Développement mobile : Android

Lesson 1

Carl Esswein

Dernière mise à jour : V1.7 - février 2024.

Table des matières

Lesson 1

Lancé en 2008, Android est actuellement l'OS pour smartphones le plus répandu au monde, et de très loin (Android équipait déjà près de 85% des smartphones vendus en 2016 et reste entre 83% et 85% depuis). Son caractère gratuit et open-source, ainsi que l'utilisation du langage Java – quoique de moins en moins – sont des atouts considérables ayant probablement beaucoup fait pour son succès.

Objectifs

L'objectif de cette première séance est de vous présenter **l'environnement** de développement Android ainsi que le **développement** d'une première application et son **déploiement**. Vous y découvrirez également la stucture d'une application android et des notions importantes telles qu'**activité** ou **ressources**.

1 Mise en place de l'environnement

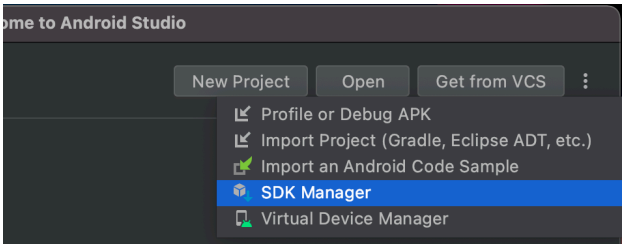
Cette première partie décrit les différentes étapes de la mise en place d'un environnement de développement Android.
Dans la machine mise à votre disposition pour les séances de cours-TP, les différents éléments nécessaires sont déjà installés ; si vous prévoyez de l'utiliser, vous pouvez passer à la section 2.
Voici tout de même, à toutes fins utiles, les étapes à suivre pour une installation de l'environnement, sur vos machines personnelles par exemple.

1. Installez **Android Studio**
   - Téléchargeable sur http://developer.android.com/studio
   - Instructions en cas de besoin : https://developer.android.com/studio/install
   - Inclus : l'installation du **SDK Android**.
   - Android Studio est l'IDE « officiel » Android depuis fin 2015.
   - L'utilisation de cet IDE n'est pas obligatoire mais nous allons l'utiliser pour toute la suite de ce cours.
   - NB : Android utilise le langage java et Android studio embarque un JDK.

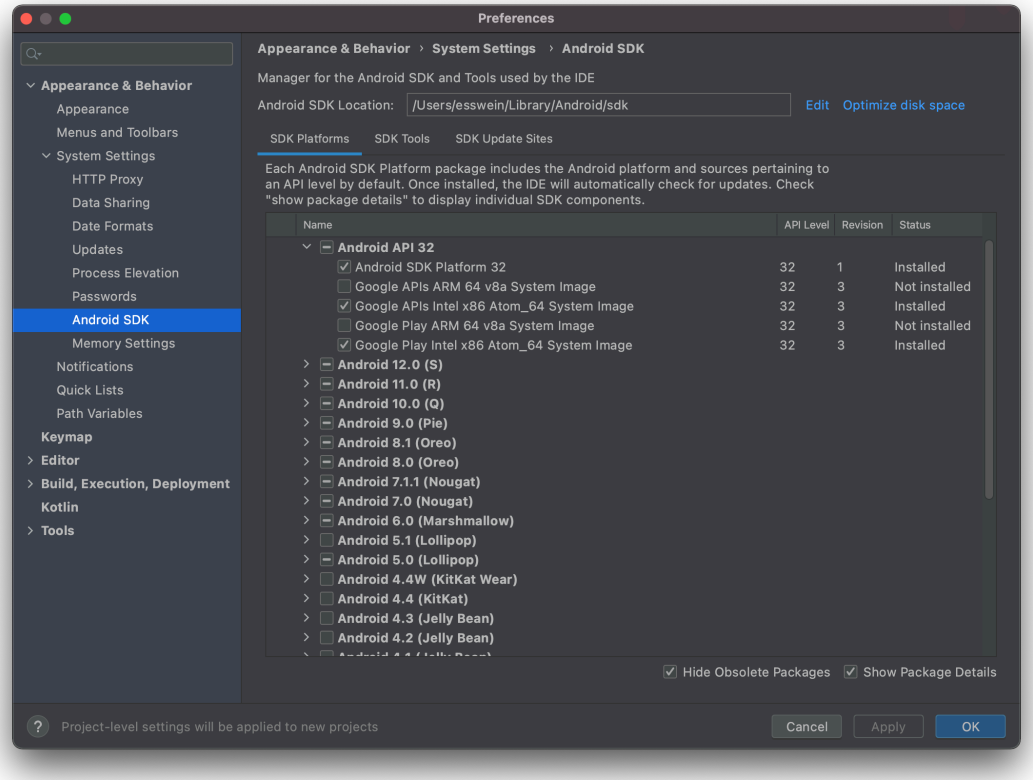
2. Lancez Android Studio une première fois.
   - **Lors de son premier lancement**, celui-ci met à jour le SDK Android ce qui peut prendre un peu de temps.

3. Mettez à jour les **outils** et la **plateforme** via le *SDK Manager*.
   - Le SDK Manager est un des outils livrés avec le SDK Android qui vous permet de gérer les différents éléments installés constituant votre environnement de développement Android, IDE exclu. Nous en reparlerons.

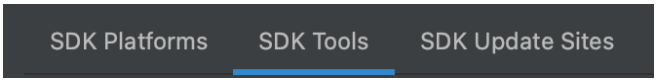
   1. Lancez Android Studio puis ouvrez le menu déroulant des outils (en haut à droite ) et lancez le *SDK Manager*  :



Le SDK Manager se lance.
Cochez « Show Package Details » en bas à droite.



2. Les 3 onglets « SDK Platforms », « SDK Tools » et « SDK Update Sites » vous permettent de mettre à jour votre SDK Android et les outils périphériques.



3. Sélectionnez les élements à installer. Je vous suggère de sélectionner au mimimum le SDK de l'API la plus récente (Android SDK Platform XX) ainsi qu'une ou deux images système pour l'émulateur.
    - *(Voir screenshot ci-dessus avec l'exemple de l'API 32)*

   Dans SDK Tools, sélectionnez les versions les plus récentes de :
    - Android SDK Build-Tools
    - Android Emulator (optionnel mais c'est mieux avec)
    - Android SDK Platform-Tools
    - Android SDK Tools

   Cliquez sur Ok pour valider les téléchargements et les installations. Cette étape peut être longue.

D'autres mises à jour peuvent être requises ultérieurement. Le SDK Manager restera accessible.
https://developer.android.com/studio/intro/update#sdk-manager vous fournit plus de détails.

Maintenant que vous disposez d'un environnement prêt à fonctionner, nous allons voir comment construire une première application simple.

2 Prise en main : une première application

2.1 Création

Ouvrez Android Studio et lancez le wizard de création d'un nouveau projet.

La première étape de création du projet consiste à sélectionner les catégories matérielles cibles et de créer automatiquement la trame de votre première activité : restez sur l'onglet *Phone and Tablet* et sélectionnez *Empty Views Activity*.


La notion d'***activité*** est centrale puisque les activités constituent un des 4 types de composants des applications android (nous y reviendrons). Les activités d'une application en constituent en quelque sorte la partie visible. Pour commencer, disons qu'une activité c'est un écran de l'IHM. Nous ajusterons cette définition par la suite.

Cliquez sur ***Next***.

L'étape suivante vous propose de configurer votre projet : choisissez un nom d'application, par exemple « My First App », changez si besoin son package racine ainsi que son emplacement (alternativement notez où Android Studio place votre projet par défaut sur votre système de fichiers...).

Sélectionnez ensuite un **langage** de programmation : Java.

NB : Depuis Android 9, Android support à la fois le langage Java et le langage **Kotlin**. Nous ferons ce cours avec Java mais vous pouvez tester Kotlin à votre guise.

Enfin, sélectionnez « API 26 : Android 8.0 (Oreo) » pour *minimum SDK*.

Plus vous sélectionnez une version ancienne plus vous élargissez la compatibilité de votre application avec des versions antérieures, et donc plus vous augmentez le nombre d'appareils qui seront en mesure d'installer votre application. Le revers de la médaille c'est que cela réduit d'autant l'usage des fonctionnalités propres aux version plus récentes.

Cliquez sur ***Finish*** pour qu'Android Studio crée votre projet.


*Soyez **très** patients au premier lancement d'Android Studio, il y a généralement une indexation chronophage en cours (voir barre d'état) et/ou "loading" affiché dans l'arborescence à gauche.*

Votre première application est maintenant créée. Voyons comment l'exécuter.

2.2 Exécution

Vous avez deux alternatives pour l'exécution de votre application à des fins de tests : via l'émulateur fourni avec les outils du SDK Android, ou sur un périphérique Android réel connecté à votre machine.

2.2.1 Avec l'émulateur

L'émulateur Android vous permet d'exécuter vos applications sur virtuellement n'importe quel périphérique Android. Pour cela, le **Device Manager**  (*aka.* AVD Manager) offre la possibilité de créer et exécuter différents périphériques virtuels appelés AVD (Android Virtual Device). Les configurations hard et soft peuvent être paramétrées.

Lancez le Device Manager (barre d'outils ou menu Tools > Device Manager).

Vous avez la possibilité de créer l'AVD de votre choix (onglet "Virtual" puis "Create Device"). Ne soyez pas trop gourmands si votre machine n'a pas les reins solides... Utilisez de préférence l'image que vous avez installée précédemment avec le SDK Manager (*Cf.* étape installation décrite plus tôt) ou téléchargez une image suggérée récente (Android 14 par exemple).

Une fois créé, **lancez l'AVD** via le bouton Start dans le Device Manager.

Lancez maintenant votre application android en sélectionnant votre AVD en tant que cible (liste déroulante à côté du bouton "Run").

Votre application est maintenant installée sur votre AVD et vous pouvez utiliser l'émulateur – presque – comme un matériel réel : utiliser votre application, naviguer dans les menus/services de votre appareil, recevoir des SMS ou passer des appels…

2.2.2 Avec un matériel réel

Voyons maintenant comment exécuter votre application sur un matériel réel (Device Manager > onglet "Physical").

En premier lieu il faut **activer le débugging USB** sur le matériel. L'option se trouve dans *Paramètres* > *Options de développement* (ou dans *Paramètres* > *Applications* > *Développement* pour des versions d'android 3.2 ou plus anciennes).

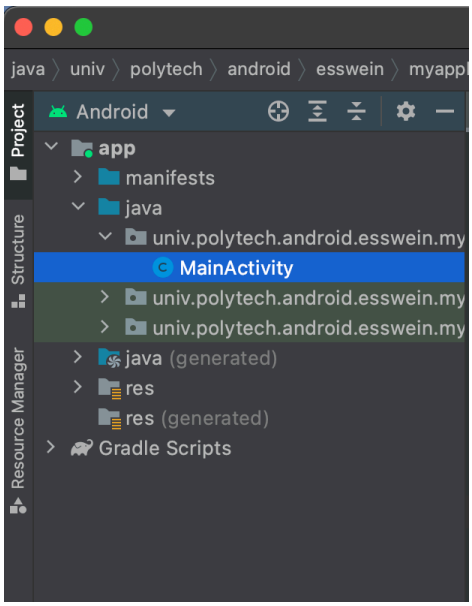
Pour Android 4.2 ou plus récent, le menu *Options de développement* est caché par défaut. Pour le faire apparaître aller dans *Paramètres* > *A propos de l'appareil* et taper 7 fois sur le numéro de build. (Non ce n'est pas une blague). Le menu *Options de développement* est maintenant accessible.

Vous pouvez également appairer le Device Manager avec votre matériel **via le Wifi**. Tous les détails sont ici : https://developer.android.com/studio/run/device#wireless

Lancez maintenant votre application en sélectionnant votre matériel en tant que cible (« *Select Deployment Target* »).

3 Structure et contenu

Voyons maintenant comment est structurée cette première application.



Ouvrez le menu *Project* sur la gauche, et si besoin sélectionnez *Android* dans la vue (liste déroulante en haut du menu).

Gradle Scripts contient les scripts et autres paramètres de Gradle qui est l'outil de build d'Android Studio.

App contient tous les éléments propres à votre projet :

- les fichiers manifest, et notamment *AndroidManifest.xml* qui décrit votre application et ses composants,
- les sources java,
- les ressources (dossier **res**)

Les dossiers marqués "(generated)", comme leurs noms l'indiquent, correspondent à des dossiers contenant des éléments générés en interne par Android. Vous n'avez a priori pas besoin d'en connaître les détails pour l'instant.

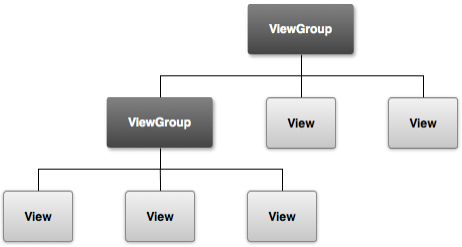
3.1 Ressources

Q1. Naviguez dans cette structure et modifiez votre application pour qu'elle affiche « Hey Polytech ! » à la place de « Hello World! ».

Q2. Quels types de ressources avez-vous pu identifier ? Lister les différentes catégories et leur utilisation.

Nous allons maintenant changer l'IHM en créant un champ de saisie et un bouton de validation, et voir comment interagir entre les layout XML et le code java.

3.2 IHM




Les composants de vos IHM sont des (classes filles de) `View`[1]. Les layout (`RelativeLayout` ou `LinearLayout` par exemple) sont des `ViewGroup` : ce sont des `View` particulières qui peuvent contenir d'autres `View`. Vous allez donc construire vos IHM comme des hiérarchies de `View` et `ViewGroup`, *i.e.* comme des hiérarchies de layouts et de composants.

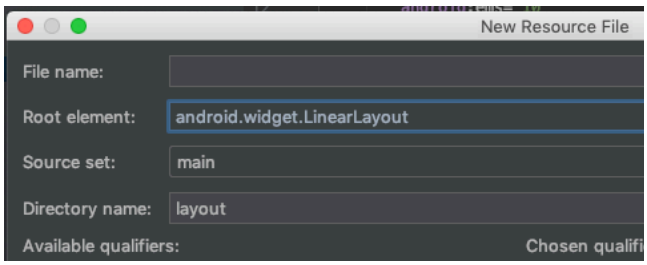
Q3. Créer un fichier de layout *activity_main_v2.xml*,


Vous disposez de trois modes d'édition des fichiers de layout XML : le mode "Code" en pur XML, le mode "Design" pour utiliser l'outil de design drag-n-drop. Il y a également une version mixte qui affiche les deux ("Split"). Le choix se fait en haut à droite de la fenêtre d'édition.

3.1. Ajoutez-y un élément `EditText` puis un élément `Button` en utilisant `wrap_content` pour les hauteurs et les largeurs.

3.2. Ajoutez l'attribut `android:hint="Saisir un message"` au champ texte (supprimez l'attribut `android:text` si besoin) et changez le libellé du bouton pour qu'il affiche "Envoyer".

3.3. **Testez** ce layout en changeant le paramètre de `setContentView()` dans votre code .java.



3.4. Faites maintenant la même chose (Q3.1 à 3.3) dans un nouveau layout – disons *activity_main_v3linear.xml* – en précisant "LinearLayout" comme élément racine (*cf.* ci-contre).
Testez successivement `horizontal` et `vertical` comme orientation du `LinearLayout`.

Même si c'est moins fréquent, il est également possible de construire/modifier vos IHM directement dans le code java, comme vous le feriez pour une application java avec Swing. Une fois l'arborescence de composants construite, on appelle `setContentView()` en lui passant le `View` racine.

3.5. Refaites l'équivalent de la question précédente à partir du code java, sans utiliser le layout XML (Utilisez `this` comme référence au *contexte* pour les constructions de vos composants, *Cf.* TIP ci-dessous).


Le `Context`[2] est l'interface d'accès à des informations & services globaux liés à votre application. C'est par exemple le moyen de lancer de nouvelles activités ou d'accéder à des constantes niveau application.

[1] du package `android.view`
[2] du package `android.context`


La solution précédente 100% java n'est pas préconisée, on préfèrera la version XML à chaque fois qu'on aura le choix. Il existe cependant des circonstances qui pourront la rendre nécessaire. Vous mettrez généralement en oeuvre des solutions mixtes code et XML.

Avec Android, les composants graphiques (`View`) peuvent optionnellement se voir attribuer un identifiant leur permettant d'être référencés depuis le code java. Pour ce faire il suffit d'ajouter au composant l'attribut `android:id` et de lui donner une valeur. Si un composant n'a pas vocation à être utilisé par le code – *e.g.* un `ViewGroup` simple – il n'est pas nécessaire de lui affecter un identifiant.

Exemple :

```
android:id="@+id/monBouton"
```

La définition ci-dessus indique que le composant concerné aura pour identifiant la valeur de la variable `monBouton`. Plus précisément, c'est **`R.id.monBouton`** qui aura pour valeur l'identifiant du composant, **`R`** étant une classe statique générée automatiquement par le compilateur et **`id`** étant une classe interne statique de `R` contenant des identifiants.

A partir du code java, l'accès à un composant de l'IHM se fait via la méthode `findViewById()` qui prend en paramètre l'identifiant du composant, `R.id.monBouton` dans notre cas.

Q4. Si ce n'est pas déjà fait, affectez des identifiants à tous vos composants dans *activity_main_v2*, layouts inclus. Compilez puis, sans toucher aux layouts XML, complétez la méthode `onCreate()` de votre activité pour modifier le texte du bouton et afficher « Go ».

Les autres catégories de ressources de */res/values/* sont également accessibles via *R.java*. En particulier on accèdera aux ressources de type `String` par :

`getString(R.string.<nomRessource>)` ou `@string/nomRessource`

...selon qu'on est dans un fichier java ou dans un layout XML.

Q5. Modifier le nom de votre application dans le fichier .XML adéquat.

Q6. Externalisez vers *strings.xml* toutes les chaines de caractère de votre programme (*i.e.* depuis le code java et les layouts). Cela peut sembler un détail mais **c'est essentiel pour la maintenabilité de vos codes**.

4 Exercices

4.1 App "plus ou moins"

Q7. Construisez un nouveau projet à partir d'une *Empty Views Activity*, et créez une application pour jouer au « plus ou moins » : au lancement, l'application génère aléaroirement un nombre entier entre 1 et 100 et l'utilisateur doit déterminer ce nombre en faisant des propositions. A chaque proposition, l'appli répond « plus grand » ou « moins grand » selon la valeur soumise par l'utilisateur.

7.1. Créer l'IHM dans le fichier de layout et mettez tout de suite toutes vos `String` dans *string.xml*.

7.2. Modifiez votre activité pour lui ajouter les attributs nécessaires et initialiser le nombre aléatoire.

7.3. Ajoutez à votre activité une méthode `check()` qui sera appelée quand le bouton de proposition d'une valeur sera cliqué et faites les actions nécessaires sur vos composants : tester la valeur soumise (& gérer les exceptions…), mettre à jour l'affichage, …


Nous verrons la prochaine fois comment faire appeler cette fonction lors de l'appui sur le bouton.

4.2 Bonus

Les questions/exercices ci-dessous sont optionnels en séance. Ils sont à faire si vous avez pris de l'avance, ou chez vous pour réviser/aller plus loin.

Q8. Dans l'app "Plus ou Moins" :

8.1. Intégrer un compteur de coup et félicitez l'utilisateur s'il va vite (relativement à `log10(n)`).

8.2. Rajouter dans l'IHM deux champs pour que l'utilisateur saisisse le min et le max avant de commencer une partie.

8.3. Faire plusieurs versions de l'IHM (i.e. plusieurs layout...) pour des config différentes, et tester via l'émulateur. Faites a minima (1) une version standard – déjà faite – (2) une version en mode paysage (3) une version pour grand écran (format tablette).

Q9. Dans l'app "Hello World" :

9.1. Changez la font du message pour qu'elle soit en gras, italique, rose, taille 30.

9.2. Centrez verticalement le message.

Conclusion

A l'issue de cette première séance, vous êtes capables de :

- Installer et/ou mettre à jour un environnement de développement android,
- Construire un projet android et l'exécuter sur cible virtuelle ou réelle,
- Construire ou modifier des fichiers de layout simples pour créer des IHM,
- Faire, dans les deux sens, le lien code<->layout

et vous savez :

- Comment est structurée une application,
- Ce que sont, au sens android, les ressources et les identifiants.

5 Annexes

5.1 Références

- La plupart des images de ce document proviennent de http://developer.android.com/
- **Référence de l'API standard** : http://developer.android.com/reference/packages
- Tous les guides de la documentation officielle, par thématique : http://developer.android.com/guide

Et aussi, éventuellement :

- http://developer.android.com/courses