



Ressourcesinformatiques

Android 7

Les fondamentaux du développement d'applications Java

Nazim BENBOURAHLA

Fichiers complémentaires
à télécharger



Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence ENI de l'ouvrage **RI7AND** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Chapitre 1

La plateforme Android

- 1. Présentation 13
- 2. Historique 14
- 3. Google Play 15
 - 3.1 Création d'un compte développeur 16
 - 3.2 Publication d'une application. 17
 - 3.3 Suivi et mise à jour d'une application 19

Chapitre 2

Environnement de développement

- 1. Environnement Java 21
- 2. Android Studio 22
 - 2.1 Présentation 22
 - 2.2 Installation 23
 - 2.3 Utilisation 23
- 3. SDK Android 25
 - 3.1 Présentation 25
 - 3.2 Configuration 25
 - 3.3 Contenu du SDK 27

4. Émulateur	28
4.1 Présentation	28
4.2 Création	29
4.2.1 Création d'un émulateur à partir d'Android Studio.	29
4.2.2 Création d'un émulateur en ligne de commande	34
4.3 Genymotion	35

Chapitre 3 **Principes de programmation**

1. Architecture Android	37
2. ART (Android RunTime).	38
3. NDK (Native Development Kit)	38
4. APK (Android Package)	39
4.1 Création d'un keystore.	40
4.2 Création d'un APK signé	42
5. Gradle	43
6. Composantes Android	47
6.1 Activity (activité)	47
6.2 Fragment	48
6.3 Service	49
6.4 Broadcast receiver (récepteur d'évènement)	49
6.5 Content provider (fournisseur de contenu)	49
6.6 Intents (intentions)	50
6.6.1 Intent filters (filtres d'intention).	50
6.7 La classe Application	51
7. Cycle de vie d'une activité	52
7.1 État d'une activité	52
7.2 Back stack	53
7.3 Le cycle de vie	54
8. Contexte d'une application	56

9. Manifeste57
9.1 Permissions58

Chapitre 4
Ma première application : HelloAndroid

1. Création du projet59
2. Architecture du projet63
3. Explications65
3.1 Android Manifest66
3.2 Ressources.....67
3.3 Fichier généré69
3.4 Fichier source70
4. Résultat71

Chapitre 5
Création d'interfaces simples

1. Les vues73
1.1 Déclarer des identifiants74
1.2 Spécifier la taille des éléments75
1.3 Combiner avec les activités75
2. Les layouts77
2.1 FrameLayout.....77
2.2 LinearLayout.....78
2.3 RelativeLayout82
2.4 GridLayout86
2.5 ScrollView.....88
2.6 ConstraintLayout.....89

3.	Les ressources	91
3.1	Drawable	91
3.2	Les valeurs (values)	92
3.2.1	Les chaînes de caractères	92
3.2.2	Les tableaux	97
3.2.3	Les dimensions	97
3.2.4	Les couleurs	98
4.	Les éléments indispensables	99
4.1	Zone d'affichage d'un texte	99
4.2	Zone d'édition d'un texte	99
4.3	Bouton	101
4.4	Case à cocher	101
4.5	Image	102
4.6	Gestion du clic	102
5.	Liste et RecyclerView	104
5.1	ListView	104
5.2	Création d'une liste	105
5.3	Personnalisation d'une liste	109
5.4	Mise à jour d'une liste	115
5.5	Optimisation d'une liste (ViewHolder)	115
5.6	RecyclerView	118
5.7	CardView	123
6.	Toolbar	125
6.1	Principe	125
6.2	Couleur des barres de navigation	130
6.3	Gestion des anciennes versions	131
6.4	La navigation avec la Toolbar	133
7.	Exercice	135

Chapitre 6
Permission

- 1. Introduction 137
- 2. Demander une permission 138
- 3. Déclarer vos permissions 141

Chapitre 7
Navigation et gestion des évènements

- 1. Principe 143
- 2. Navigation entre écrans 144
- 3. Passage de données entre écrans 144
 - 3.1 Obtenir un résultat 149
 - 3.2 Parcelable..... 151
- 4. Appeler d'autres applications..... 155
- 5. Exécuter une action 157
- 6. Exercice 158

Chapitre 8
Débogage et gestion des erreurs

- 1. Principes 161
- 2. Android Device Monitor 161
- 3. Les logs 163
 - 3.1 Logcat 163
 - 3.2 Utiliser les logs 164
- 4. Android Lint 165
- 5. ADB (Android Debug Bridge) 166
- 6. Débogage pas à pas..... 167

6 **Android 7**

Les fondamentaux du développement d'applications Java

7. Interaction avec l'émulateur	168
7.1 Simuler des actions sur l'émulateur	168
8. Tests sur le téléphone	170
8.1 Utiliser votre téléphone	170
9. Options de développement	171
10. Tests unitaires	172
11. Tests fonctionnels	174
12. Monkey Stress Test	176

Chapitre 9 **Personnalisation**

1. Material Design	179
1.1 Thèmes	180
1.1.1 Définition	180
1.1.2 Implémentation	182
1.2 État des composants	183
1.2.1 Les différents états	183
1.2.2 Implémentation	183
1.3 Dégradé	185
1.4 Élévation	188
1.5 Ripple Effect	188
2. Polices	190
3. Icônes	194
4. Animations	195
4.1 Tween Animation	195
4.2 Frame Animation	198
4.3 Animation de transition	199
5. Gestion des évènements	201
5.1 Appui sur des touches	201
5.2 Surveiller l'état de la saisie	202

- 5.3 Toucher l'écran 206
- 6. Gestion de la rotation 206
 - 6.1 Gestion du mode paysage 207
 - 6.2 Bloquer la vue dans un seul mode 211
 - 6.3 Gérer manuellement la rotation de l'écran 212
- 7. La bibliothèque de design de Google 213
 - 7.1 Floating Button 213
 - 7.2 TextInputLayout 214

Chapitre 10
Notifications

- 1. Principe 217
- 2. Apparence 218
- 3. Implémentation 220
 - 3.1 Créer une notification 220
 - 3.2 Compatibilité avec les anciennes versions 224
- 4. Les actions 225
- 5. Répondre depuis une notification 226
- 6. Les priorités 227
- 7. La visibilité 227
- 8. La catégorie 228
- 9. Contenu de grande taille 228
- 10. Android Wear 229
- 11. Exercice 233

Chapitre 11

Création d'interfaces avancées

1. Fragment	235
1.1 Cycle de vie d'un fragment	236
1.2 La classe Fragment	237
1.3 Les fragments statiques	237
1.4 Les fragments dynamiques	245
1.5 Gestion des fragments	247
1.6 Gestion des anciennes versions	248
2. Data Binding	248
3. ViewPager	252
3.1 Implémentation	252
3.2 Onglets	255
4. NavigationDrawer	257
5. Les pop-ups	262
5.1 Les toasts	262
5.2 Snackbar	263
5.3 AlertDialog	264
5.4 ProgressDialog	265
5.5 Boîte de dialogue personnalisée	267
6. WebView	268
6.1 Exemple d'une page web distante	268
6.2 Paramètres de la WebView	269
6.3 Gestion du bouton retour	270
6.4 Utilisation de code natif dans du JavaScript	270
7. Préférences	272
8. Interfaces dynamiques	276
9. Création de vues personnalisées	278

Chapitre 12

Persistance et partage de données

- 1. Introduction 283
- 2. SharedPreferences 284
- 3. Stockage interne 286
 - 3.1 Écriture d'un fichier 286
 - 3.2 Lecture d'un fichier 287
 - 3.3 Utilisation de fichiers de cache 288
- 4. Stockage externe 288
 - 4.1 Tester la disponibilité du stockage externe 288
 - 4.2 Accéder aux fichiers d'une application 290
 - 4.3 Accéder aux fichiers partagés 291
- 5. Stockage en base de données 292
- 6. ContentProvider 302
 - 6.1 Créer un ContentProvider 303
 - 6.2 Utiliser un ContentProvider 308
- 7. Partager vos données avec d'autres applications 311
- 8. Recevoir des données depuis d'autres applications 313

Chapitre 13

Traitement en tâche de fond

- 1. Principe 315
- 2. AsyncTask 317
- 3. Thread et Handler 322
- 4. Les services 325
 - 4.1 Créer et utiliser un service 326
 - 4.2 Lier un service à une activité 328

5. Broadcast Receiver	330
5.1 Recevoir un évènement	330
5.2 Envoyer un évènement.	333
6. Alarme	333
6.1 Présentation	333
6.2 Implémentation	334

Chapitre 14

Webservice et Parsing

1. Récupérer des données stockées en ligne	337
1.1 Se connecter au réseau Internet d'un appareil	337
1.2 Gestion du changement de connectivité.	338
1.3 Connexion à une adresse distante.	339
1.4 Parsing XML	341
1.5 Parsing JSON	343
1.6 Retrofit 2.	345

Chapitre 15

Google Maps et géolocalisation

1. Prérequis	349
1.1 Installation des API Google	349
1.2 Récupération de votre clé Google Maps	350
1.2.1 Génération de votre empreinte SHA-1	350
1.2.2 Récupération de votre clé.	352
2. Intégration d'une Google Map.	354
2.1 Création de la vue Google Maps	354
2.2 Ajout d'options à la Google Map	358
2.2.1 Définir le niveau de zoom	358
2.2.2 Affichage en mode satellite	358
3. Localisation	359

4. Placement d'un marqueur sur la carte 363
5. Conversion position/adresse 365

Chapitre 16
Téléphonie et matériel

1. Obtention d'informations sur les caractéristiques du téléphone . . 367
2. Gestion des appels et des messages 368
 2.1 Gestion des appels 369
 2.1.1 Passer un appel 369
 2.1.2 Gérer les appels entrants 370
 2.2 Gestion des messages 371
 2.2.1 Envoi de SMS 371
 2.2.2 Réception d'un message 374
3. Caméra 375
 3.1 Utiliser l'application Caméra de l'appareil 375
 3.2 Contrôler la caméra 381
4. Les capteurs sous Android 382
 4.1 Principe 382
 4.2 Accéléromètre 383
 4.3 Gyroscope 387
 4.4 Capteur magnétique 388
5. Bluetooth 390
 5.1 Activer le Bluetooth 391
 5.2 Permettre à d'autres appareils de se connecter 392
6. NFC 395
7. TTS (Text To Speech) 398

Chapitre 17 Aller plus loin

1. Widget	401
1.1 Présentation	401
1.2 Implémentation	402
2. Gradle avancé	407
3. Java 8	408
4. Firebase	409
5. Bonnes pratiques	411
5.1 Être indépendant de la résolution de l'écran	411
5.2 Être indépendant de la taille de l'écran	411
5.3 Être indépendant de la version d'Android utilisée	412
5.4 Être performant	412
6. Optimiser ses interfaces	413
6.1 Inspecter la hiérarchie de ses interfaces	413
6.2 Fusionner des layouts	415
6.3 Inclure des vues	416
6.4 Chargement paresseux (Lazy Loading) des layouts	417
7. Mise à jour d'une application vers Nougat	419
Index	421

Chapitre 5

Création d'interfaces simples

1. Les vues

La création d'une interface sous Android peut s'effectuer de deux manières :

- **La création statique**, qui s'effectue en XML.
- **La création dynamique**, qui s'effectue en Java.

■ Remarque

On peut combiner ces deux méthodes pour créer des interfaces plus complexes (cf. chapitre Création d'interfaces avancées - Interfaces dynamiques).

Une interface se compose :

- **D'un ou plusieurs fichiers XML** : ils représentent la partie statique d'une interface. Elle est constituée de différents éléments (bouton, texte, zone d'édition, etc.).
- **D'un fichier JAVA (Activité)** : il représente la partie dynamique d'une interface, les interactions utilisateur et les traitements à effectuer, etc.

■ Remarque

*Tous les éléments basiques d'une vue (bouton, zone de texte...) héritent de la classe **View**.*

Modifier une vue peut s'effectuer de deux manières :

- Mettre à jour le code XML de l'interface (onglet **Text** sous Android Studio).
- Mettre à jour la vue à l'aide de l'éditeur d'interface (onglet **Design** sous Android Studio).

1.1 Déclarer des identifiants

Un identifiant correspond à un nom unique affecté à un élément d'une vue. Grâce à cet identifiant, vous pouvez mettre en place les interactions et les traitements pour l'élément possédant cet identifiant.

Pour associer un identifiant à un élément d'une vue, il faut utiliser l'attribut suivant :

```
■ android:id="@+id/nom_identifiant"
```

La déclaration d'un identifiant se compose de plusieurs éléments :

- **android:id** : nom de l'attribut.
- **@+** : indique la déclaration d'un nouvel identifiant.
- **id** : correspond à la catégorie « l'identifiant ».
- **nom_identifiant** : correspond à l'identifiant d'un élément.

La syntaxe suivante permet d'accéder à l'identifiant d'un élément depuis un fichier Java :

```
■ R.id.nom_identifiant
```

ou depuis un fichier XML :

```
■ @id/nom_identifiant
```

1.2 Spécifier la taille des éléments

À chaque déclaration d'un élément d'une vue (conteneur ou composant), vous devez spécifier sa hauteur et sa largeur (**android:layout_height** et **android:layout_width**).

Vous pouvez spécifier ces valeurs de plusieurs manières :

- **match_parent** : signifie que la taille de l'élément est égale à celle de l'élément parent.
Par exemple, un bouton possédant une largeur définie à `match_parent` occupera le même espace que son conteneur.
- **wrap_content** : signifie que la taille de l'élément est égale à celle de son contenu.
Par exemple, un bouton possédant une largeur définie à `wrap_content` aura pour taille la somme de la taille de son contenu et des différents espaces internes (`padding`).
- **en spécifiant une valeur** : vous pouvez définir la taille d'un élément à l'aide de valeurs fixes.

■ Remarque

Il faut spécifier la taille des éléments en dp (density-independent pixels) et non en px. Les tailles spécifiées en dp conservent les mêmes proportions quelle que soit la densité de l'écran.

1.3 Combiner avec les activités

Une fois la partie statique (xml) d'une interface déclarée, il faut créer une classe Java représentant votre activité.

■ Remarque

Chaque nouvelle activité créée doit être déclarée dans le manifeste de l'application.

Cette classe doit :

- Hériter de la classe **AppCompatActivity**.
- Surcharger au minimum la méthode **onCreate** (cf. chapitre Principes de programmation - Cycle de vie d'une activité).
- Lier l'activité à l'interface à l'aide de la méthode **setContentView**.

► Pour créer une nouvelle activité, faites un clic droit sur le dossier **src** de votre projet, puis sélectionnez l'option **New - Activity** et choisissez le type d'activité que vous voulez créer (**Blank Activity**, **Login Activity**, etc.).

Prenons l'exemple d'une interface créée dans le fichier **home.xml**. Pour pouvoir la lier à une activité, la méthode **onCreate** doit contenir au minimum le code ci-dessous :

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

Remarque

*Vous pouvez remarquer l'utilisation du fichier **R.java** pour récupérer le layout voulu.*

► N'oubliez pas de déclarer votre activité dans le fichier manifeste de votre application. La déclaration de nouveaux composants (activité, service...) s'effectue entre les balises **<application>**.

```
<activity android:name="chemin.package.MyNewActivity"
android:label="@string/activity_title">
```

► Vous pouvez lui spécifier des propriétés ou comportements à l'aide des filtres d'intention. Les filtres d'intention se divisent en plusieurs catégories :

- **Les actions** (balise **action**) : permettent de spécifier des actions (comportements) à un composant, par exemple : **ACTION_CALL** (pour passer un appel téléphonique), **ACTION_MAIN** (activité principale de l'application), **ACTION_SEND** (utilisé pour le partage de données), etc.
- **Les données** (balise **data**) : permettent de spécifier le type de données traité par le composant.

- **Les catégories** (balise **category**) : permet de spécifier la catégorie du composant, par exemple **CATEGORY_BROWSABLE** (peut être invoqué par le navigateur pour afficher des données), **CATEGORY_LAUNCHER** (l'activité sera disponible depuis le lanceur d'application), etc.
- **Les extras** : représente des données additionnelles qui seront fournies à l'activité. Par exemple, pour l'envoi d'un e-mail, utiliser la clé **EXTRA_EMAIL** pour spécifier le destinataire du mail.
- Différents flags utiles à l'activité.

Par exemple :

```
<activity android:name=".MyActivity"
          android:label="@string/activity_title">

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

</activity>
```

2. Les layouts

Les layouts facilitent l'organisation des différents éléments qui composent une interface. Ils servent de conteneur aux composantes d'une vue. Tous les layouts Android héritent de la classe **ViewGroup**.

■ Remarque

La classe **ViewGroup** hérite de la classe **View**.

2.1 FrameLayout

Le Framelayout est le conteneur le plus simple, il représente un espace qui affiche l'objet de votre choix.

Un élément ajouté à un FrameLayout se positionne en haut à gauche du layout. Vous pouvez changer cette position à l'aide de l'attribut **android:gravity**.

Vous avez la possibilité d'ajouter plusieurs éléments dans un même Framelayout et de modifier la visibilité de ces éléments pour afficher ou cacher plusieurs éléments au même emplacement.

2.2 LinearLayout

Le LinearLayout permet d'aligner des éléments (dans l'ordre des déclarations) dans une direction (verticale ou horizontale).

Vous pouvez définir les attributs suivants :

- L'orientation du layout.
- La gravité des éléments.
- Le poids des éléments.

Orientation

À la création d'un LinearLayout, vous devez préciser son orientation (horizontale ou verticale) à l'aide de l'attribut **android:orientation**.

■ Remarque

L'orientation possède par défaut la valeur horizontale.

Positionnement d'un élément

Pour définir le positionnement d'un élément dans un LinearLayout, deux attributs sont disponibles :

- **layout_gravity** : spécifie le positionnement d'un élément dans son conteneur.
- **gravity** : spécifie le positionnement du contenu d'un élément (par exemple, on peut spécifier la position d'un texte dans un bouton).