



Expert  
EXPO

Développez une application

# Android

Programmation en Java  
sous **Android Studio**

Téléchargement  
[www.editions-eni.fr](http://www.editions-eni.fr)



Sylvain HÉBUTERNE

Les éléments à télécharger sont disponibles à l'adresse suivante :  
**<http://www.editions-eni.fr>**  
Saisissez la référence de l'ouvrage **EIASAND** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

## Avant-propos

### Chapitre 1 Environnement de développement

- 1. Architecture d'Android ..... 9
  - 1.1 Présentation d'Android ..... 9
  - 1.2 Architecture ..... 12
  - 1.3 Play Store ..... 13
- 2. Android Studio ..... 14
  - 2.1 Installation sous Windows ..... 14
  - 2.2 Installation sous Max OS X ..... 18
  - 2.3 Installation sous Linux ..... 18
- 3. Compléments ..... 18

### Chapitre 2 Principes de base d'Android

- 1. Concepts de base d'une application Android ..... 25
  - 1.1 Contexte d'exécution/Sécurité ..... 25
  - 1.2 Package ..... 26
  - 1.3 Fichier AndroidManifest ..... 26
  - 1.4 Les activités ..... 27
- 2. Un premier projet : Hello World ..... 27

# 2 \_\_\_\_\_ Développez une application Android

Programmation en Java sous Android Studio

3. Préparer un terminal virtuel . . . . .	37
3.1 Utiliser le terminal défini par défaut . . . . .	38
3.2 Créer un terminal virtuel . . . . .	40
4. Configurer un terminal physique . . . . .	43
5. Débogage, trace. . . . .	44
5.1 Messages Toast . . . . .	47
5.2 Logcat . . . . .	48
5.3 Autres outils . . . . .	49

## Chapitre 3

### Préparation du projet LocDVD

1. Gérer la fragmentation . . . . .	51
1.1 Densité d'écran . . . . .	51
1.2 Les ressources . . . . .	52
1.3 Density-independent pixel. . . . .	54
2. Jongler avec plusieurs versions d'API . . . . .	56
3. Internationalisation . . . . .	57
4. Généralisation. . . . .	58
5. Préparation du projet LocDVD . . . . .	60

## Chapitre 4

### Consultation et saisie de données

1. Introduction . . . . .	63
2. Les activités – cycle de vie d'un écran . . . . .	63
2.1 Création d'une nouvelle activité . . . . .	64
2.2 Cycle de vie d'une activité . . . . .	65
2.3 Initialisation de l'activité . . . . .	66

- 3. Construction d'un écran de consultation des données ..... 67
  - 3.1 Création de l'interface ..... 68
    - 3.1.1 Code source ..... 70
    - 3.1.2 Les propriétés ..... 71
    - 3.1.3 Propriétés spécifiques pour les composants  
 LinearLayout et TextView ..... 73
    - 3.1.4 Définir un identifiant ..... 74
    - 3.1.5 Construction de l'interface ..... 75
  - 3.2 Liaison du fichier de layout avec le code de l'activité ..... 77
  - 3.3 Inscription dans le Manifest ..... 82
- 4. Saisie des données, contrôles principaux ..... 84
  - 4.1 Création du squelette de l'écran ..... 85
  - 4.2 Gestion des boutons ..... 90
    - 4.2.1 Réagir au clic ..... 91
    - 4.2.2 Ajouter un composant ..... 92
  - 4.3 Test de l'écran de saisie ..... 95
- 5. Mettre en forme un écran, gérer l'affichage adaptatif ..... 97
  - 5.1 Faire défiler l'écran ..... 97
  - 5.2 Contrôler la saisie ..... 100
  - 5.3 Gérer la rotation de l'écran ..... 103
    - 5.3.1 Sauvegarder les données ..... 103
    - 5.3.2 Restaurer les données ..... 104

**Chapitre 5**

**Persistance des données**

- 1. Introduction ..... 109
- 2. Création et modification d'une base de données ..... 109
  - 2.1 Création de la base de données ..... 110
  - 2.2 Modification de la base de données ..... 114

# 4 \_\_\_\_\_ Développez une application Android

Programmation en Java sous Android Studio

3. Manipulation des données . . . . .	115
3.1 Création de la classe DVD . . . . .	115
3.2 Requêtes de sélection . . . . .	118
3.3 Requête d'enregistrement . . . . .	121
3.4 Requête de suppression . . . . .	124
3.5 Transaction . . . . .	124
4. Sauvegarde des préférences utilisateurs . . . . .	125
5. Lire et écrire dans un fichier . . . . .	127
5.1 Intégrer un fichier de données . . . . .	127
5.2 Lire les données et les enregistrer . . . . .	128
5.3 Mémoriser la lecture du fichier . . . . .	131

## Chapitre 6 Contrôles avancés

1. Les listes . . . . .	133
1.1 Intégrer une liste . . . . .	133
1.1.1 Intégrer un composant ListView . . . . .	134
1.1.2 Déclaration d'un layout pour les éléments de la liste . .	136
1.1.3 Implémenter un adaptateur . . . . .	137
1.2 Relier le composant ListView à l'adaptateur . . . . .	141
1.3 Gérer le clic sur un élément . . . . .	146
1.4 Afficher le DVD sélectionné . . . . .	153
2. GridView, liste déroulante . . . . .	157
2.1 Composant GridView . . . . .	157
2.2 Liste déroulante . . . . .	157
3. TimePicker/DatePicker . . . . .	158
3.1 Ajout d'un champ date de visionnage . . . . .	158
3.2 Saisie de la date de visionnage . . . . .	160

- 4. Créer son propre composant réutilisable ..... 165
  - 4.1 Surcharger un composant de la plateforme ..... 165
  - 4.2 Définir des attributs personnalisés ..... 166
  - 4.3 Intégrer le composant dans un layout..... 168

**Chapitre 7**  
**Les fragments**

- 1. Présentation ..... 171
  - 1.1 Fragment et activité ..... 171
  - 1.2 Cycle de vie ..... 172
  - 1.3 Compatibilité ..... 173
- 2. Travailler avec les fragments ..... 174
  - 2.1 Création du fragment..... 174
  - 2.2 Modification de l'activité hôte ..... 179
- 3. Mise en œuvre du modèle Master/Detail ..... 183
  - 3.1 Mise en place du layout ..... 185
  - 3.2 Modification de la vue détaillée..... 189
  - 3.3 Gestion des fragments ..... 195

**Chapitre 8**  
**Navigation et boîtes de dialogue**

- 1. Les menus ..... 201
  - 1.1 Définition du menu ..... 201
  - 1.2 Prise en compte par l'activité..... 205
- 2. Le navigation drawer ..... 209
  - 2.1 Modification du layout ..... 209
  - 2.2 Prise en charge par l'activité..... 211
  - 2.3 Manipuler le panneau de navigation..... 216
- 3. Afficher une boîte de dialogue standard ..... 218
- 4. Créer des boîtes de dialogue personnalisées ..... 224

# 6 \_\_\_\_\_ Développez une application Android

Programmation en Java sous Android Studio

## Chapitre 9

### Tâches asynchrones et services

1. Exécuter des actions en tâche de fond. . . . .	229
2. Développer un service . . . . .	242
3. Communiquer avec un service . . . . .	245
4. Utiliser les récepteurs d'évènement. . . . .	248
4.1 Définir un récepteur d'évènement. . . . .	249
4.2 Intention et filtre d'intention . . . . .	249
4.3 Inscrire le récepteur d'évènement . . . . .	250

## Chapitre 10

### Réseau et Internet

1. Présentation de Volley . . . . .	253
1.1 L'application LocDVD . . . . .	254
1.2 Intégrer la bibliothèque Volley . . . . .	254
2. Interroger un service web. . . . .	259
2.1 Préparation . . . . .	259
2.2 Demander les permissions . . . . .	262
2.3 Interrogation du service web. . . . .	263
3. Travailler avec le format JSON . . . . .	270
3.1 Interprétation du format JSON. . . . .	270
3.2 Création de la liste . . . . .	272
3.3 Vue détaillée . . . . .	281
3.4 Optimisations possibles . . . . .	295
4. Intégrer un navigateur . . . . .	297

## **Chapitre 11** **Exploiter le téléphone**

1. Envoyer/recevoir des SMS .....	301
1.1 Envoyer un SMS .....	301
1.2 Recevoir un SMS .....	307
2. Utiliser les capteurs de l'appareil .....	310
3. Géolocaliser l'utilisateur .....	313
3.1 LocationManager .....	314
3.2 Location .....	317

## **Chapitre 12** **Sortir de l'application**

1. Développer un widget .....	319
2. Investir la barre de notification .....	331
3. Partager, utiliser les réseaux sociaux .....	336

## **Chapitre 13** **Design avancé**

1. Mettre en place un thème, utiliser les styles .....	343
2. Créer des images redimensionnables .....	349
3. Dessiner en XML .....	353
4. Animer les transitions d'écrans .....	357



# 8 \_\_\_\_\_ Développez une application Android

Programmation en Java sous Android Studio

## Chapitre 14 Images, son et vidéo

1. Prendre une photo .....	365
1.1 Préparation .....	365
1.2 Implémenter la prise de vue .....	374
1.3 Sauvegarder le résultat .....	376
2. Jouer un son .....	384
2.1 Lire un fichier son local .....	384
2.2 Lire un flux sonore .....	385
3. Jouer une vidéo .....	388

## Chapitre 15 Publier une application

1. Ouvrir un compte développeur .....	393
2. Préparer la fiche .....	396
3. Publier un APK .....	400
4. ... Et ensuite .....	404
5. Pour aller plus loin .....	404

Index .....	407
-------------	-----

## Chapitre 9

# Tâches asynchrones et services

### 1. Exécuter des actions en tâche de fond

Pour toute application, le confort et l'expérience utilisateur sont des points essentiels : une application doit, sur smartphone et tablette, réagir immédiatement à chaque sollicitation de l'utilisateur. Pour garantir une réponse optimale, la plateforme Android introduit une règle : toute application qui ne réagit pas à une demande utilisateur dans un délai de 10 secondes est réputée ne pas répondre. Dans une telle situation, une erreur ANR, pour *Application Not Responding* (« l'application ne répond pas »), est levée, et l'application est susceptible d'être arrêtée par le système.

Pour les opérations qui peuvent potentiellement prendre du temps (dont la réponse n'est pas immédiate), il est donc fortement recommandé d'effectuer des traitements asynchrones : l'opération est exécutée en arrière-plan, et l'utilisateur peut être prévenu de la progression de l'opération.

Bien qu'il soit possible de mettre en place un tel mécanisme en utilisant les classiques classes de gestion des threads de Java, la plateforme fournit une classe abstraite, `android.os.AsyncTask`, qui prend en charge la majeure partie de la mise en place d'une solution asynchrone, et allège d'autant le travail du développeur.

■ `AsyncTask<Params, Progress, Result>`

Les types génériques `Params`, `Progress`, `Result` représentent, comme détaillé ci-dessous, les types de données passés en paramètre ou retournés par les méthodes exposées par la classe.

Pour simplifier la conception d'une opération d'arrière-plan, `AsyncTask` sépare le traitement en plusieurs phases, chacune étant représentée par une méthode.

- `void onPreExecute()` : cette méthode est exécutée au lancement de la tâche asynchrone. Elle s'exécute sur le thread principal, ce qui permet de manipuler les composants de l'interface utilisateur. Il ne faut pas invoquer cette méthode directement, mais appeler la méthode `execute`, qui lance le traitement.
- `Result doInBackground(Params... params)` : cette méthode est abstraite, elle doit obligatoirement être surchargée, et s'exécute dans un thread d'arrière-plan, lorsque la méthode `onPreExecute` est terminée. C'est dans cette méthode que le traitement doit être effectué, aucune opération sur les composants de l'interface n'étant par ailleurs possible. `doInBackground` prend en paramètre un ensemble de données de type générique `Params` et doit renvoyer en retour un objet de type générique `Result`.
- `void onPostExecute(Result result)` : cette méthode est invoquée après la méthode `doInBackground`. Elle prend en paramètre l'objet de type générique `Result` renvoyé par `doInBackground`. Cette méthode étant exécutée par le thread principal, la manipulation des composants de l'interface utilisateur est possible.
- `void onProgressUpdate(Progress... values)` : cette méthode s'exécute sur le thread principal, lorsque la méthode `publishProgress` est invoquée par la méthode `doInBackground`. Elle est typiquement prévue pour gérer l'affichage d'une boîte de dialogue de progression.

Dans le projet `LocDVD`, l'insertion des DVD exemples est une tâche qui peut potentiellement prendre un certain temps : il n'y a ici que quelques DVD, mais on pourrait imaginer fournir un fichier plus étoffé, qui prendrait du temps à être interprété. Il est donc plus que recommandé de transformer ce traitement en traitement asynchrone.

- Éditez le fichier `MainActivity.java` qui contient la méthode effectuant l'insertion des DVD exemples.
- Dans la classe `MainActivity`, définissez une classe `AsyncReadEmbeddedData` qui étend la classe `AsyncTask<String, Integer, Boolean>`.

```
class AsyncReadEmbeddedData extends AsyncTask<String, Integer, Boolean> {  
}
```

- Il faut surcharger les méthodes `onPreExecute`, `doInBackground`, `onProgressUpdate` et `onPostExecute` :

```
class AsyncReadEmbeddedData extends AsyncTask<String, Integer, Boolean> {  
  
    @Override  
    protected void onPreExecute() {  
    }  
  
    @Override  
    protected Boolean doInBackground(String... params) {  
        return false;  
    }  
  
    @Override  
    protected void onProgressUpdate(Integer... values) {  
    }  
  
    @Override  
    protected void onPostExecute(Boolean result) {  
    }  
  
};
```

- La méthode `doInBackground` reprend le code de la méthode `readEmbeddedData` qui effectue la lecture et l'insertion des DVD exemples. Le nom du fichier, au lieu d'être inscrit directement dans le corps de la méthode, est passé en paramètre de `doInBackground`.

Le corps de `doInBackground` est, pour l'instant, le suivant :

```
@Override  
protected Boolean doInBackground(String... params) {  
    String dataFile = params[0];  
    InputStreamReader reader = null;  
    InputStream file=null;
```

```
BufferedReader bufferedReader=null;
try {
    file = getAssets().open(dataFile);
    reader = new InputStreamReader(file);
    bufferedReader = new BufferedReader(reader);
    String line= null;
    while((line=bufferedReader.readLine())!=null) {
        String [] data = line.split("\\|");
        if(data!=null && data.length==4) {
            DVD dvd = new DVD();
            dvd.titre = data[0];
            dvd.annee = Integer.decode(data[1]);
            dvd.acteurs = data[2].split(",");
            dvd.resume = data[3];
            dvd.insert(MainActivity.this);
        }
    }
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if(bufferedReader!=null) {
        try {
            bufferedReader.close();
            reader.close();
            SharedPreferences sharedPreferences =
getSharedPreferences("com.exemple.locDVD.prefs",
Context.MODE_PRIVATE);
            SharedPreferences.Editor editor =
sharedPreferences.edit();
            editor.putBoolean("embeddedDataInserted", true);
            editor.commit();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
return false;
}
```

- Pour informer l'utilisateur de l'avancement de l'initialisation, il faut publier, à chaque insertion, le nombre de DVD insérés dans la base. Pour cela, il faut implémenter un compteur, et invoquer, à chaque itération de la boucle `while`, la méthode `publishProgress` :

```
try {  
  
    int counter = 0;  
  
    file = getAssets().open(dataFile);  
    reader = new InputStreamReader(file);  
    bufferedReader = new BufferedReader(reader);  
    String line= null;  
    while((line=bufferedReader.readLine())!=null) {  
        String [] data = line.split("\\|");  
        if(data!=null && data.length==4) {  
            DVD dvd = new DVD();  
            dvd.titre = data[0];  
            dvd.annee = Integer.decode(data[1]);  
            dvd.acteurs = data[2].split(",");  
            dvd.resume = data[3];  
            dvd.insert(MainActivity.this);  
  
            publishProgress(++counter);  
  
        }  
    }  
} catch (IOException e) {  
    e.printStackTrace();  
}
```

- `doInBackground` doit renvoyer vrai si l'insertion s'est correctement déroulée, et faux dans le cas contraire. Dans le cadre de l'application, il est en outre intéressant de rajouter une temporisation entre l'insertion de deux DVD, pour visualiser plus facilement le traitement en arrière-plan. Une version complète de la méthode est proposée ci-dessous :

```
@Override  
protected Boolean doInBackground(String... params) {  
    boolean result = false;  
    String dataFile = params[0];  
    InputStreamReader reader = null;  
    InputStream file=null;  
    BufferedReader bufferedReader=null;
```

```
try {
    int counter = 0;
    file = getAssets().open(dataFile);
    reader = new InputStreamReader(file);
    bufferedReader = new BufferedReader(reader);
    String line= null;
    while((line=bufferedReader.readLine())!=null) {
        String [] data = line.split("\\|");
        if(data!=null && data.length==4) {
            DVD dvd = new DVD();
            dvd.titre = data[0];
            dvd.annee = Integer.decode(data[1]);
            dvd.acteurs = data[2].split(",");
            dvd.resume = data[3];
            dvd.insert(MainActivity.this);
            publishProgress(++counter);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if(bufferedReader!=null) {
        try {
            bufferedReader.close();
            reader.close();
            SharedPreferences sharedPreferences =
getSharedPreferences("com.exemple.locDVD.prefs",
Context.MODE_PRIVATE);
            SharedPreferences.Editor editor =
sharedPreferences.edit();
            editor.putBoolean("embeddedDataInserted", true);
            editor.commit();
            result = true;
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
return result;
}
```