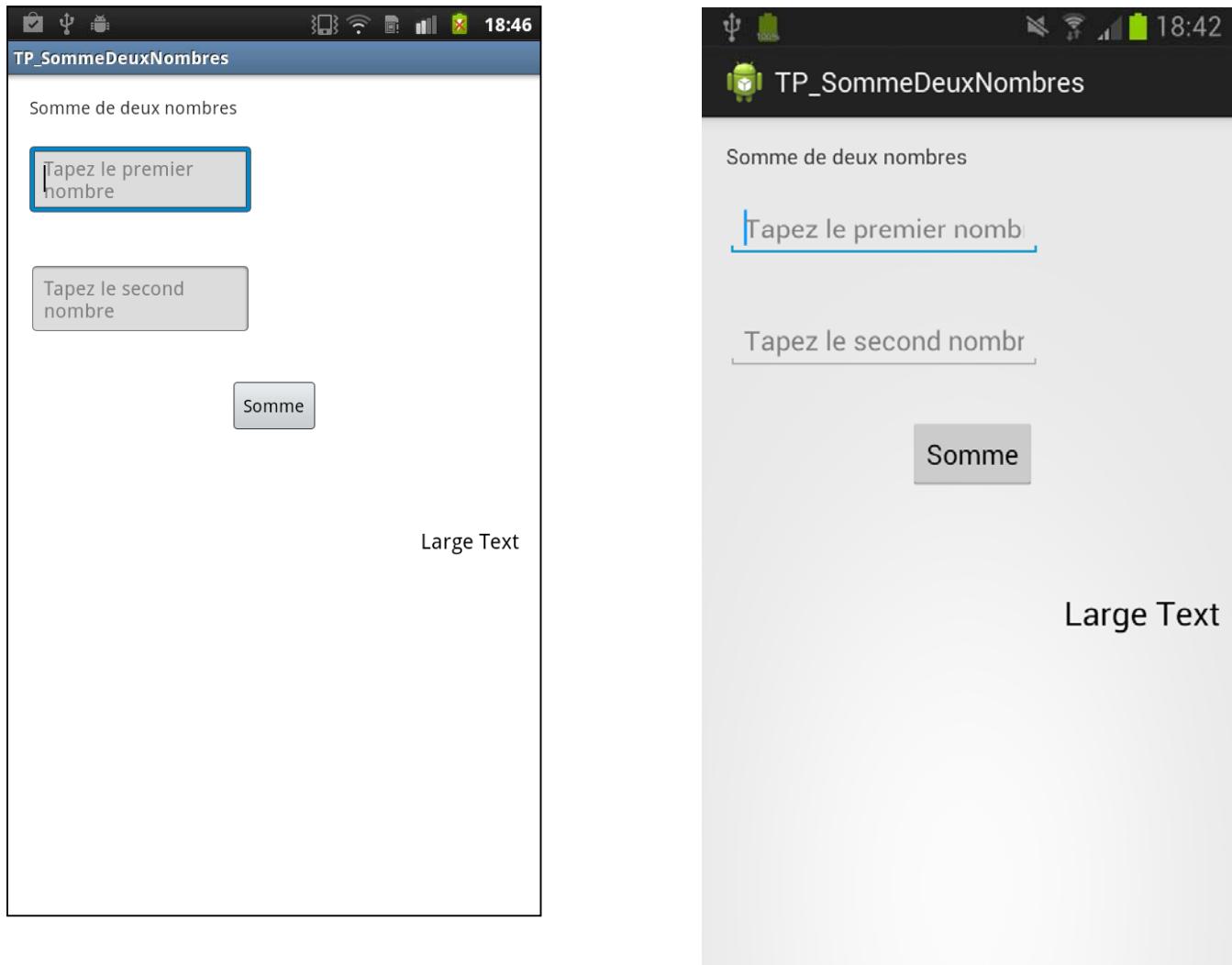
- Cf. projet TP_Widgets_de_base
 - Cf. les clics sur ces widgets

Les fichiers sources sont accessibles à
https://gitlab.com/m2eservices/TP_Widgets_de_base.git



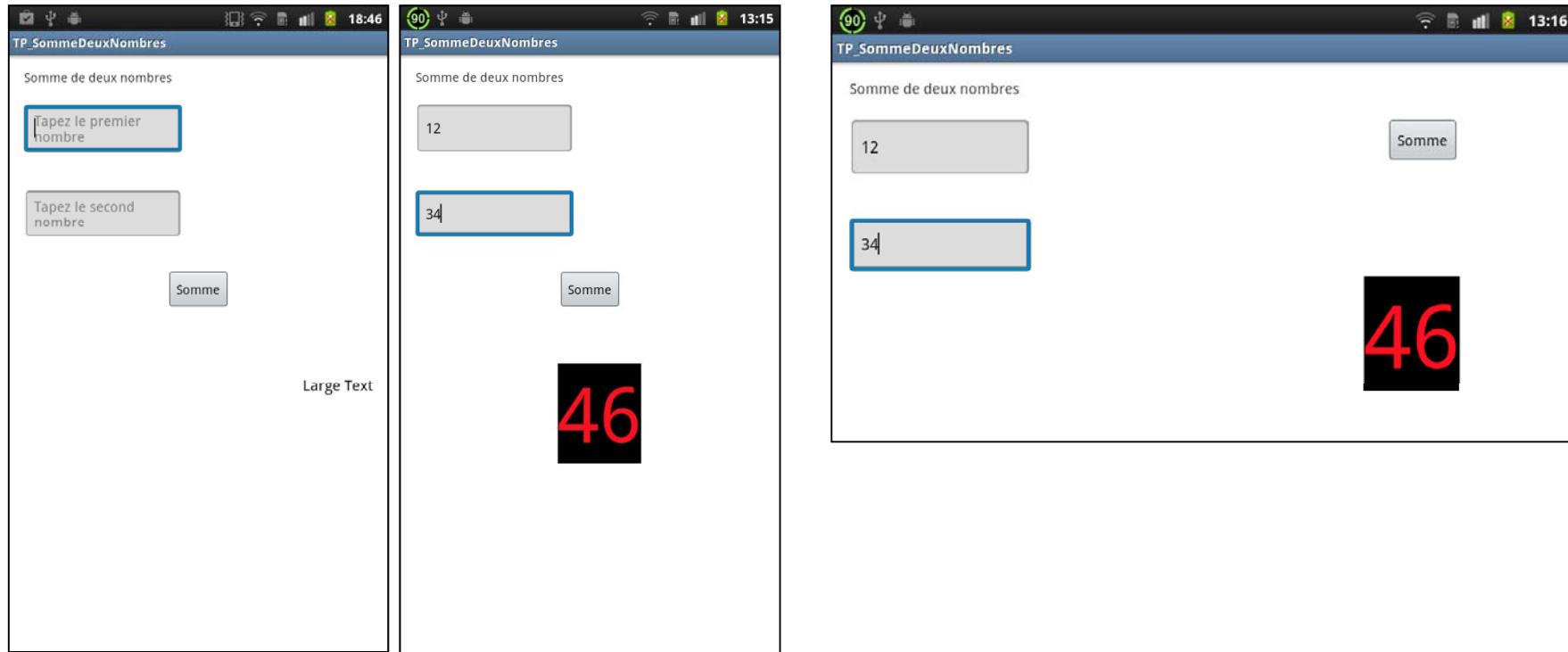


TP : somme de deux nombres





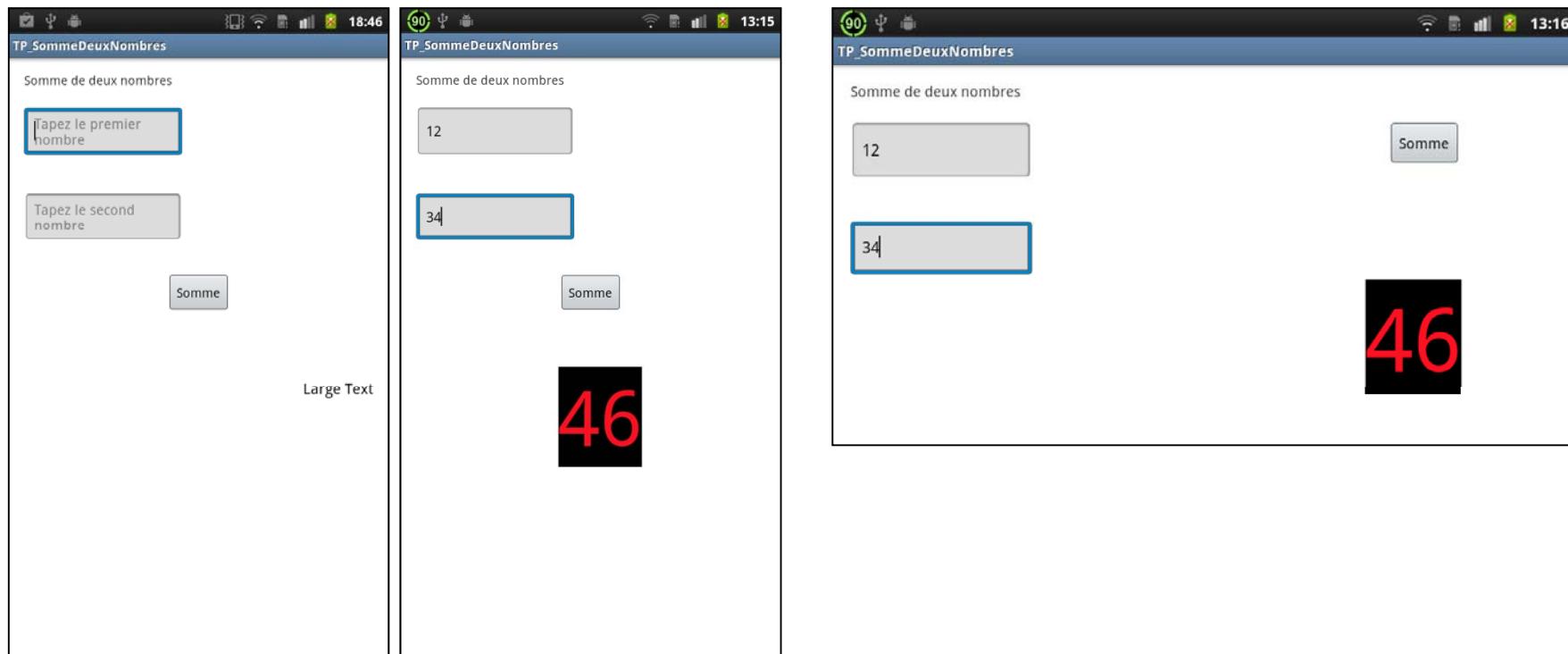
TP : somme de deux nombres



Puis utilisez les layouts
portrait/paysage, les strings, les
couleurs, ~~les styles~~ (1^{er} TP à venir)...



TP : somme de deux nombres



https://gitlab.com/m2eservices/TP_Somme_de_deux_nombres.git



Styles (layout)

Cf. 1^{er} TP ☺



dimens.xml

< /

Cf. 1^{er} TP ☺



styles.xml

<!

Cf. 1^{er} TP ☺

ge</item



Compatibilité !

- <http://developer.android.com/training/material/compatibility.html>
- Define Alternative Styles
 - You can configure your app to use the material theme on devices that support it and revert to an older theme on devices running earlier versions of Android:
 - Define a theme that inherits from an older theme (like Holo) in res/values/styles.xml.
 - Define a theme with the same name that inherits from the material theme in res/values-v21/styles.xml.
 - Set this theme as your app's theme in the manifest file.
 - **Note:** If your app uses the material theme but does not provide an alternative theme in this manner, your app will not run on versions of Android earlier than 5.0.
- Provide Alternative Layouts
 - If the layouts that you design according to the material design guidelines do not use any of the new XML attributes introduced in Android 5.0 (API level 21), they will work on previous versions of Android. Otherwise, you can provide alternative layouts. You can also provide alternative layouts to customize how your app looks on earlier versions of Android.
 - Create your layout files for Android 5.0 (API level 21) inside res/layout-v21/ and your alternative layout files for earlier versions of Android inside res/layout/. For example, res/layout/my_activity.xml is an alternative layout for res/layout-v21/my_activity.xml.
 - To avoid duplication of code, define your styles inside res/values/, modify the styles in res/values-v21/ for the new APIs, and use style inheritance, defining base styles in res/values/ and inheriting from those in res/values-v21/.
- Use the Support Library
 - The v7 Support Libraries r21 and above includes the following material design features:
 - Material design styles for some system widgets when you apply one of the Theme.AppCompat themes.
 - Color palette theme attributes in the Theme.AppCompat themes.
 - The RecyclerView widget to display data collections.
 - The CardView widget to create cards.
 - The Palette class to extract prominent colors from images.