# TP 1-2 de programmation Android

**Résumé**

L'objectif de ce premier TP est de ce familiariser avec l'environnement de développement Android, construire une première application simple avec différents écrans (Activity, View), comprendre les intentions (Intent) et les permissions.

Une adresse utile : `http://developer.android.com/guide/components/index.html`

En local : `/mnt/n7fs/ens/tp_cregut/android-sdk-linux_86/docs/offline.html`

## 1 Préparation d'Éclipse

Utiliser la version Éclipse qui est installée dans :

`/mnt/n7fs/ens/tp_android/eclipse/eclipse`

et créez-vous un nouveau Worspace.

Il faut commencer par indiquer à Éclipse (au plugin ADT en fait) où se trouve le SDK d'Android. Faire Windows > Preferences, puis sélectionner Android et renseigner SDK Location avec :

`/mnt/n7fs/ens/tp_android/android-sdks`

Si la perspective proposée par défaut n'est pas la perspective Java, ouvrez perspective Java (Window > open perspective > other > Java).

## 2 Un peu de documentation

Lire rapidement l'introduction sur les les applications Android en prêtant une attention particulièrement au cycle de vie d'un composant. Voir : `http://developer.android.com/guide/topics/fundamentals.html`

## 3 Première application

### 3.1 Créer et lancer un AVD

Un AVD (Android Virutual Device) est un émulateur de dispositif Android.

Lancer le gestionnaire d'AVD depuis Eclipse en faisant : Windows > Android Virtual Device Manager. Créer un AVD en faisant New, puis en donnant un nom (avd42), un type de smartphone ou tablette (Nexus One par exemple), une cible (Android 4.2, API Level 18), une capacité RAM (20Mo) et une capacité pour la carte SD (20 Mo). Lancer en le sélectionnant et en faisant Start.

### 3.2 Créer un projet Android

Créer un "Android Project" (File > New > Project > Android Application Project) . Renseigner les informations demandées en particulier le nom du projet (Browser), le nom de l'application (Web Browser), le paquetage (fr.enseeiht.LOGIN.navigateur), le nom de l'activité principale (BrowserActivity). Attention, LOGIN est à remplacer par votre login. Regarder les différents éléments du projet créé, en particulier les ressources (res), les sources (src), la classe R.java engendrée automatiquement dans le répertoire (gen), le fichier de manifeste (AndroidManifest.xml). Le projet ne devrait pas contenir d'erreur. On peut le lancer sur l'émulateur en faisant Run As > Android Application. L'application devrait se lancer sur l'émulateur.

Remarque : On pourra consulter le tutorial Hello World : `http://developer.android.com/resources/tutorials/hello-world.html`

Les fichiers les plus importants sont :

- SRC > BrowserActivity.java : notre code source
- Res > Layout > NomDeL'Appli.xml ou main.xml suivant la version d'Eclipse : le fichier xml qui gère les différents composantes de la fenêtre.
- Res > Values : string.xml : les valeurs des chaines de caractère.

### 3.3 La classe Log et la vue LogCat

L'application s'exécutant sur un appareil sans console, il n'est pas pratique de faire des traces avec un système tel que System.out.println. Aussi, il faut (et il est de toute façon préférable) d'utiliser un outil de journalisation (Logger). Android fournit la classe Log. Pour visualiser les traces correspondantes, on utilisera la vue LogCat sous Eclipse.

Sous Eclipse, ajouter la vue LogCat (Windows > Show View > other > LogCat), et faire de même pour ajouter la vue "File Explorer".

Dans la classe BrowserActivity, ajouter l'instruction suivante :

**Log.d(TAG, "onCreate") ;**

au début de la méthode onCreate. On définira TAG dans la classe de la manière suivante (attribut constant de la classe) :

**private static final String TAG = "BrowserActivity" ;**

Eclipse vous permet d'ajouter les import qui manquent. Il suffit par exemple de faire (Ctrl-Shift-O).

Exécuter de nouveau l'application et constater que le message est affiché dans la vue LogCat. Ajouter la permission (uses permission) correspondant à INTERNET. On pourra utiliser l'onglet permission de l'assistant permettant d'éditer le manifest (AndroidManifest.xml).

Déployer et exécuter l'application. Tout devrait maintenant fonctionner ! Mettre quelques Log permettra de suivre l'exécution de votre application.

### 3.4 Modifier l'écran d'accueil de l'application

Modifier l'écran d'accueil de l'application pour qu'il fasse apparaître sur la même ligne : une zone de saisie (EditText ou Plain Text), un bouton (Go) et un bouton ( ?). La zone de saisie sera initialisée avec l'URL "`http://www-tr.enseeiht.fr`".

Modifier le layout (main.xml ou NomDeL'Appli.xml). On pourra utiliser soit le mode

Toutes les chaînes de caractères devront être définies dans le fichier strings.xml dans les ressources "values".

Exécuter l'application sur l'AVD pour vérifier.

On pourra consulter le tutorial Hello Views : `http://developer.android.com/resources/tutorials/views/index.html`.

### 3.5 Définir le comportement du bouton Go

Lorsque l'utilisateur clique sur le bouton Go, l'URL saisie dans la zone de texte doit être visualisée. L'objectif est d'utiliser le navigateur Web d'Android. La communication entre composants Android se fait par envoi de messages appelés intention (Intent).

Il faut donc :

- Retrouver le composant EditText (en utilisant findViewById)
- Lui associer un listener
- Le listener doit créer l'intention et lancer l'activité.

Aussi, il suffit de :

- créer une intention implicite correspondant à l'action souhaitée (ici Intent.ACTION_VIEW). Ici, il faudra aussi fournir l'URI obtenue grâce à la méthode Uri.parse
- démarrer l'activité correspondant à cette intention (startActivity).

Exécuter l'application !

## 4 Lecteur video

Nous allons maintenant transformer l'application pour qu'elle puisse lire des vidéos.

### 4.1 Transfert de video sur la carte SD

Il faut commencer par transférer une vidéo sur la carte SD de l'émulateur. Ouvrez la perspective DDMS pour accéder au file manager (ou ouvrez la vue si vous l'avez ajoutée), et faites "push file on device" (petit bouton en haut à droite). Choisissez la vidéo :

`/mnt/n7fs/ens/tp_android/TP_01/documentariesandyou.mp4`

### 4.2 Créer une nouvelle intention

Modifier l'application précédente pour qu'elle ouvre le fichier sur la carte SD. On utilisera pour récupérer le chemin d'accès à la carte SD : **Environment.getExternalStorageDirectory().getPath()**

Du point de vue de l'intention, il faut maintenant définir à la fois la donnée et le type de cette donnée, avant de démarrer l'activité. Voir la méthode **setDataAndType** de la classe intention.

## 4.3 Intégration de la vidéo

Dans la version précédente, une nouvelle activité est lancée pour afficher la vidéo. Il serait intéressant que la vidéo soit lancée directement dans notre application. Plusieurs classes peuvent être utilisées, notamment VideoView (simple à mettre en place, mais limité) ou MediaPlayer (plus de fonctionnalités).

### 4.3.1 Composant VideoView

Dans le layout, créer un cadre VideoView dans lequel nous allons afficher la vidéo.

Transformer l'application afin qu'elle visualise la vidéo dans le cadre prévu à cet effet au lieu de lancer une nouvelle activité pour le faire.

Pour cela :

- Créer un nouveau composant VideoView pour que l'application visualise directement la vidéo.
- Ajouter un bouton de lecture pour démarrer la vidéo, qui sera affichée dans le composant VideoView.
- Ajouter un bouton pause.

### 4.3.2 Media Player

La classe VideoView ne propose que des activités limitées sur la vidéo. La classe MediaPlayer est plus étendue. Transformer l'application pour que la vidéo soit gérée par cette classe. Pour cela ajoutez une vue **surfaceView** dans le Layout > main.xml, et activez la vue en faisant :

**getWindow().setFormat(PixelFormat.UNKNOWN) ;**
**surfaceView = (SurfaceView)findViewById(R.id.surfaceview) ;**
**surfaceHolder = surfaceView.getHolder() ;**
**surfaceHolder.addCallback(this) ;**
**surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS) ;**

A l'aide de mediaplayer, redéfinissez le comportement des boutons pour lire ou mettre en pause la vidéo, qui sera affichée dans la vue précédente en utilisant la fonction **public void surfaceCreated(SurfaceHolder holder) { mediaPlayer.setDisplay(surfaceHolder) ; }** .

Les fonctions **mediaPlayer.***** permettront de contrôler la vidéo, ainsi que d'obtenir de nombreuses informations sur la vidéo (taille, durée...) que vous afficherez sous la vidéo. En utilisans les différents Listener de la classe et le LogCat, vous étudierez les différentes étapes du chargement de la vidéo.

En utilisant SeekBar, vous afficherez enfin une barre de défilement dans la vidéo.

## 4.4 Permissions

Ajouter la permission (use permission) correspondant à la lecture et l'écriture sur la carte SD. On pourra utiliser l'onglet permission de l'assistant permettant d'éditer le manifest (AndroidManifest.xml).

Déployer et exécuter l'application. Tout devrait maintenant fonctionner !

Votre application devrait maintenant ressembler à ca :