Développement Android

Jean-François COUCHOT
couchot[arobase]femto-st[point]fr

8 février 2019

Table des matières

Chapitre 1

Projet sous Android Studio et interfaces utilisateur

1.1 Projet Android. Organisation d'interface

Exercice 1.1. *Organisation d'un projet Android.*

1. *Construire un projet Android pour téléphone et tablette à base de « Navigation Drawer » et comprendre les options proposées.*
2. *Dans la vue « Project » d'Android Studio, étudier chaque dossier et sous-dossier.*
3. *Comprendre le chaîne de compilation d'un projet Android présentée à l'adresse :* `https://developer.android.com/studio/build/index.html`.
4. *Représenter graphiquement à l'aide d'un arbre la hiérarchie des composants de l'interface utilisateur à partir de* `setContentView(R.layout.activity_main)` *du fichier* `MainActivity`.

A la fin de cette section vous devez avoir compris :
— ce que contient un projet Android notamment, le manifest, les fichiers gradle, les ressources, les sources ;
— comment est compilée un projet Android ;
— comment est organisée le projet basé sur « Navigation Drawer ».

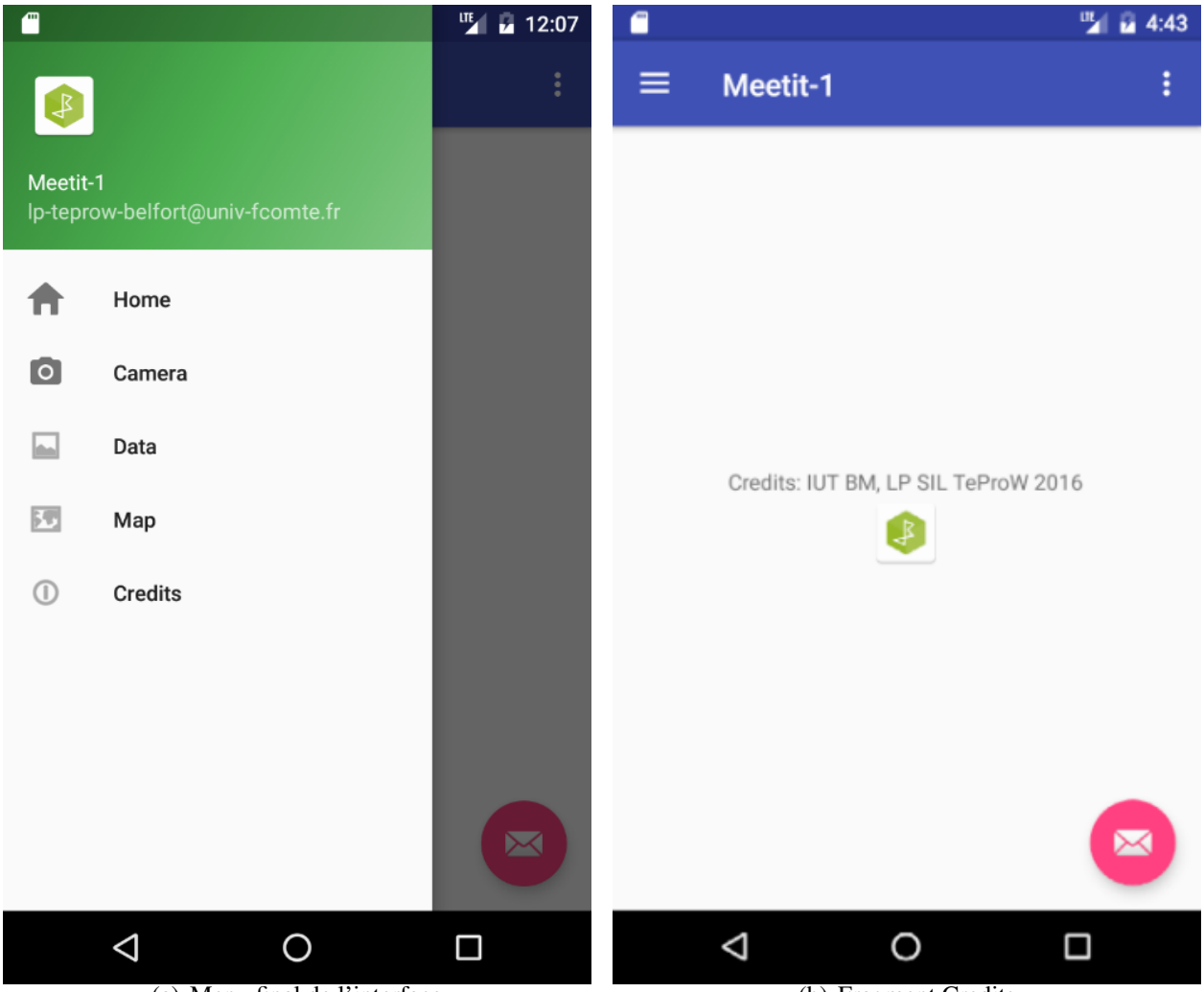
1.2 Ressources

Exercice 1.2. *Modification de l'entête du menu de navigation. Tout ce qui suit concerne l'entête du menu de navigation, c'est à dire la partie supérieure gauche de la figure 1.1(a).*

1. *Repérer le fichier xml qui définit ceci.*
2. *Modifier les fichiers* `strings.xml` *et* `nav_header_main.xml` *pour faire en sorte que le nom de l'application et votre adresse mail apparaissent en lieu et place de l'existant.*
3. *Pour modifier le logo (et d'une manière générale ajouter une image) :*
   (a) *Récupérer un logo en haute définition.*
   (b) *Demander la création d'une nouvelle* `Image Asset`.
   (c) *Préciser que c'est une icône de lancement, construite à partir d'une image dont on dira où elle est enregistrée et lui donner le nom* `ic_launcher`.
   (d) *Vérifier que l'image produite a été ajoutée dans le dossier mipmap sous différentes tailles.*
   (e) *Dans l'élément* `ImageView`, *renseigner* `@mipmap/ic_launcher`.

Exercice 1.3. *Modification du contenu du menu de navigation. Tout ce qui suit concerne le contenu du menu de navigation, représenté dans la partie gauche de la figure 1.1(a).*

1. *Repérer le fichier xml qui définit ceci.*
2. *Modifier le contenu du menu de navigation pour avoir le même menu que celui présenté à la figure 1.1(a). Les identifiants des menus doivent être respectivement :* `nav_home`, `nav_camera`, `nav_data`, `nav_map` *et* `nav_credits`.
3. *Modifier l'activité* `MainActivity` *pour qu'elle accepte ces identifiants : ceci se fait dans la méthode* `onNavigationItemSelected`.



(a) Menu final de l'interface (b) Fragment Credits

FIGURE 1.1 – Premières interfaces de l'application

A la fin de cette section vous devez avoir compris :
— où sont stockées les ressources d'un projet Android et sous quelles formes,
— comment en ajouter, comment les modifier.

1.3 Activités et Fragments

Exercice 1.4. *Migration vers les fragments.*

L'objectif de cet exercice est ne plus avoir de TextView affichant "Hello World", mais d'avoir un fragment qui effectue ceci dynamiquement.

1. *Dans le layout contenant le TextView affichant "Hello World", supprimer ce TextView.*

2. *Repérer l'identifiant* `content_main` *dans le RelativeLayout de ce fichier ; s'il n'y est pas (dépend de la version de Studio) préciser que ce RelativeLayout a pour identifiant* `@+id/content_main` *puis réexécuter le programme. Que constatez-vous ?*
3. *Demander à Studio de créer un nouveau fragment blanc, nommé HomeFragment. Constater la création de la classe HomeFragment et du layout* `fragment_home.xml`.
4. *On va demander qu'à la création l'activité principale MainActivity, le fragment précédemment défini soit positionné à la place de l'élément identifié par* `content_main` *vu à la question 2. Pour cela :*
   (a) *Préciser que l'activité MainActivity interagit avec ce fragment*

```
public class MainActivity extends AppCompatActivity
  implements NavigationView.OnNavigationItemSelectedListener,
   HomeFragment.OnFragmentInteractionListener
```

   et demander à Studio d'implanter la méthode requise.

   (b) *Créer les deux attributs privés de MainActivity après avoir compris leur sens :*

```
private FragmentManager fm = null;
private Fragment fragment = null;
```

   (c) *Dans sa méthode* `onCreate`*, après* `setContentView(R.layout.activity_main)`*, ajouter les lignes suivantes après avoir compris leur sens :*

```
fm = getSupportFragmentManager();
fragment = new HomeFragment();
fm.beginTransaction().replace(R.id.content_main,fragment).commit();
```

   Constater que l'objectif est atteint.

Exercice 1.5. *Remplacement d'un fragment par un autre.*

L'objectif de cet exercice est de faire remplacer le fragment `HomeFragment` *par le fragment nommé* `CreditsFragment` *lorsque l'utilisateur valide l'item « Credits » du menu latéral.*

1. *Le fragment* `CreditsFragment` *contient les éléments graphiques représentés à la figure 1.1(b).*
2. *Reprendre partiellement l'exercice précédent pour réaliser ceci. Le code à exécuter lorsque l'utilisateur choisit un item du menu latéral se trouve dans la méthode* `MainActivity.onNavigationItemSelected`.
3. *Améliorer votre code en remplaçant la cascade de if dans cette méthode par un* `switch....case`.
4. *Faire de même avec* `MyCameraFragment`, `MyDataFragment` *et* `MyMapFragment`.

A la fin de cette section vous devez avoir compris :
— ce qu'est une activité, ce qu'est un fragment, leurs points communs, leurs différences ;
— comment ajouter une activité et ce que cela implique ;
— comment ajouter un fragment et comment le charger dans une activité.

1.4 Une liste de Fragments avec `RecyclerView`

Il est souvent intéressant d'afficher des données sous la forme d'une liste. Lorsque la liste est longue, pour des raisons d'efficacité, le système ne devrait créer que les vues visibles à l'écran. Lorsque que l'on fait défiler les vues, celles qui disparaissent devraient être « recyclées » pour être réutilisées. C'est le principe des `RecyclerView`.

Les personnages de l'application vont être affichés dans le fragment `MyDataFragment`. Ils seront extraits d'une base de données au chapitre suivant. Dans cette section on ne va s'intéresser qu'à la partie affichage pour laisser le stockage des données au prochain chapitre.

1.4.1 Un objet `Character` et son fragment pour chaque personnage

Exercice 1.6. *Construction du modèle et de sa vue*

1. *Construire la classe* `Character` *avec les attributs privés suivants :*
   — `Int uid //ce sera la clef primaire de la base de données`,
   — `String firstname`,
   — `String familyname`,
   — `String weburl`,
   — `float latitude`,
   — `float longitude`,
   — `String bmppath`.
2. *Générer les getters, les setters et 2 constructeur : le premier pendra tous les attributs tandis que le second les prendra tous, sauf* `uid`.
3. *Construire un gabarit nommé* `fragment_character` *permettant l'affichage d'un personnage comme donné à la figure 1.2.*
4. *Fixer les identifiants des vues comme suit :*
   — `@+id/image_view_character_picture` *pour l'ImageView qui contient la photo du personnage ;*
   — `@+id/text_view_character_first_name` *pour le TextView contenant le prénom ;*
   — `@+id/text_view_character_familiy_name` *pour le TextView contenant le nom ;*
   — `@+id/text_view_character_latitude` *pour le TextView contenant la latitude ;*
   — `@+id/text_view_character_longitude` *pour le TextView contenant la longitude.*

FIGURE 1.2 – Le gabarit des personnages

1.4.2 Des objets pour afficher ces vues

Chaque personnage va être affiché au moyen d'un `RecyclerView.ViewHolder`.

Exercice 1.7. *Un conteneur de vue dédié pour chaque personnage*

1. *Construire la classe* `CharacterViewHolder` *qui étend* `RecyclerView.ViewHolder` *en insérant le code suivant :*

```
public class CharacterViewHolder extends RecyclerView.ViewHolder{
  public TextView firstNameView, familyNameView, latitudeView, longitudeView;
  public ImageView pictureView;


  public CharacterViewHolder(View itemView) {
```

```
      super(itemView);
      firstNameView = (TextView) itemView.findViewById(R.id.text_view_character_first_name);
      familyNameView = (TextView) itemView.findViewById(R.id.text_view_character_familiy_name);
      latitudeView = (TextView) itemView.findViewById(R.id.text_view_character_latitude);
      longitudeView = (TextView) itemView.findViewById(R.id.text_view_character_longitude);
      pictureView = (ImageView) itemView.findViewById(R.id.image_view_character_picture);
    }
  }
```

2. *Quel est l'objectif d'un tel objet ?*

Il faut construire un adaptateur (`RecyclerView.Adapter`) pour afficher uniquement les vues visibles de la liste de tous les personnages.

Exercice 1.8. *Un adaptateur pour la liste des personnage*

1. *Construire la classe* `ChraracterListAdapter` *qui étend* `RecyclerView.Adapter<CharacterViewHolder>` *et demander à studio d'implanter les méthodes requises.*
2. *ajouter l'attribut* `List<Character> characterList` *et un constructeur permettant d'initialiser cet attribut.*
3. *Que contiendra cet attribut ?*
4. *Remplacer la méthode vide* `onCreateViewHolder(ViewGroup parent, int viewType)` *par le code*

```
public CharacterViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
  View itemView = LayoutInflater.from(parent.getContext())
                  .inflate(R.layout.fragment_character,parent,false);
  return new CharacterViewHolder(itemView);
}
```

   Cette méthode construit le conteneur de la vue du personnage et le retourne.
5. *Modifier le code de la méthode* `getItemCount()` *pour qu'il retourne la taille de la liste des personnages.*
6. *Dans la méthode* `onBindViewHolder(CharacterViewHolder holder, int position)`
   (a) *Récupérer le personnage* `Character teacher` *de la liste* `characterList` *situé à l'indice* `position`
   (b) *Modifier le conteneur reçu en paramètre en fixant son prénom avec celui du personnage récupéré à la question précédente :*
   `holder.firstNameView.setText(teacher.getFirstname());`
   (c) *Faire de même avec toutes les autres vues du conteneur.*

1.4.3 Le Fragment `MyDataFragment`

Dans le Fragment `MyDataFragment` il ne reste plus qu'à
— créer une `RecyclerView` dans le gabarit qui contiendra la liste déroulante et
— modifier le créateur de vue de ce fragment pour lui dire comment remplir correctement cette vue.

Exercice 1.9. *Une* `RecyclerView` *en plus*

1. *Ajouter le code suivant dans le gabarit de* `MyDataFragment`.

```
<android.support.v7.widget.RecyclerView
  android:id="@+id/character_recycler_view"
  android:layout_width="match_parent"
  android:layout_height="match_parent" />
```

2. *Laisser Studio modifier le build.gradle pour importer la bibliothèque* `com.android.support:recyclerview-v7`.

Exercice 1.10. *Paraméter la* `RecyclerView` *de* `MyDataFragment`

1. *Ajouter les attributs suivants dans* `MyDataFragment`
   — `private List<Character> characterList = new ArrayList<>();`
   — `private RecyclerView recyclerView;`
2. *Modifier la méthode* `onCreateView` *comme suit :*

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
  View v = inflater.inflate(R.layout.fragment_my_data, container, false);

  recyclerView =  v.findViewById(R.id.character_recycler_view);
  recyclerView.setLayoutManager(new LinearLayoutManager(getActivity().getApplicationContext()));
  recyclerView.setItemAnimator(new DefaultItemAnimator());
  recyclerView.addItemDecoration(new DividerItemDecoration(getContext(),
                                   LinearLayoutManager.VERTICAL));
  recyclerView.setAdapter(new CharacterListAdapter(characterList));

  prepareCharacterData();

  return v;
}
```

3. *Comprendre chaque ligne du code précédent.*
4. *Développer la méthode* `prepareCharacterData` *qui ajoute trois personnages dans la liste* `characterList` *selon le modèles suivant :*

```
characterList.add(new Character("Stéphane","Domas",
                                "http://info.iut-bm.univ-fcomte.fr/staff/sdomas/",
                                47.6387143f, 6.8370225f,
                                "domas.png");
```

A la fin de cette section, vous devriez avoir compris les bases de `RecyclerView` avec ses deux parties `RecyclerView.ViewHolder` et `RecyclerView.Adapter`.

1.5 Paramètres

Exercice 1.11. *introduction aux paramètres de l'application. L'objectif est de construire une interface permettant à l'utilisateur de saisir ses préférences vis à vis de l'application. Ceci se fait classiquement en construisant une activité de paramètres (Settings), comme représentée à la figure 1.3(b) et qui s'affiche lorsque l'utilisateur clique sur les ":" en haut à droite de l'application.*

1. *Demander à Studio de créer une nouvelle activité de type* `Settings Activity`. *Constater l'ajout :*
   — *de la classe abstraite* `AppCompatPreferenceActivity` *qu'on ne modifiera pas ;*
   — *de la classe* `SettingsActivity` *; constater que cette classe définit aussi les classes* `GeneralPreferenceFragment`, `NotificationPreferenceFragment` *et* `DataSyncPreferenceFragment` *;*
   — *du dossier* `xml` *de ressources ;*
   — *des layout* `pref_general.xml`, `pref_data_sync.xml`, `pref_notification.xml` *et* `pref_ headers.xml` *enregistrés dans le dossier* `xml` *; ces gabarit définissent l'affichage des fragments présentés ci-avant.*
2. *Modifier la méthode* `MainActivity.onOptionsItemSelected` *pour qu'elle demande à Android le démarrage de l'activité* `SettingsActivity`. *Il suffit pour cela d'adapter le code comme suit, qu'on doit comprendre :*

```
if (id == R.id.action_settings) {
  Intent intent = new Intent(this, SettingsActivity.class);
  startActivity(intent);
  return true;
}
```

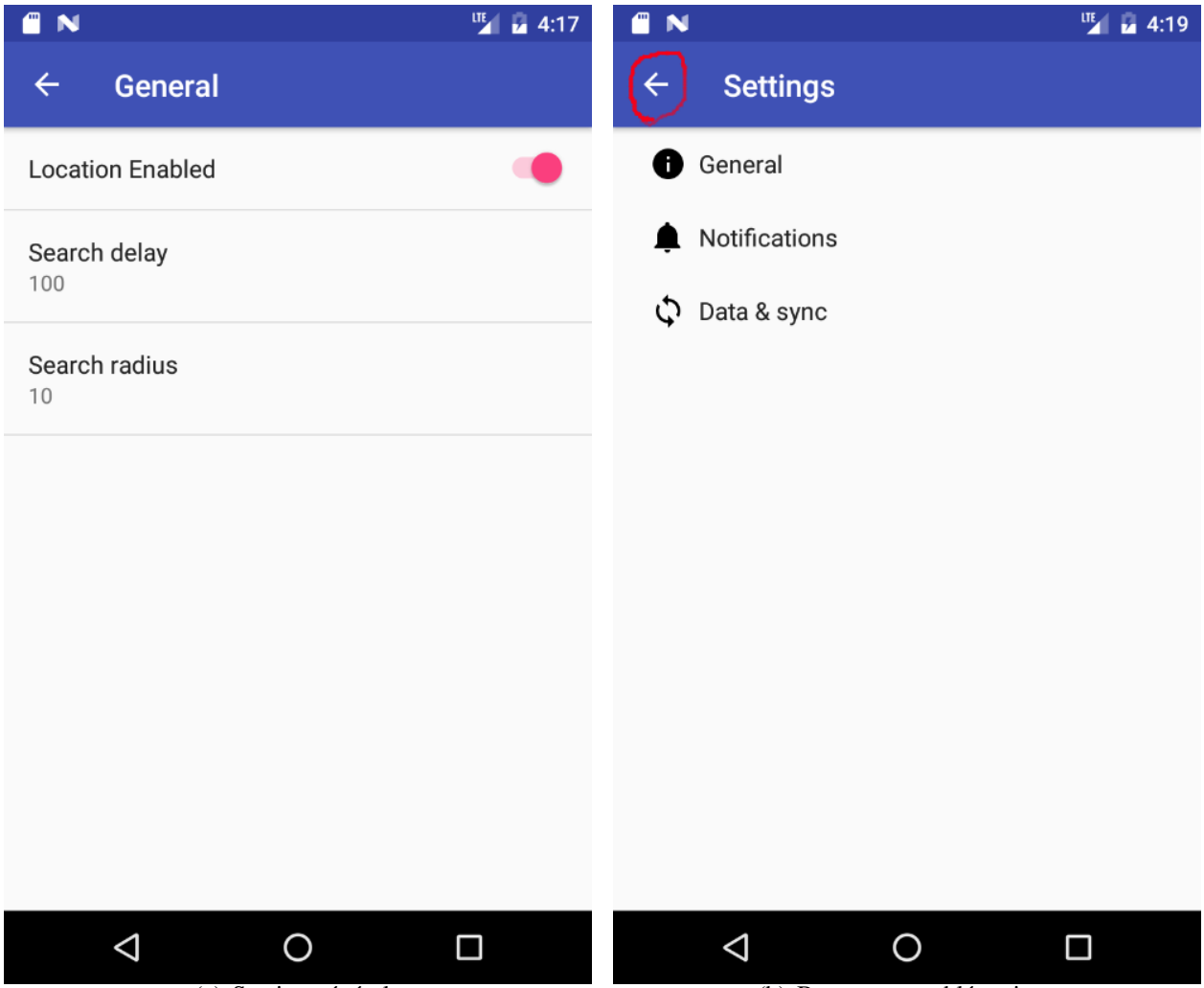
Exercice 1.12. *Modification des préférences par défaut.*

1. *A l'aide de Studio, remplir l'élément* `PreferenceScreen` *du fichier* `pref_general.xml` *comme présenté à la figure figure 1.3. Les libellés seront stockés dans le fichier* `strings.xml` *et les valeurs par défaut des éléments présentés sont* `false`*, 100 et 10 respectivement.*
2. *Faire en sorte que l'interrupteur* `Location enabled` *active la possibilité de saisir le rayon de recherche ainsi que le délai entre deux demandes de localisation.*
3. *L'invocation de* `bindPreferenceSummaryToValue` *dans la méthode* `onCreate` *de la classe* `GeneralPreferenceFragment` *peut générer des erreurs. Corriger le code pour les supprimer.*

Exercice 1.13. *Correction d'un bug dans les préférences.*

Constater qu'appuyer sur le bouton ← de la barre de `Settings` *ne permet pas de revenir à l'activité principale (cf figure 1.3(b)). Ajouter la méthode* `SettingsActivity.onOptionsItemSelected` *comme suit (et la comprendre) :*

```
public boolean onOptionsItemSelected(MenuItem item) {
  int id = item.getItemId();
  if (id == android.R.id.home) {
    this.finish();
    return true;
  }
  return super.onOptionsItemSelected(item);
}
```



(a) Section générale (b) Bouton ← problématique

FIGURE 1.3 – Interfaces de Paramètres

A la fin de cette section vous devez avoir compris :
— ce qu'est une activité de type Settings,
— comment la démarrer, comment l'arrêter,
— comment modifier les paramètres à mémoriser, particulièrement leurs types, leurs valeurs.