Chapitre 6
Android : les éléments incontournables

1. Les listes

1.1 Présentation

Afficher une liste pour présenter des informations est incontournable dans le développement d'une application. Il y a deux formes de listes fréquemment utilisées :

– ListView : toutes les données sont chargées dès la création de la liste.

– RecyclerView : seulement les données visualisées sont chargées.

Il est aujourd'hui souvent recommandé d'utiliser les RecyclerView pour afficher des listes de données. Les ListView sont présentées dans un but pédagogique étant donné que c'est la façon de faire historique et idéale pour débuter.

Les CardView seront aussi abordées dans ce chapitre, car elles sont indissociables des listes. Elles permettent de perfectionner le style des listes.

1.2 ListView

1.2.1 Présentation

Pour créer une liste à l'aide des ListView il est nécessaire de passer par quatre étapes :

- Étape 1 : définir un élément de type ListView dans l'IHM d'une activité ou d'un fragment.
- Étape 2 : définir le style des lignes d'informations affichées dans la liste. Ceci se fait dans un fichier de type layout.
- Étape 3 : définir une classe adapter qui permettra de lier les données au style. L'adapter va créer autant d'éléments View que de données.
- Étape 4 : lier l'adapter à la ListView.

1.2.2 Définir une ListView

La balise `ListView` permet de définir la liste. Elle est associée à l'identifiant `la_liste_view`, cet identifiant est utilisé dans le code de l'activité dans le but de récupérer un objet représentant la ListView. L'objet sera ensuite utilisé pour y lier l'adapter.

Remarque

Les exemples sont réalisés dans un nouveau projet dont l'activité principale se nomme MainActivity.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:
android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="fr.acos.demolistviewwithkotlin.MainActivity">

    <ListView
        android:id="@+id/la_liste_view"
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </ListView>

</android.support.constraint.ConstraintLayout>
```

Cet exemple présente comment définir une ListView dans un layout. En gras, les éléments intéressants et relatifs à la ListView.

1.2.3 Définir le style des éléments de la liste

Le fichier style_dune_ligne.xml contenant le style des éléments de la liste se situe dans le dossier res/layout. Chaque élément de la liste affiche un nom et un prénom grâce aux vues de type `TextView`.

style_dune_ligne.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <TextView
        android:id="@+id/tv_nom"
        android:text="XXXXX"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_margin="10dp"
        />

    <TextView
        android:id="@+id/tv_prenom"
        android:text="YYYYY"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_margin="10dp"
        />

</android.support.constraint.ConstraintLayout>
```

Cet exemple présente comment définir le style de chaque item/élément de la ListView. En gras, les éléments intéressants et relatifs à la présentation.

1.2.4 Créer une classe adapter

La classe `Personne` permet de créer des objets qui seront affichés dans la liste. On définit cette classe dans le fichier MainActivity.kt. La classe `PersonneAdapter` permet de lier les données (Liste d'objets de type `Personne`) au style (défini dans style_dune_ligne.xml). La classe `PersonneAdapter` est définie dans le fichier MainActivity.kt.

Classe Personne et Classe PersonneAdapter (fichier MainActivity.kt)

```
/**
* Classe BO permettant de stocker les données à afficher
*/
data class Personne(val id: Long, val nom: String, val prenom: String)

/**
 * Classe adapter permettant de charger les données dans la liste
 */
class PersonneAdapter(context: Context?, resource: Int, objects:
MutableList<Personne>?) : ArrayAdapter<Personne>(context,
resource, objects)
{
    //Variable permettant de stocker l'identifiant du layout de
    //style d'une ligne de liste
    private val ma_ligne:Int;

    //Au moment de l'initialisation de l'objet adapter, on récupère
    //l'identifiant du layout passé en paramètre
    //pour le fournir à la variable ma_ligne.
    init
    {
        ma_ligne = resource;
    }

    /**
     * Fonction appelée autant de fois qu'il y a d'objets à afficher
     * dans la liste.
     *
     * @param position index de l'élément de la liste à afficher
     * @param convertView Vue représentant le layout
     * style_dune_ligne.xml
     * @param parent Parent de la liste
```

```
     */
    override fun getView(position: Int, convertView: View?, parent:
ViewGroup?): View
    {
        //Au premier appel on charge le fichier
style_dune_ligne.xml, puis on réutilise convertView
        val uneLigneView = convertView ?:
 LayoutInflater.from(parent!!.getContext()).inflate(ma_ligne,
parent, false)

       //Ces variables représentent les TextView de la ligne à créer.
        val tvNom =
uneLigneView.findViewById<TextView>(R.id.tv_nom)
        val tvPrenom =
uneLigneView.findViewById<TextView>(R.id.tv_prenom)

        //Récupère les données à afficher.
        val personne = getItem(position)

        //Met les données dans les TextView
        tvNom.text = personne.nom
        tvPrenom.text = personne.prenom

        //La ligne est créée, on la retourne.
        return uneLigneView
    }
}
```

Cet exemple présente comment définir un adapter qui permettra de remplir la liste. En gras, les éléments intéressants et relatifs à l'adapter.

1.2.5 Lier l'adapter à la ListView

La classe `MainActivity` se situe dans le fichier MainActivity.kt. La fonction `onCreate` de la classe `MainActivity` est appelée lors de la création de l'activité. Un bouchon (données de test) constitué d'une liste d'objets `Personne` est créé. Un objet `PersonneAdapter` est créé en lui passant en paramètre la liste des objets `Personne` et le style des éléments de la liste. L'adapter est ensuite lié à la ListView.

MainActivity.kt

```
/**
 * Activité qui affiche la ListView.
 */
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        //Bouchon
        val p1 = Personne(1,"Adnet","Romain")
        val p2 = Personne(2,"Roussel","Guenole")
        val p3 = Personne(3,"Baudouin","Vincent")
        val personnes = mutableListOf(p1,p2,p3)

        //Création de l'adapter
        val adapter =
PersonneAdapter(this,R.layout.style_dune_ligne,personnes)

        //On lie l'adapter à la ListView
        la_liste_view.adapter = adapter
    }
}
```

Cet exemple présente comment combiner tous les éléments dans le but de remplir la liste. En gras, les éléments intéressants.

1.3 RecyclerView

1.3.1 Présentation

Pour créer une liste à l'aide des RecyclerView il est nécessaire de passer par cinq étapes :

– Étape 1 : ajouter la dépendance pour utiliser les RecyclerView dans Gradle.
– Étape 2 : définir un élément de type RecyclerView dans l'IHM d'une activité ou d'un fragment.