



# Android-Java

## Les ListViews

# Sommaire

1.1	- Les listes : ListView.....	5
1.1.1	- Présentation.....	5
1.1.1.1	- IHM.....	5
1.1.1.2	- Remplissage.....	7
1.1.2	- Une liste simple dans une classe héritant de Activity.....	8
1.1.2.1	- Objectif.....	8
1.1.2.2	- Diagramme de classes.....	9
1.1.2.3	- Démarche et syntaxes.....	10
1.1.2.4	- Le tableau de Strings.....	11
1.1.2.5	- Le layout.....	12
1.1.2.6	- L'activité.....	13
1.1.2.7	- Récupération du libellé de l'item de la sélection.....	14
1.1.2.8	- Code complet de l'activité.....	15
1.1.3	- Remplissage d'une ListView à partir d'un tableau Java.....	16
1.1.3.1	- Syntaxes.....	18
1.1.3.2	- Codes.....	19
1.1.3.3	- Exercice : prix soldés.....	21
1.1.4	- Une liste avec une icône identique pour chaque item.....	22
1.1.4.1	- Objectif.....	22
1.1.4.2	- Principe et diagramme de composants.....	23
1.1.4.3	- Démarche.....	24
1.1.4.4	- Le layout de ligne.....	24
1.1.4.5	- Syntaxes.....	25
1.1.4.6	- Exercice : mise en place de la ListView avec une icône.....	25
1.1.5	- Une liste avec une structure de composants.....	26
1.1.5.1	- Objectif.....	26
1.1.5.2	- Principe et démarche.....	27
1.1.5.3	- Syntaxes et exemples.....	28
1.1.5.4	- Codes.....	29
1.1.5.5	- Exercice : ajoutez le prénom et l'e-mail.....	31
1.1.6	- Une liste avec une structure mixte : image et textes.....	32
1.1.6.1	- Objectif.....	32
1.1.6.2	- Codes.....	33
1.1.6.3	- Exercice : une liste avec une icône différente pour chaque item.....	37
1.1.7	- Mettre à jour une ListView.....	38
1.1.7.1	- Si toutes ou grande partie des items sont à modifier....	38
1.1.7.2	- Si un seul item est à ajouter.....	38

1.1.7.3	- Supprimer un item.....	38
1.1.8	- Les listes et la sélection multiple.....	39
1.1.8.1	- Objectif.....	39
1.1.8.2	- Syntaxes.....	40
1.1.8.3	- Layout.....	41
1.1.8.4	- Activité.....	42
1.1.8.5	- La même chose mais plus dans le layout.....	44
1.1.8.6	- Sélectionner tous les items d'une ListView.....	47
1.2	- ExpandableListView (Accordéon).....	48
1.2.1	- Objectif.....	48
1.2.2	- Démarche.....	49
1.2.3	- Codes.....	50
1.2.3.1	- Les codes des layouts.....	50
1.2.3.2	- Le code de l'adapter et le code de l'activité.....	52
1.3	- Remplir une ListView avec des images de la SD.....	56
1.4	- Remplir une ListView avec des images du WEB.....	58
1.5	- Annexes.....	59
1.5.1	- Une liste simple dans une classe héritant de ListActivity.....	59
1.5.1.1	- Objectif.....	59
1.5.1.2	- Diagramme de classes.....	60
1.5.1.3	- Démarche et syntaxes.....	61
1.5.1.4	- Codes.....	64
1.5.2	- Comparatif des syntaxes.....	66
1.5.3	- Menu sous forme de ListView ... pour la suite.....	67
1.5.3.1	- L'objectif.....	67
1.5.3.2	- Le layout.....	68
1.5.3.3	- L'activité (version générique).....	69
1.5.3.4	- L'activité (version spécifique).....	71
1.5.4	- Personnalisation d'une ListView : colorisation.....	72
1.5.4.1	- Coloriser les lignes.....	72
1.5.4.2	- Coloriser la sélection.....	74
1.5.4.3	- Changer la taille de la police (ou une autre propriété) des items de la liste.....	76
1.5.5	- Appui long sur un item.....	77
1.5.6	- Simuler un click dans une ListView.....	79
1.5.6.1	- Quand l'activité est de type ListActivity.....	79
1.5.6.2	- Quand l'activité est de type Activity.....	79
1.5.7	- Récupérer le contenu d'une ListView.....	80
1.5.7.1	- Une ListView simple.....	80
1.5.8	- Pré-sélectionner un élément d'une ListView.....	81
1.5.8.1	- Une ListView simple.....	81

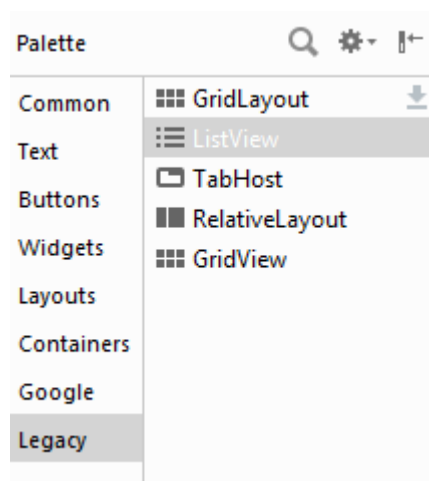
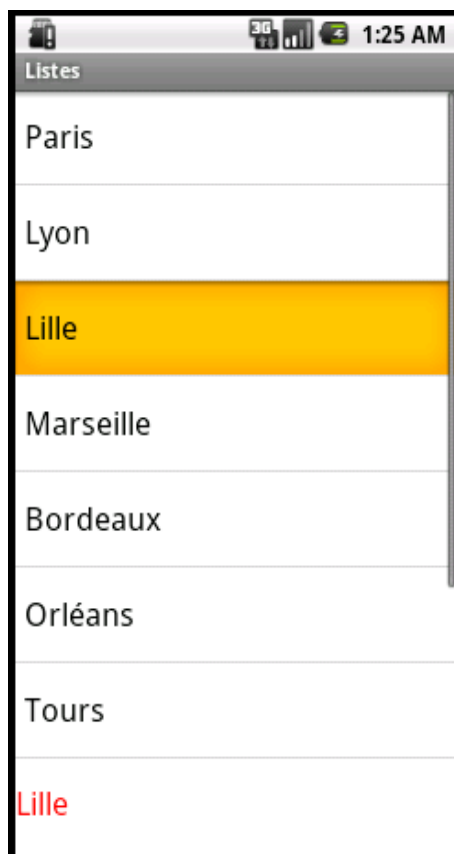
<a href="#">1.5.9 - Webographie.....</a>	<a href="#">82</a>
------------------------------------------	--------------------

## 1.1 - LES LISTES : LISTVIEW

### 1.1.1 - Présentation

#### 1.1.1.1 - IHM

Une ListView permet de présenter des informations (Items de menu, liste de contacts, ...) sous forme de liste.



Une ListView est un widget ajouté à un layout.

Il y a deux possibilités de traiter une ListView dans une activité :

- ✓ soit l'activité hérite de **ListActivity**, et la gestion est simplifiée,
- ✓ soit l'activité hérite d'une autre classe, **Activity** par exemple.

Dans le premier cas le widget, dans le layout, doit être identifié avec un identifiant Android : @android:id/list.

```
<ListView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@android:id/list" />
```

Dans le deuxième cas le widget, toujours dans le layout, est identifié à votre convenance, toujours de façon sémantiquement significative.

```
<ListView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/listViewVilles" />
```

Dans quel cas va-t-on devoir choisir la deuxième solution ?

Si vous voulez afficher deux ListView dans la même activité !!!,  
Si vous devez créer une activité qui doit hériter d'une classe comme AppCompatActivity , Activity, Fragment, FragmentActivity, ActionBarActivity, FragmentPagerAdapter, etc.

### 1.1.1.2 - Remplissage

Possibilités pour le remplissage de la ListView :

Dans un premier temps la ListView sera remplie avec des données provenant d'une **ressource XML**.

Dans un deuxième temps la ListView sera remplie avec des données provenant d'un **tableau statique** initialisé dans le code Java.

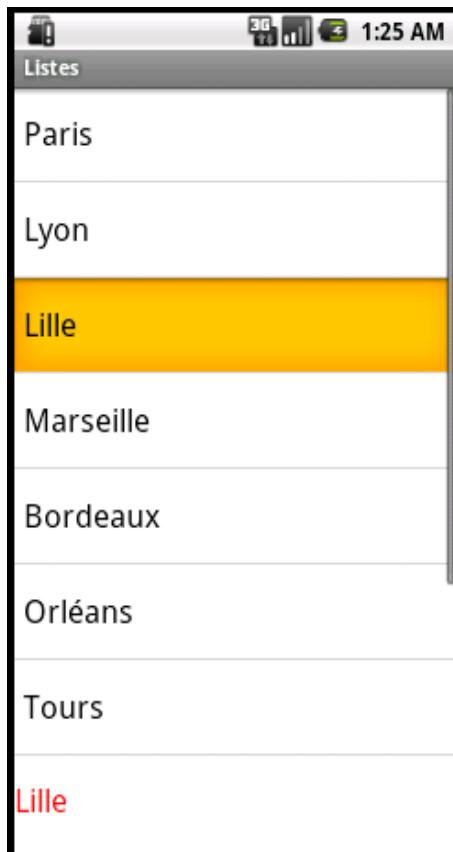
Par la suite, dans des chapitres suivants, la ListView sera remplie avec des données provenant d'une **BD locale** ou d'une **BD distante** ou d'une source de données XML ou JSON ...

### 1.1.2 - Une liste simple dans une classe héritant de Activity

Ou autre que Activity (AppCompatActivity, Fragment, ActionBarActivity, ...)

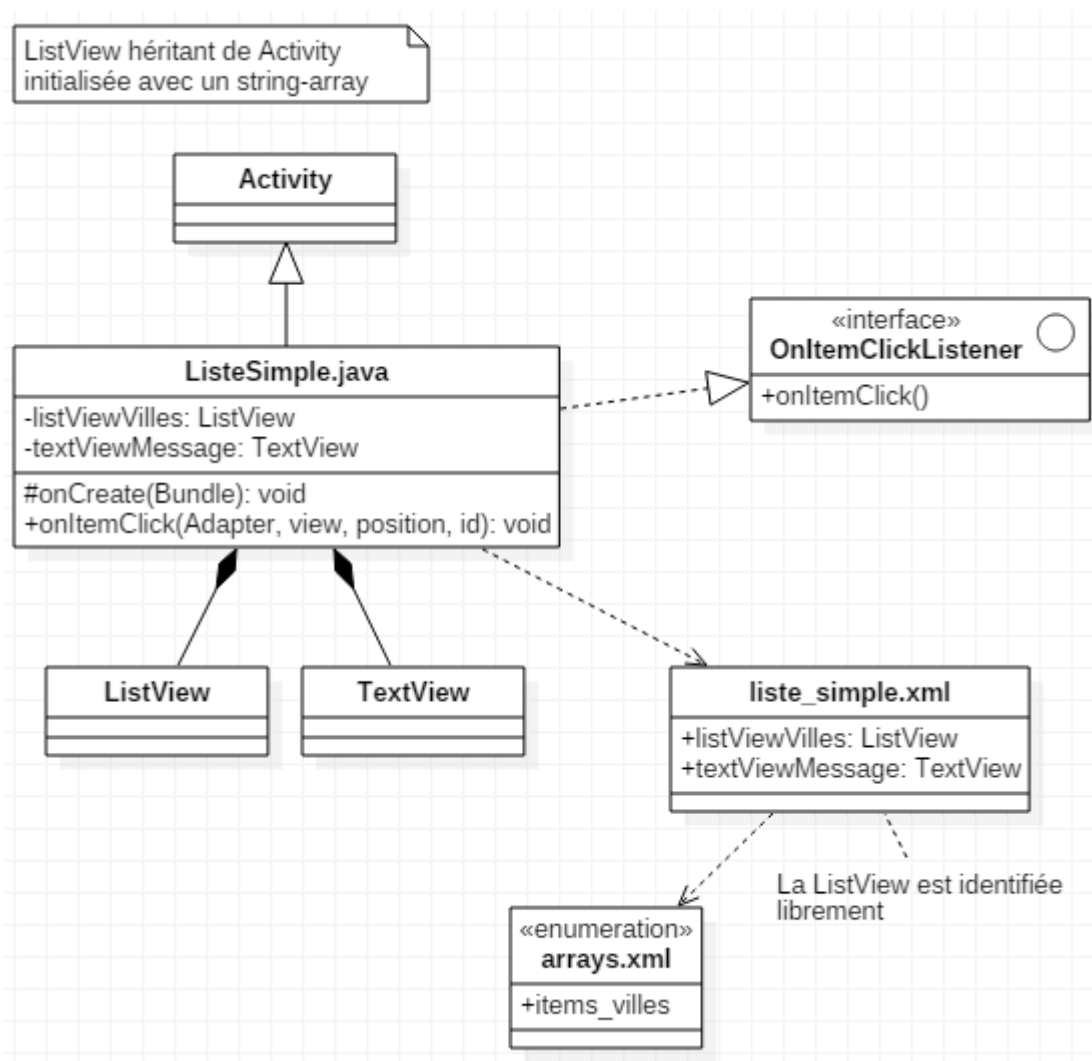
#### 1.1.2.1 - Objectif

Créer une liste et, lorsque l'on sélectionne un item, afficher la valeur dans un TextView (en rouge).





## 1.1.2.2 - Diagramme de classes



### 1.1.2.3 - Démarche et syntaxes

Création d'une activité avec l'assistant (New / Activity / Empty Activity ) – nommée ListeSimple.java – et son layout nommé liste\_simple.xml.

Création d'un tableau nommé item\_villes dans le fichier arrays.xml dans le dossier /res/values.

Implémentation de l'interface OnItemClickListener.

#### 1.1.2.4 - Le tableau de Strings

##### **/res/values/arrays.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="items_villes">
        <item>Paris</item>
        <item>Lyon</item>
        <item>Marseille</item>
        <item>Lille</item>
        <item>Bordeaux</item>
        <item>Orléans</item>
        <item>Tours</item>
    </string-array>
</resources>
```

### 1.1.2.5 - Le layout

Le layout (de type LinearLayout orientation verticale) est donc nommé **liste\_simple.xml**.

Ajoutez une ListView (Legacy/ListView) et identifiez-la comme vous voulez.

Ajoutez un TextView identifié par textViewMessage.

Remarque : si le contenu de la liste dépasse la hauteur de l'écran ou de la dimension allouée, automatiquement une barre de défilement - ScrollView - est créée.

Une ListView sera initialisée avec un tableau statique de Strings stocké dans /res/values/**arrays.xml** au moyen de l'attribut **android:entries="@array/nom\_du\_tableau"**.

#### liste\_simple.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">
    <ListView
        android:id="@+id/listViewVilles"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="90"
        android:entries="@array/items_villes">
    </ListView>
    <TextView
        android:id="@+id/textViewMessage"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="10"
        android:text="Message"
        android:textSize="15sp" />
</LinearLayout>
```

### 1.1.2.6 - L'activité

La nouvelle activité est nommée **ListeSimple**.

```
public class ListeSimple extends Activity {
```

Les imports nécessaires :

```
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.*;
```

Implémentation du Listener (AdapterView.OnItemClickListener)

```
public class ListeSimple extends Activity implements
AdapterView.OnItemClickListener {
```

Implémentation de la méthode onItemClick()

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position,
    long id) {
    // TODO
} /// onItemClick
```

### 1.1.2.7 - Récupération du libellé de l'item de la sélection

La méthode `onListItemClick` d'une Activity possède 4 arguments :

parent de type <code>ListView</code>	Correspond à la liste
vue de type <code>View</code>	Correspond à l'élément sélectionné dans la liste, ici un <code>TextView</code> Par défaut <code>TextView</code>
position de type <code>int</code>	Correspond à l'index de l'élément sélectionné dans la liste
id de type <code>long</code>	Correspond à l'identifiant de l'élément sélectionné dans la liste

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
    // Récupère le libellé de l'item sélectionné de la ListView
    String lsSelection = parent.getItemAtPosition(position).toString();
} /// onItemClick()
```

ou

```
String lsSelection = parent.getAdapter().getItem(position).toString();
```

ou plus court

```
String lsSelection = ((TextView)vue).getText().toString();
```

ou bof !

```
String lsSelection = tVilles[position];
```

## 1.1.2.8 - Code complet de l'activité

```
package fr.pb.withkotlin

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.AdapterView
import android.widget.Toast
import kotlinx.android.synthetic.main.liste_simple.*

class ListeSimple : AppCompatActivity() {

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

        listViewVilles.setOnItemClickListener = object :
        AdapterView.OnItemClickListener {

            override fun onItemClick(parent: AdapterView<*>, view: View,
                                    position: Int, id: Long) {

                // value of item that is clicked
                val itemValue = listViewVilles.getItemAtPosition(position)
as String

                // Toast the values
                Toast.makeText(applicationContext, "Position : $position\
nVille : $itemValue", Toast.LENGTH_LONG).show()
            } /// onItemClick

        } /// onItemClickListener

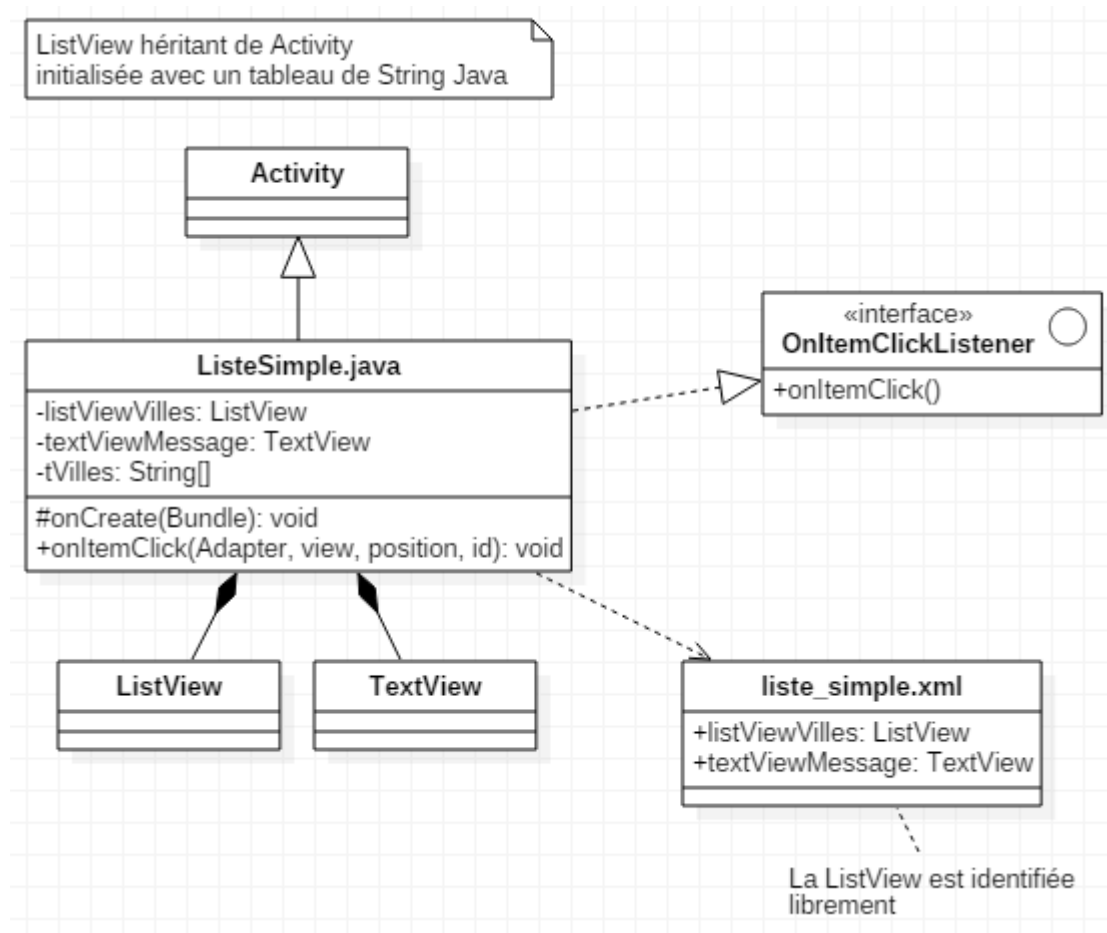
    } /// onCreate

} /// class
```

### 1.1.3 - Remplissage d'une ListView à partir d'un tableau Java

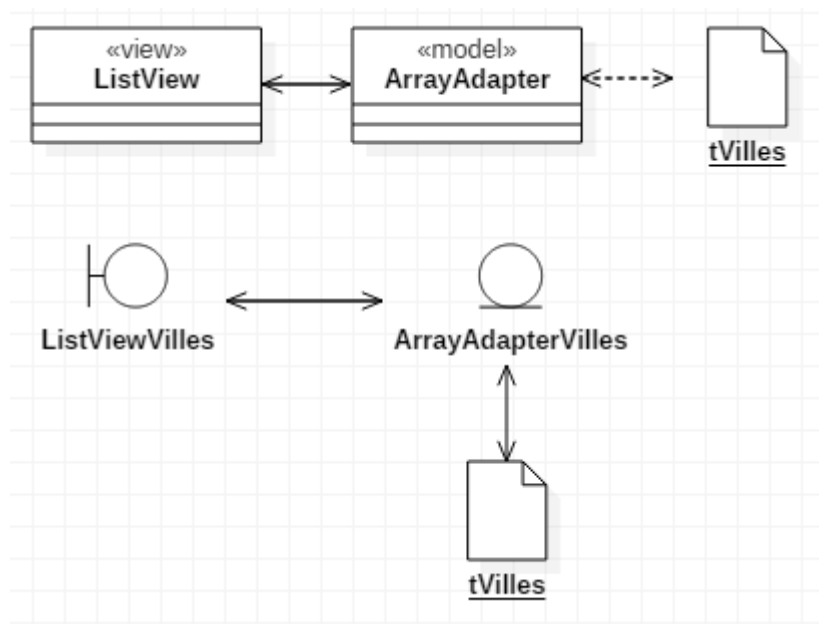
A l'inverse du précédent code la ListView (la View du MVC) est remplie à partir d'un tableau statique Java (La source de données) par l'intermédiaire d'un ArrayAdapter (le Model du MVC).

Tableau statique → ArrayAdapter → ListView.





Le point de vue MVC (Model View Controller).



### 1.1.3.1 - Syntaxes

**Remplissage d'une ListView** à partir un tableau de String (\*) via un ArrayAdapter.

Remplissage de l'ArrayAdapter :

```
ArrayAdapter(contexte, identifiant de ressource de la vue, tableau)
```

Le contexte c'est l'activité donc this.

L'identifiant de la ressource de la vue est ici la constante Android android.R.layout.simple\_list\_item\_1 puisque chaque item est un String (enfin une valeur de type élémentaire).

Le tableau est un tableau d'objets.

(\*) un ArrayAdapter peut être rempli avec un tableau de String ou d'Integer, etc (pas de types primitifs), une List.

Remplissage de la liste :

```
ListView.setAdapter(ArrayAdapter);
```

Exemple :

```
String[] tVilles = {"Paris", "Lyon", "Lille", "Marseille", "Bordeaux"};
ArrayAdapter<String>aaListe = new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, tVilles);
listViewVilles.setAdapter(aaListe);
```

Ici l'id de ressource est : android.R.layout.simple\_list\_item\_1.

**Vidage d'une ListView :**

```
listView.setAdapter(null);
```

**Suppression d'un item dans une Listview :**

Il faut supprimer l'item dans la source (plus facile si c'est un ArrayList).

Et ensuite « rafraîchir » la ListView en notifiant à l'ArrayAdapter que la source a changé.

```
aaListe.notifyDataSetChanged();
```

### 1.1.3.2 - Codes

#### Le fichier liste\_simple\_dynamique.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">

    <ListView
        android:id="@+id/listViewVilles"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="90">
    </ListView>

    <TextView
        android:id="@+id/textViewMessage"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="10"
        android:text="Message"
        android:textSize="15sp" />

</LinearLayout>
```

**ListeSimpleDynamique.java**

```
package fr.pb.withkotlin

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.AdapterView
import android.widget.AdapterView.OnItemClickListener
import android.widget.ArrayAdapter
import android.widget.ListView
import android.widget.Toast

class ListeSimpleDynamique : AppCompatActivity() {

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

        var tVilles = arrayOf("Paris", "Lyon", "Marseille", "Lille",
            "Rouen", "Bordeaux", "Orléans")
        val adapter =
            ArrayAdapter(this, android.R.layout.simple_list_item_1, tVilles)
        val listViewVilles: ListView = findViewById(R.id.listViewVilles)
        listViewVilles.setAdapter(adapter)

        listViewVilles.setOnItemClickListener = object :
            AdapterView.OnItemClickListener {

            override fun onItemClick(parent: AdapterView<*>, view: View,
                position: Int, id: Long) {
                // Value of item that is clicked
                val itemValue = listViewVilles.getItemAtPosition(position)
                as String
                // Toast the values
                Toast.makeText(applicationContext,
                    "Position : $position\nVille : $itemValue",
                    Toast.LENGTH_LONG)
                    .show()
            } /// onItemClick

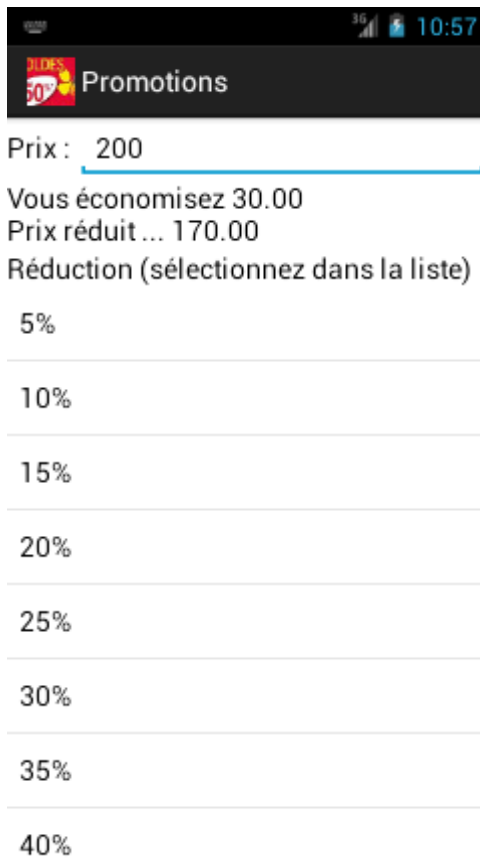
        } /// onItemClickListener

    } /// onCreate

} /// class
```

### 1.1.3.3 - Exercice : prix soldés

Saisie du prix, sélection du pourcentage de réduction dans la ListView, affichage de la réduction et du prix réduit.



The screenshot shows an Android application titled "Promotions". It features a text input field for "Prix" with the value "200". Below the input, it displays "Vous économisez 30.00" and "Prix réduit ... 170.00". A label "Réduction (sélectionnez dans la liste)" is followed by a vertical list of percentage options: 5%, 10%, 15%, 20%, 25%, 30%, 35%, and 40%.

Prix : 200

Vous économisez 30.00

Prix réduit ... 170.00

Réduction (sélectionnez dans la liste)

- 5%
- 10%
- 15%
- 20%
- 25%
- 30%
- 35%
- 40%

### 1.1.4 - Une liste avec une icône identique pour chaque item

#### 1.1.4.1 - Objectif

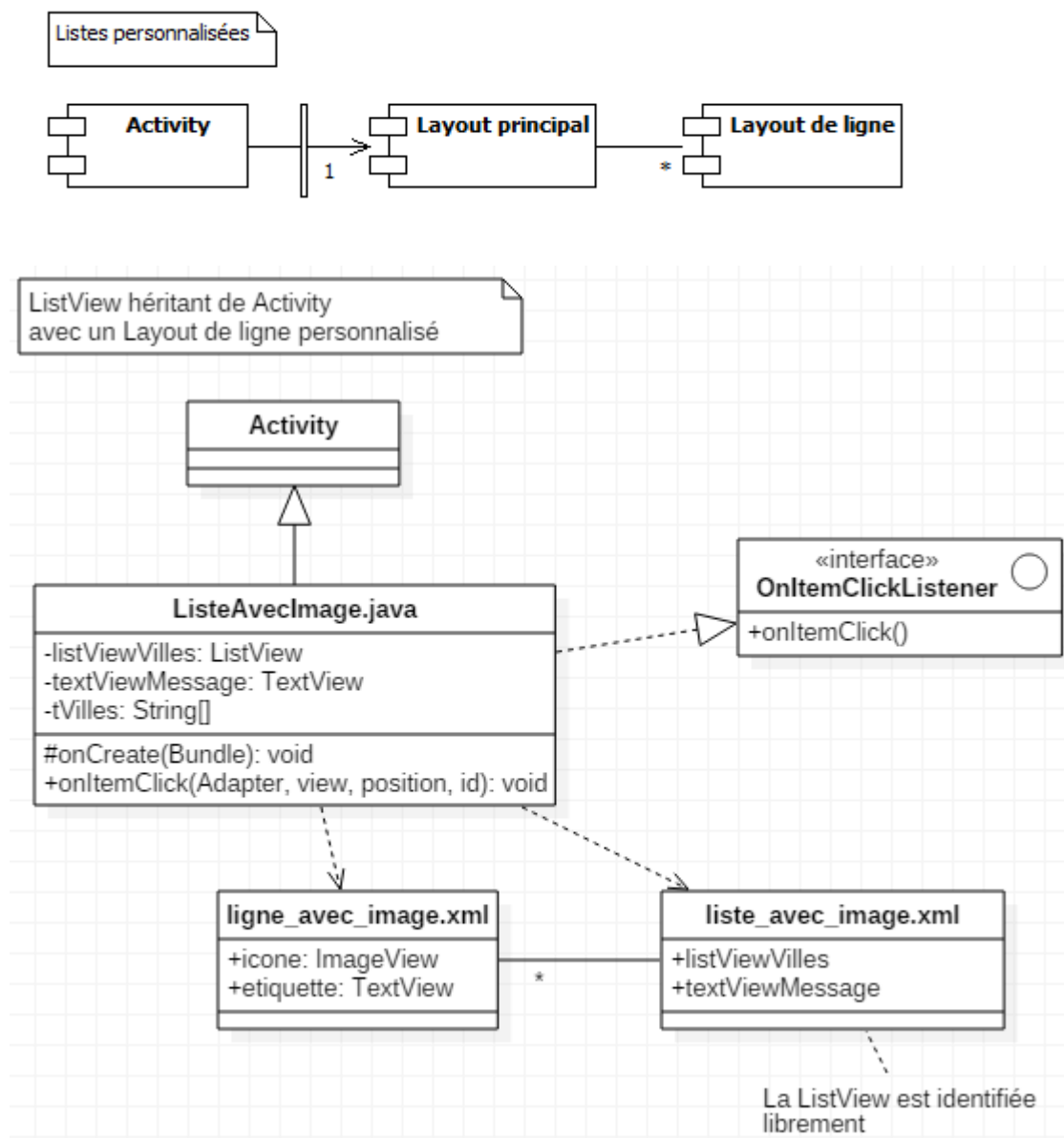


[Orléans](#)

## 1.1.4.2 - Principe et diagramme de composants

**Pour avoir un élément composite, donc autre chose qu'une String standard, pour chaque item de la liste il faut créer, à côté du layout principal qui contient la ListView, un layout "externe" correspond à la ligne.**

Pour cet exemple ce layout sera de type LinearLayout et horizontal.  
Il contiendra l'icône et le texte donc une ImageView et un TextView.  
L'icône est le même pour chaque item de la liste.  
Le texte est différent pour chaque item de la liste.



#### 1.1.4.3 - Démarche

Le layout principal est le même que le précédent.

Ajoutez une icône dans /res/drawable/. **ok.png** par exemple ou utilisez une image système **@android:drawable/btn\_star\_big\_off** par exemple.

Créez un nouveau layout pour la ligne. Nommez-le **ligne\_avec\_image.xml**.

Seule une ligne change dans le code de l'activité. C'est celle de la création de l'ArrayAdapter.

#### 1.1.4.4 - Le layout de ligne

```
<?xml version="1.0" encoding="utf-8"?>
<!-- ligne_avec_image.xml -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <ImageView
        android:id="@+id/icone"
        android:padding="2dip"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@android:drawable/btn_star_big_off"
        android:contentDescription="Image OK"
    />

    <TextView
        android:id="@+id/etiquette"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30sp"
    />

</LinearLayout>
```



#### 1.1.4.5 - Syntaxes

Créez une nouvelle activité ou modifiez l'activité précédente; **il n'y a qu'une seule ligne à modifier**. Le `setContentView()` ne change pas.

On fait référence au layout correspondant à la ligne spécifique lors de la création de l'ArrayAdapter.

Ici le constructeur de la classe ArrayAdapter est :

```
new ArrayAdapter(contexte, layout de vue perso, identifiant du texte,
tableau)
```

il y a 4 paramètres alors que précédemment il y en avait 3.

```
ArrayAdapter<String> aaListe = new ArrayAdapter<String>(this,
R.layout.ligne_avec_image, R.id.etiquette, tVilles);
```

Note : `R.id.etiquette` correspond au widget du layout de ligne qui est alimenté avec des textes différents qui se trouvent dans le tableau correspondant au paramètre n° 4.

Par comparaison l'ancien code était :

```
ArrayAdapter<String> aaListe = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, tVilles);
```

Récupération de la sélection (comme précédemment)

```
String lsSelection = parent.getAdapter().getItem(position).toString();
textViewSelection.setText(lsSelection);
```

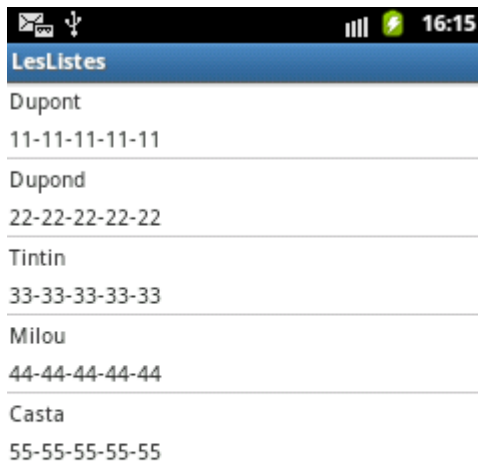
#### 1.1.4.6 - Exercice : mise en place de la ListView avec une icône

Mettez en place les codes dans une nouvelle activité nommée **ListeAvecImage.java**.

## 1.1.5 - Une liste avec une structure de composants

### 1.1.5.1 - Objectif

Afficher une liste où chaque item contient plusieurs éléments dynamiques.  
Récupérer les informations sélectionnées dans un TextView.



Casta : 55-55-55-55-55

A partir du moment où l'on veut afficher dans chaque item d'une ListView un élément composite et que plusieurs valeurs sont obtenues dynamiquement il faut utiliser un **SimpleAdapter** et non plus un ArrayAdapter.

Et comme précédemment créer un layout de ligne.

#### 1.1.5.2 - Principe et démarche

On utilise un **SimpleAdapter** à la place d'un **ArrayAdapter**.

Le principe est quasiment le même que précédemment.

Il faut un layout principal pour la liste. Nous réutiliserons **liste\_simple.xml**.

Il faut un layout pour la ligne. Il sera nommé **ligne\_complexe.xml**.

Ce qui diffère fondamentalement c'est que les données sont multiples et diverses pour chaque item, alors que dans le cas précédent seul le texte changeait et l'image était identique pour chaque item.

Il n'est donc plus possible d'utiliser un **ArrayAdapter**. Il faut utiliser un **SimpleAdapter**.

Ce qui diffère accessoirement c'est que le 2<sup>ème</sup> layout est un layout vertical contenant 2 TextView. Il pourrait y avoir plus de widgets et même d'autres layouts contenant d'autres widgets, le principe serait le même.

Les TextView de la ListView seront remplis à partir de 2 tableaux de String initialisés avec des constantes.

### 1.1.5.3 - Syntaxes et exemples

#### Syntaxe

```
SimpleAdapter sa = new SimpleAdapter (
    contexte,
    les données sous forme de List de Map,
    layout pour la ligne,
    String[] tableau des noms des clés des éléments des Map,
    int[] tableau des références des éléments du layout pour la ligne
);
```

Cette fois-ci il y a 5 paramètres :

Le contexte,  
les données,  
le layout spécifique pour la ligne de la liste,  
un tableau de string avec les noms des clés du Map de la List des données,  
un tableau d'int avec les ID des widgets du layout de ligne.

#### Exemple

L'initialisation de la liste :

```
SimpleAdapter sa = new SimpleAdapter (
    this.getBaseContext(),
    listeItems,
    R.layout.ligne_complexe,
    new String[] { "nom", "telephone" },
    new int[] { R.id.nom, R.id.telephone }
);
```

La récupération de la sélection :

```
Map<String,String> map = (Map<String, String>)
parent.getAdapter().getItem(position);
```

#### 1.1.5.4 - Codes

##### liste\_simple.xml

cf plus haut, rien de nouveau.

##### ligne\_complexe.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- ligne_complexe.xml -->

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/nom"
        android:padding="3dip"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
    />

    <TextView
        android:id="@+id/telephone"
        android:padding="3dip"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
    />

</LinearLayout>
```

**ListeComplexe.java**

```
import android.os.Bundle;
import android.view.View;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.AdapterView;
import android.widget.TextView;
import android.app.Activity;
import java.util.HashMap;
import java.util.ArrayList;
import java.util.Map;
import java.util.List;

// -----
public class ListeComplexe extends Activity implements
AdapterView.OnItemClickListener{
    /*
     * Attributs
     */
    private TextView textViewSelection;
    private ListView listView1;

    /*
     * Methodes
     */
    @Override
    // -----
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.liste_simple);

        listView1 = findViewById(R.id.listView1);
        listView1.setOnItemClickListener(this);

        String[] tNoms = {"Dupont", "Dupond", "Tintin", "Milou", "Casta"};
        String[] tTelephones = {"11-11-11-11-11", "22-22-22-22-22", "33-33-33-33-33", "44-44-44-44-44", "55-55-55-55-55"};

        textViewSelection = (TextView)
        findViewById(R.id.textViewSelection);

        // --- Creation de l'ArrayList pour remplir la ListView
        List<Map<String, String>> listeItems = new ArrayList();

        // --- HashMap pour les informations pour un item
        Map<String, String> hm;

        // --- Remplissage dynamique
        for(int i = 0; i < tNoms.length; i++) {
            hm = new HashMap();
            hm.put("nom", tNoms[i]);
            hm.put("telephone", tTelephones[i]);
            listeItems.add(hm);
        }

        // --- Utilisation d'un adaptateur pour affecter les valeurs aux
        éléments de la liste

        /*
```

An easy adapter to map static data to views defined in an XML file. You can specify the data backing the list as an ArrayList of Maps. Each entry in the ArrayList corresponds to one row in the list. The Maps contain the data for each row. You also specify an XML file that defines the views used to display the row, and a mapping from keys in the Map to specific views. Binding data to views occurs in two phases. First, if a SimpleAdapter.ViewBinder is available, `setViewValue(android.view.View, Object, String)` is invoked. If the returned value is false, the following views are then tried in order:

A view that implements Checkable (e.g. CheckBox). The expected bind value is a boolean.

TextView. The expected bind value is a string and `setViewText(TextView, String)` is invoked.

ImageView. The expected bind value is a resource id or a string and `setViewImage(ImageView, int)` or `setViewImage(ImageView, String)` is invoked.

```

        */

        // SimpleAdapter(Context context, List<? extends Map<String, ?>>
data, int resource, String[] from, int[] to)

        SimpleAdapter sa = new SimpleAdapter (
            this,
            listeItems,
            R.layout.ligne_complexe,
            new String[] {"nom", "telephone"},
            new int[] {R.id.nom, R.id.telephone}
        );

        // Affectation directe du Simple Adapter a la ListView
        listView1.setAdapter(sa);

    } /// onCreate

    // -----
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {

        Map<String,String> map = (Map<String, String>)
parent.getAdapter().getItem(position);

        textViewSelection.setText(map.get("nom") + " : " +
map.get("telephone"));

    } /// onItemClick()

} /// class ListeComplexe

```

Affaire classée !

#### 1.1.5.5 - Exercice : ajoutez le prénom et l'e-mail.

## 1.1.6 - Une liste avec une structure mixte : image et textes

### 1.1.6.1 - Objectif

Afficher une image et 3 textes par item de ListView.

Pas de grandes différences avec le précédent sauf pour les références vers les images stockées dans /res/drawable/. Chaque ID des images est stocké sous forme de String dans un tableau.

Lorsque l'utilisateur sélectionne un item le nom du pays et le drapeau sont affichés en bas de l'écran.



Cf projet Listes pour une démonstration statique. Les données proviennent de tableaux de String et les images des ressources.

Cf projet SQL Distant pour une démonstration dynamique. Les données proviennent de la BD et les images du serveur distant.



## 1.1.6.2 - Codes

**liste\_pays\_complexe.xml : le layout principal**

```
<?xml version="1.0" encoding="utf-8"?>
<!-- liste_pays_complexe.xml -->

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@+id/listViewPays"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="70" >
    </ListView>

    <TextView
        android:id="@+id/textViewPays"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="10"
        android:text="@string/pays"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="#0000FF" />

    <ImageView
        android:id="@+id/imageViewPays"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="20"
        android:contentDescription="drapeau"
        android:src="@drawable/france" />

</LinearLayout>
```

**pays\_ligne.xml : le layout de ligne**

```
<?xml version="1.0" encoding="utf-8"?>
<!-- pays_ligne.xml -->

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <ImageView
        android:id="@+id/imageViewDrapeau"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="image" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/nom"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="3dip" />

        <TextView
            android:id="@+id/code_alpha"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="3dip" />

        <TextView
            android:id="@+id/code_tel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="3dip" />
    </LinearLayout>
</LinearLayout>
```

**ListePaysComplexe.java**

```
package fr.pb.listes;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import android.app.Activity;
import android.os.Bundle;

import android.widget.SimpleAdapter;
import android.widget.AdapterView;
import android.widget.Toast;

import android.widget.TextView;
import android.widget.ImageView;
import android.view.View;
import android.widget.ListView;

/*
 * ListePaysComplexe : liste des pays avec drapeau, nom, code ISO,
 indicatif telephonique
 */
public class ListePaysComplexe extends Activity implements
AdapterView.OnItemClickListener{

    private ListView listViewPays;
    private TextView textViewPays;
    private ImageView imageViewPays;

    @Override
    // -----
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.liste_pays_complexe);

        listViewPays = findViewById(R.id.textViewPays);
        textViewPays = findViewById(R.id.textViewPays);
        imageViewPays = findViewById(R.id.imageViewPays);

        String[] tNomsPays = {"Fidji", "Finlande", "France"};
        String[] tCodeAlpha = {"FJ", "FI", "FR"};
        String[] tIndicatifsTelephoniques = {"679", "358", "33"};

        // Récupère l'id de l'image fidji ... en base decimale
        String[] tImages = new String[3];
        tImages[0] = String.valueOf(R.drawable.fidji);
        tImages[1] = String.valueOf(R.drawable.finlande);
        tImages[2] = String.valueOf(R.drawable.france);

        try {
            List<Map<String, String>> listePays = new
ArrayList<Map<String, String>>();
            Map<String, String> hm;

            for(int i = 0; i < tNomsPays.length; i++) {
                hm = new HashMap<String, String>();
```

```

        hm.put("image", tImages[i]);
        hm.put("nom", tNomsPays[i]);
        hm.put("code_alpha", "Code ISO : " +
tCodeAlpha[i]);
        hm.put("code_tel", "Indicatif téléphonique : " +
tIndicatifsTelephoniques[i]);

        listePays.add(hm);
    }

    SimpleAdapter sa = new SimpleAdapter(
        this.getContext(),
        listePays,
        R.layout.pays_ligne,
        new String[] { "image", "nom", "code_alpha",
"code_tel" },
        new int[] { R.id.imageViewDrapeau, R.id.nom,
R.id.code_alpha, R.id.code_tel }
    );

    listViewPays.setAdapter(sa);

    listViewPays.setOnItemClickListener(this);

    } catch (Exception e) {
        Toast.makeText(getContext(), "Erreur ? " +
e.getMessage(), Toast.LENGTH_LONG).show();
    }

    } // / onCreate
} // / classe

```

Pour afficher le nom du pays et l'image :

```

@Override
// -----
public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {

    Map<String, String> hm = (Map<String, String>)
parent.getItemAtPosition(position);

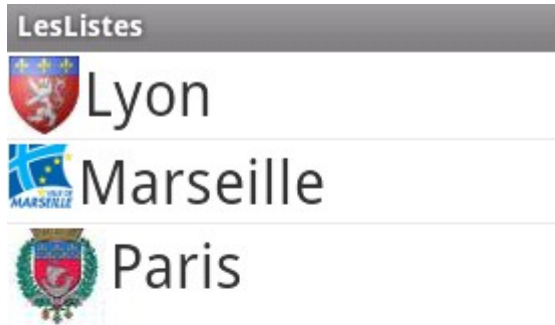
    textViewPays.setText(hm.get("nom"));
    int liIdImage = Integer.valueOf(hm.get("image"));
    imageViewPays.setImageResource(liIdImage);

} // / onItemClick

```

### 1.1.6.3 - Exercice : une liste avec une icône différente pour chaque item

#### Objectif



Les icônes sont dans les /res/drawable !!!  
L'affaire est un peu, ..., beaucoup plus complexe !

### 1.1.7 - Mettre à jour une ListView

1.1.7.1 - Si toutes ou grande partie des items sont à modifier

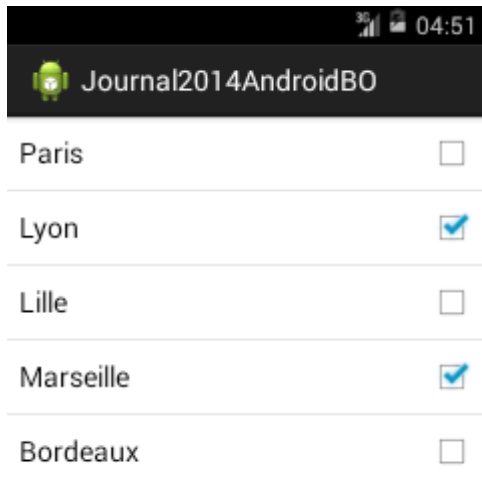
Tout simplement en ré-instanciant l'adaptateur et en le réaffectant à la ListView.

1.1.7.2 - Si un seul item est à ajouter

1.1.7.3 - Supprimer un item

## 1.1.8 - Les listes et la sélection multiple

### 1.1.8.1 - Objectif



Lyon-Marseille-

### 1.1.8.2 - Syntaxes

```
ArrayAdapter<String> aa = new ArrayAdapter<String>(this,  
android.R.layout.simple_list_item_multiple_choice, String[]);
```

```
ListView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
```

#### Éventuellement dans le layout

```
android:entries="@array/villes" android:choiceMode="multipleChoice">
```

### Gérer les items sélectionnés

#### Récupère l'ensemble des items sélectionnés

```
SparseBooleanArray sba = listView.getCheckedItemPositions();
```

#### Récupère le nombre d'items sélectionnés

```
sba.size()
```

#### Récupère le texte de l'item sélectionné

```
listView.getItemAtPosition(sba.keyAt(i)).toString()
```

```
SparseBooleanArray sba = listView.getCheckedItemPositions();  
for (int i = 0; i < sba.size(); i++) {  
    String lsTexte = listView.getItemAtPosition(sba.keyAt(i)).toString();  
    // suite du code  
}
```



### 1.1.8.3 - Layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@+id/listViewVilles"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="80" >
    </ListView>

    <TextView
        android:id="@+id/textViewSelection"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="20"
        android:textAppearance="?android:attr/textAppearanceMedium" />

</LinearLayout>
```

## 1.1.8.4 - Activité

```
package fr.pb.listes;

import android.os.Bundle;
import android.app.Activity;
import android.util.SparseBooleanArray;
import android.view.View;
import android.widget.*;

/*
http://openclassrooms.com/courses/creez-des-applications-pour-android/des-
widgets-plus-avancees-et-des-boites-de-dialogue
*/

/**
 *
 * @author Pascal
 *
 */
public class ListeMultiple extends Activity implements
    AdapterView.OnItemClickListener{

    /*
     * Attributs
     */
    private TextView textViewSelection;
    private String[] tVilles = { "Paris", "Lyon", "Lille", "Marseille",
        "Bordeaux" };

    private ListView laListeView;
    private StringBuilder isbSelections;
    private ListView listViewVilles;

    /**
     *
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        isbSelections = new StringBuilder();

        setContentView(R.layout.liste_multiple);
        listViewVilles = findViewById(R.id.listViewVilles);

        textViewSelection = findViewById(R.id.textViewSelection);

        /*
         * Choix multiple et donc case à cocher ...
         */
        ArrayAdapter<String> aaListe = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_multiple_choice,
            tVilles);
        listViewVilles.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);

    }
}
```

```

        * Choix unique et donc bouton radio ... Mais possibilité de
        TextView ou de case a cocher ...
        */
        // ArrayAdapter<String> aaListe = new
        ArrayAdapter<String>(this,
        // android.R.layout.simple_list_item_single_choice, tVilles);
        // listViewVilles.setChoiceMode(ListView.CHOICE_MODE_SINGLE);

        // --- Methode de la classe ListActivity pour remplir la liste
        listViewVilles.setAdapter(aaListe);
    }

    @Override
    // -----
    public void onItemClick(AdapterView<?> parent, View view, int
    position, long id) {

        // Récupère le libelle de l'item de la ListView
        String lsSelection =
        parent.getItemAtPosition(position).toString();
        isbSelections.append(lsSelection);
        isbSelections.append("-");
        // textViewSelection
        // .setText(parent.getItemAtPosition(position).toString());

        // Dans le cas de choix unique
        // int liPosition = laListeView.getCheckedItemPosition();

        // Dans le cas de choix multiples
        SparseBooleanArray sba = laListeView.getCheckedItemPositions();
        // Ou
        if (laListeView.getCheckedItemPositions().get(0)) {
            Toast.makeText(this, "Position 1 est sélectionnée",
            Toast.LENGTH_SHORT).show();
        }
        if (laListeView.getCheckedItemPositions().get(1)) {
            Toast.makeText(this, "Position 2 est sélectionnée",
            Toast.LENGTH_SHORT).show();
        }
        if (laListeView.getCheckedItemPositions().get(2)) {
            Toast.makeText(this, "Position 3 est sélectionnée",
            Toast.LENGTH_SHORT).show();
        }

        textViewSelection.setText(isbSelections.toString());

    } // / onItemClick()

    // ----- BOF !!! KO pour l'instant
    public void onItemClickLongClick(AdapterView<?> adapter, View view,
        int position, long id) {
        Toast.makeText(this, "Appui long !!!",
        Toast.LENGTH_SHORT).show();
    }

} // / class

```

### 1.1.8.5 - La même chose mais plus dans le layout

#### liste\_multiple\_000.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="80"
        android:entries="@array/villes"
        android:choiceMode="multipleChoice">

        <!-- android:choiceMode="singleChoice" -->
        <!-- android:choiceMode="multipleChoice" -->
    </ListView>

    <TextView
        android:id="@+id/textViewSelection"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="20"
        android:textAppearance="?android:attr/textAppearanceMedium" />

</LinearLayout>
```

**ListeMultiple000.java**

```
package fr.pb.listes;

import android.os.Bundle;
import android.app.Activity;
import android.util.SparseBooleanArray;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

/*
http://openclassrooms.com/courses/creez-des-applications-pour-android/des-  
widgets-plus-avancees-et-des-boites-de-dialogue
*/

/**
 *
 * @author pascal
 */
public class ListeMultiple000 extends Activity implements
    AdapterView.OnItemClickListener{

    /**
     * Attributs
     */
    private TextView textViewSelection;
    private ListView listView;
    private StringBuilder isbSelections;

    /**
     *
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        isbSelections = new StringBuilder();

        setContentView(R.layout.liste_multiple_000);
        listView = findViewById(R.id.listView);
        listView.setOnItemClickListener(this);
        // Aucun effet !!!
        listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);

        textViewSelection = (TextView)
            findViewById(R.id.textViewSelection);
    }

    @Override
    // -----
    public void onItemClick(AdapterView<?> parent, View view, int
        position, long id) {

        // Récupère le libelle de l'item de la ListView
    }
}
```

```
        String lsSelection =
parent.getItemAtPosition(position).toString();
        isbSelections.append(lsSelection);
        isbSelections.append("-");

        // Dans le cas de choix unique
        // int liPosition = laListView.getCheckedItemPosition();

        // Dans le cas de choix multiples
        SparseBooleanArray sba = laListView.getCheckedItemPositions();
        // Ou
        if (laListView.getCheckedItemPositions().get(0)) {
            Toast.makeText(this, "Position 1 est sélectionnée",
Toast.LENGTH_SHORT).show();
        }
        if (laListView.getCheckedItemPositions().get(1)) {
            Toast.makeText(this, "Position 2 est sélectionnée",
Toast.LENGTH_SHORT).show();
        }
        if (laListView.getCheckedItemPositions().get(2)) {
            Toast.makeText(this, "Position 3 est sélectionnée",
Toast.LENGTH_SHORT).show();
        }

        textViewSelection.setText(isbSelections.toString());

    } // / onItemClick()

    // ----- BOF !!! KO pour l'instant
    public void onItemClick(AdapterView<?> adapter, View view,
        int position, long id) {
        Toast.makeText(this, "Appui long !!!",
Toast.LENGTH_SHORT).show();
    }

} // / class
```

#### 1.1.8.6 - Sélectionner tous les items d'une ListView

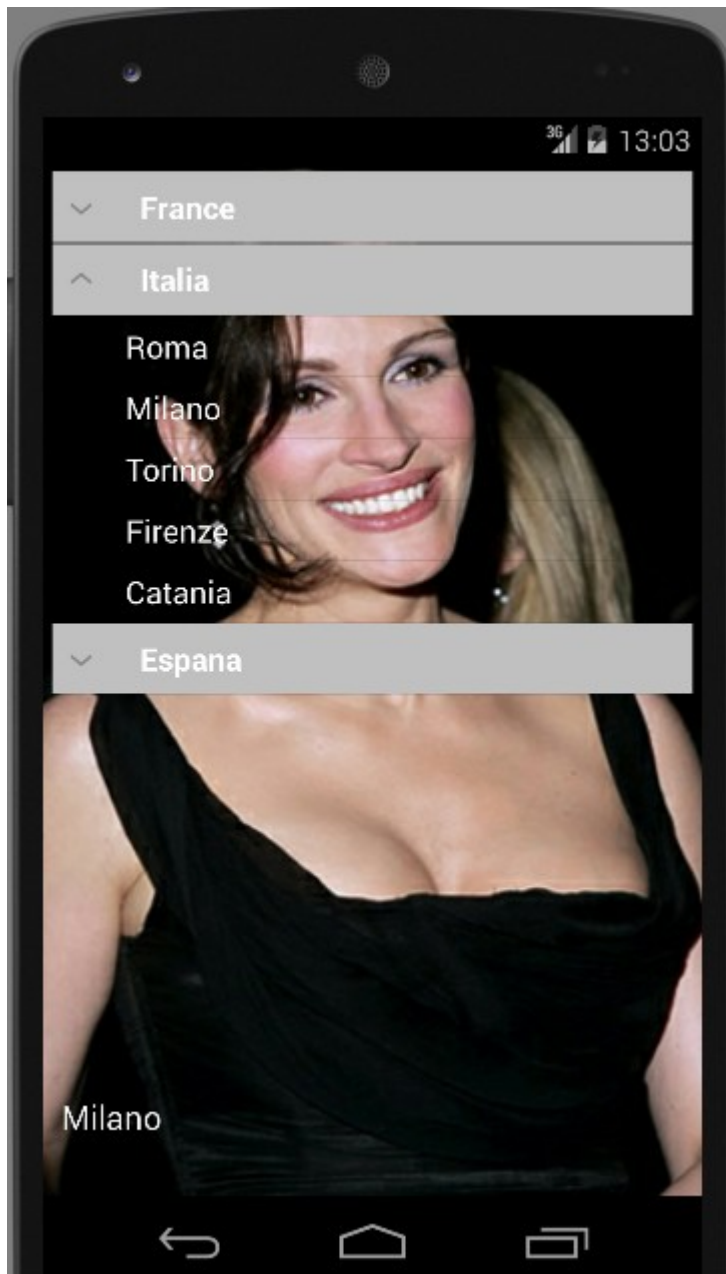
```
for (int i = 0; i < listViewXXX.getChildCount(); i++) {  
    listViewXXX.setItemChecked(i, true);  
}
```

false pour tout dé-sélectionner.

## 1.2 - EXPANDABLELISTVIEW (ACCORDÉON)

### 1.2.1 - Objectif

Créer une ListView de type accordéon contenant plusieurs soufflets.





## 1.2.2 - Démarche

Il faut créer 3 layouts :

- ✓ le layout principal avec le widget « ExpandableListView (accordeon.xml),
- ✓ le layout pour chaque en-tête de soufflet (en\_tete\_soufflet.xml),
- ✓ le layout pour chaque item des soufflets (item\_soufflet.xml).

Tous les layouts sont de type « LinearLayout Vertical ».  
Les deux derniers layouts contiennent un TextView.

Il faut créer 2 classes Java :

- ✓ une activité (Accordeon.java),
- ✓ une autre pour l'adapter qui hérite de la classe BaseExpandableListAdapter (ExpandableListAdapter.java).

<b>ExpandableListAdapter</b>
(-) Context contexte
(-) List<String> listeEnTetes
(-) Map<String, List<String>> mapItems
(+) ExpandableListAdapter()
(+) getChild()
(+) getChildId()
(+) getChildView()
(+) getChildrenCount()
(+) getGroup()
(+) getGroupCount()
(+) getGroupId()
(+) getGroupCount()
(+) hasStableIds()
(+) isChildSelectable()

ExpandableListAdapter() : le constructeur. Les arguments dépendent du choix de votre mise en place de l'Adapter (List de Map, Map de String et de List, List de List, ...).

La méthode getChild() : renvoie un item.

La méthode getChildId() : renvoie l'ID d'un item (sa position dans le soufflet).

La méthode getChildView() : renvoie un item.

La méthode getChildrenCount() : renvoie le nombre d'item par soufflet.

La méthode getGroup() : renvoie un soufflet.

La méthode getGroupCount() : renvoie le nombre de soufflets.

La méthode getGroupId() : renvoie l'ID d'un soufflet (sa position).

La méthode getGroupCount() : renvoie le nombre de soufflets de l'accordéon.

La méthode hasStableIds() : renvoie un booléen indiquant que l'ID est stable.

La méthode isChildSelectable() : renvoie true ou false en fonction du fait que vous vouliez rendre sélectionnable un item.

## 1.2.3 - Codes

### 1.2.3.1 - Les codes des layouts

#### accordeon.xml

C'est le layout principal.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/julia_roberts_robe_noire"
    android:orientation="vertical"
    android:padding="5dp">

    <ExpandableListView
        android:id="@+id/expandableListView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="90" />

    <TextView
        android:id="@+id/textViewSelection"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="10"
        android:padding="5dp"
        android:text="Sélection"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="@color/blanc" />

</LinearLayout>
```

**en\_tete\_soufflet.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/dimgray"
    android:orientation="vertical"
    android:padding="8dp">

    <TextView
        android:id="@+id/textViewEnteteSoufflet"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

        android:paddingLeft="?android:attr/expandableListPreferredItemPaddingLeft"
        android:textColor="@color/blanc"
        android:textSize="17dp" />

</LinearLayout>
```

**item\_soufflet.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="55dip"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textViewItemSoufflet"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="5dp"

        android:paddingLeft="?android:attr/expandableListPreferredChildPaddingLeft"
        android:paddingTop="5dp"
        android:textColor="@color/blanc"
        android:textSize="17dip" />

</LinearLayout>
```

### 1.2.3.2 - Le code de l'adapter et le code de l'activité

#### Le code de l'adapteur : `ExpandableListAdapter.java`

```
package fr.pb.listesavancees;

import java.util.HashMap;
import java.util.List;

import android.content.Context;
import android.graphics.Typeface;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseExpandableListAdapter;
import android.widget.TextView;

/**
 *
 */
public class ExpandableListAdapter extends BaseExpandableListAdapter {

    private Context contexte;
    private List<String> listeEnTetes;
    private HashMap<String, List<String>> mapItems;

    public ExpandableListAdapter(Context contexte, List<String>
listeEnTetes, HashMap<String, List<String>> mapItems) {
        this.contexte = contexte;
        this.listeEnTetes = listeEnTetes;
        this.mapItems = mapItems;
    }

    @Override
    public Object getChild(int groupPosition, int childPositon) {
        return this.mapItems.get(this.listeEnTetes.get(groupPosition))
            .get(childPositon);
    }

    @Override
    public long getChildId(int groupPosition, int childPosition) {
        return childPosition;
    }

    @Override
    public View getChildView(int groupPosition, final int childPosition,
boolean isLastChild, View convertView, ViewGroup parent) {
        String lsTexteItem = (String) getChild(groupPosition,
childPosition);

        if (convertView == null) {
            LayoutInflater inflater = (LayoutInflater) this.contexte
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            convertView = inflater.inflate(R.layout.item_soufflet, null);
        }

        TextView textViewItem = (TextView) convertView
            .findViewById(R.id.textViewItemSoufflet);
    }
```

```
        textViewItem.setText(lsTexteItem);
        return convertView;
    }

    @Override
    public int getChildrenCount(int groupPosition) {
        return this.mapItems.get(this.listeEnTetes.get(groupPosition))
            .size();
    }

    @Override
    public Object getGroup(int groupPosition) {
        return this.listeEnTetes.get(groupPosition);
    }

    @Override
    public int getGroupCount() {
        return this.listeEnTetes.size();
    }

    @Override
    public long getGroupId(int groupPosition) {
        return groupPosition;
    }

    @Override
    public View getGroupView(int groupPosition, boolean isExpanded,
                               View convertView, ViewGroup parent) {
        String titreSoufflet = (String) getGroup(groupPosition);
        if (convertView == null) {
            LayoutInflater inflater = (LayoutInflater) this.contexte
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            convertView = inflater.inflate(R.layout.en_tete_soufflet,
null);
        }

        TextView textViewEntete = (TextView) convertView
            .findViewById(R.id.textViewEnteteSoufflet);
        textViewEntete.setTypeface(null, Typeface.BOLD);
        textViewEntete.setText(titreSoufflet);

        return convertView;
    }

    @Override
    public boolean hasStableIds() {
        return false;
    }

    @Override
    public boolean isChildSelectable(int groupPosition, int childPosition)
    {
        return true;
    }
} // // class
```

**Le code de l'activité : Accordeon.java**

```
package fr.pb.listesavancees;

import java.util.ArrayList;
import java.util.Map;
import java.util.HashMap;
import java.util.List;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ExpandableListView;
import android.widget.ExpandableListView.OnChildClickListener;
//import android.widget.ExpandableListView.OnGroupClickListener;
//import android.widget.ExpandableListView.OnGroupCollapseListener;
//import android.widget.ExpandableListView.OnGroupExpandListener;

/**
 *
 */
public class Accordeon extends Activity implements OnChildClickListener {

    private ExpandableListAdapter listAdapter;
    private ExpandableListView expandableListView;
    private List<String> listeEnTetes;
    private Map<String, List<String>> mapItems;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.accordeon);

        expandableListView = (ExpandableListView)
        findViewById(R.id.expandableListView);

        remplirListe();

        listAdapter = new ExpandableListAdapter(this, listeEnTetes,
        mapItems);
        expandableListView.setAdapter(listAdapter);
    } // onCreate

    /**
     * Remplissage de la liste
     */
    private void remplirListe() {
        listeEnTetes = new ArrayList();
        mapItems = new HashMap();

        // Les items
        List<String> listeVillesFrance = new ArrayList();
        listeVillesFrance.add("Paris");
        listeVillesFrance.add("Lyon");
        listeVillesFrance.add("Marseille");
        listeVillesFrance.add("Bordeaux");
        listeVillesFrance.add("Grenoble");

        List<String> listeVillesItalie = new ArrayList();
        listeVillesItalie.add("Roma");
        listeVillesItalie.add("Milano");
```

```
listeVillesItalie.add("Torino");
listeVillesItalie.add("Firenze");
listeVillesItalie.add("Catania");

List<String> listeVillesEspagne = new ArrayList();
listeVillesEspagne.add("Madrid");
listeVillesEspagne.add("Barcelona");
listeVillesEspagne.add("Valencia");
listeVillesEspagne.add("Granada");
listeVillesEspagne.add("Alicante");

// Les en-tetes
listeEnTetes.add("France");
listeEnTetes.add("Italia");
listeEnTetes.add("Espana");

mapItems.put(listeEnTetes.get(0), listeVillesFrance);
mapItems.put(listeEnTetes.get(1), listeVillesItalie);
mapItems.put(listeEnTetes.get(2), listeVillesEspagne);

} /// remplirListe

@Override
public boolean onChildClick(ExpandableListView parent, View vue, int
groupPosition, int childPosition, long id) {
    /*
    parent : ExpandableListView
    vue : LinearLayout
    groupPosition : position du groupe (commence a 0)
    childPosition : position dans le groupe (commence a 0)
    id : position dans le groupe (commence a 0)
    */
    LinearLayout itemSelectionne = (LinearLayout) vue;
    TextView textView = (TextView) itemSelectionne.getChildAt(0);
    String lsSelection = textView.getText().toString();
    this.textViewSelection.setText(lsSelection);
    return true;
} /// onChildClick

} /// class
```

### 1.3 - REMPLIR UNE LISTVIEW AVEC DES IMAGES DE LA SD

Le layout comprend une ListView nommée listViewPhotos.

```
package fr.pb.listes;

import android.net.Uri;
import android.os.Environment;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.widget.*;
import java.io.*;
import java.util.*;

/**
 *
 */
public class ListViewSD extends ActionBarActivity {

    private ListView listViewPhotos;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.essai);

        listViewPhotos = (ListView) findViewById(R.id.listViewPhotos);

        remplirListView();
    } /// onCreate

    /**
     *
     */
    private void remplirListView() {
        File dir;
        String lsChemin = "";
        String[] tFichiers;
        List<String> listeJPG = new ArrayList<>();
        List<Map<String, Object>> listeItems;
        Map<String, Object> hm;
        String lsEnr;
        File fichier;
        Uri uriFichier;
        SimpleAdapter sa;

        try {
            // Renvoie Pictures de la SD
            dir = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_P
ICTURES).getAbsolutePath() + "/salma");
            lsChemin = dir.getAbsolutePath() + "/";

            // Récupère la liste des fichiers du dossier selectionne
tFichiers = dir.list();
        }
    }
}
```



```
// Ne Récupère que les fichiers d'extension .jpg
for (int i = 0; i < tFichiers.length; i++) {
    if (tFichiers[i].endsWith(".jpg")) {
        listeJPG.add(tFichiers[i]);
    }
}

// --- Creation de l'ArrayList pour remplir la ListView
listeItems = new ArrayList<Map<String, Object>>();

// --- Remplissage dynamique
for (int i = 0; i < listeJPG.size(); i++) {
    lsEnr = listeJPG.get(i);

    hm = new HashMap<String, Object>();

    fichier = new File(lsChemin + "/" + lsEnr);
    uriFichier = Uri.fromFile(fichier);

    hm.put("nomPhoto", (Object) lsEnr);
    hm.put("imagePhoto", uriFichier);

    listeItems.add(hm);
} /// for

sa = new SimpleAdapter(
    this,
    listeItems,
    R.layout.essai_ligne,
    new String[]{"imagePhoto", "nomPhoto"},
    new int[]{R.id.imageViewPhoto, R.id.textViewPhoto}
);

listViewPhotos.setAdapter(sa);

} catch (Exception e) {
    Toast.makeText(this, e.getMessage(),
Toast.LENGTH_LONG).show();
} finally {

    } // / finally

} /// remplirListView

} /// class
```

## 1.4 - REMPLIR UNE LISTVIEW AVEC DES IMAGES DU WEB

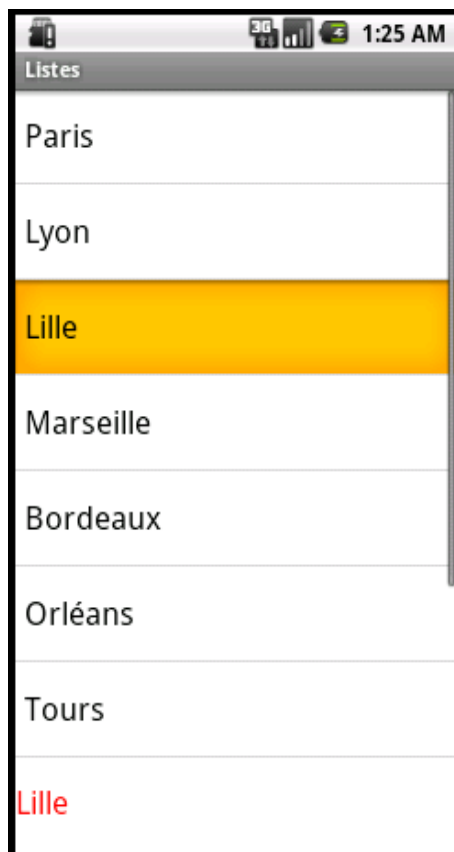


## 1.5 - ANNEXES

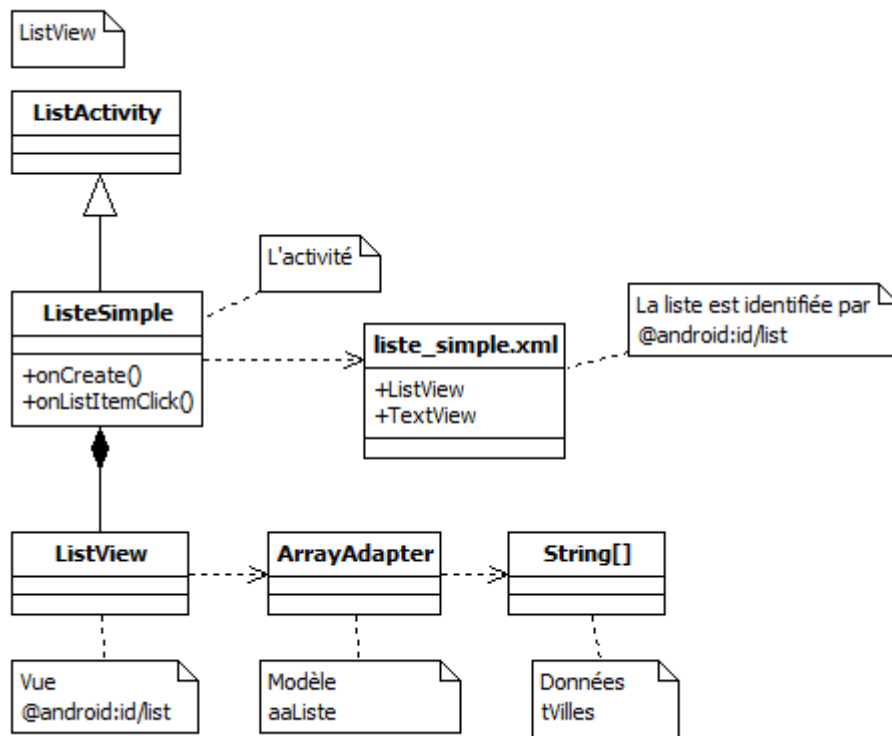
### 1.5.1 - Une liste simple dans une classe héritant de ListActivity

#### 1.5.1.1 - Objectif

Créer une liste et, lorsque l'on sélectionne un item, afficher la valeur dans un TextView (en rouge).



## 1.5.1.2 - Diagramme de classes



Note : ajouter le « schéma » pour une ListView alimentée par un array-strings.

### 1.5.1.3 - Démarche et syntaxes

#### Le layout

Créez un layout (de type LinearLayout orientation verticale) nommé **liste\_simple.xml**.

Ajoutez une ListView (Composite/ListView).

Ajoutez un TextView identifié par textViewSelection.

```
<ListView
    android:id="@android:id/list"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="90" >
</ListView>
```

**L'id de la liste doit être @android:id/list si l'activité hérite de la classe ListActivity. Et dans ce cas il ne pourra y avoir qu'une seule liste dans ce layout.**

Remarque : si le contenu de la liste dépasse la hauteur de l'écran ou de la dimension allouée, automatiquement une barre de défilement - ScrollView - est créée.

Comme pour un Spinner, une ListView peut être initialisée avec un tableau statique de Strings stocké dans /res/values/**arrays.xml** au moyen de l'attribut **android:entries="@array/nom\_du\_tableau"**.

Par exemple : /res/values/arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="items_villes">
        <item>Paris</item>
        <item>Lyon</item>
        <item>Marseille</item>
        <item>Lille</item>
        <item>Bordeaux</item>
        <item>Orléans</item>
        <item>Tours</item>
    </string-array>
</resources>
```

## L'activité

Dans le même projet créez une nouvelle activité nommée **ListeSimple**.

Puisque dans le layout la liste est unique et est identifiée par **@android:id/list** l'activité **doit** hériter de la classe **ListActivity**.

```
public class ListeSimple extends ListActivity {
```

Les imports nécessaires :

```
import android.app.ListActivity;
import android.view.View;
import android.widget.ListView;
```

Remplissez la liste avec un tableau de String (\*) via un ArrayAdapter.

```
ArrayAdapter<String>aaListe = new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, tVilles);
```

Ici l'id de ressource est : android.R.layout.simple\_list\_item\_1.

La syntaxe est la même que pour le Spinner :

```
ArrayAdapter(contexte, identifiant de ressource de la vue, tableau)
```

Remplissage de la liste

```
this.setListAdapter(ArrayAdapter);
```

ou si l'on récupère au préalable la liste

```
ListView liste = this.getListView();
liste.setAdapter(ArrayAdapter);
```

(\*) un ArrayAdapter peut être rempli avec un tableau de String ou d'Integer, etc (pas de types primitifs), une List.

## Récupération du libellé de l'item de la sélection

Avec la méthode `onListItemClick` de `ListActivity` possède 4 arguments :

parent de type <code>ListView</code>	Correspond à la liste
vue de type <code>View</code>	Correspond à l'élément sélectionné dans la liste, ici un <code>TextView</code> Par défaut <code>TextView1</code>
position de type <code>int</code>	Correspond à l'index de l'élément sélectionné dans la liste
id de type <code>long</code>	Correspond à l'identifiant de l'élément sélectionné dans la liste

```
@Override
public void onListItemClick(ListView parent, View vue, int position, long
id) {
    // Récupère le libelle de l'item selectionne de la ListView
    String lsSelection = parent.getItemAtPosition(position).toString();
} /// onListItemClick()
```

ou

```
String lsSelection = parent.getAdapter().getItem(position).toString();
```

ou plus court

```
String lsSelection = ((TextView)vue).getText().toString();
```

ou bof !

```
String lsSelection = tVilles[position];
```

## 1.5.1.4 - Codes

## Le fichier liste\_simple.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- liste_simple.xml -->

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="90"
        >
    </ListView>

    <TextView
        android:id="@+id/textViewSelection"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="10"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="@color/bleu" />

</LinearLayout>
```



**ListeSimple.java**

```
package fr.pb.listes;

import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.view.View;

// -----
public class ListeSimple extends ListActivity {
    private TextView textViewSelection;

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

        textViewSelection = (TextView)
        findViewById(R.id.textViewSelection);

        String[] tVilles =
        {"Paris", "Lyon", "Lille", "Marseille", "Bordeaux", "Orléans", "Tours", "Arras"};

        ArrayAdapter<String> aaListe = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, tVilles);

        // --- Methode de la classe ListActivity pour remplir la liste
        this.setListAdapter(aaListe);
    } /// onCreate()

    @Override
    public void onListItemClick(ListView parent, View vue, int position,
    long id) {
        // Récupère le libelle de l'item de la ListView

        textViewSelection.setText(parent.getItemAtPosition(position).toString());
    } /// onListItemClick()
} /// classe
```

## 1.5.2 - Comparatif des syntaxes

**Adapter adapté à la structure d'un item de la ListView :**

Contenu de la ListView	Constructeur de l'Adapter
String	ArrayAdapter(this, android.R.layout.simple_list_item_1, tableau de valeurs)
Image unique + String	ArrayAdapter(this, R.layout.ligne_avec_image, R.id.etiquette, tableau de valeurs)
Composite	SimpleAdapter (this, List de HashMap, R.layout.ligne_complexe, tableau des clés du HashMap, tableau des identifiants des widgets du layout de ligne)
Image différente + String	idem

	Liste simple	Liste avec une image	Liste composite
Adapter	ArrayAdapter	ArrayAdapter	SimpleAdapter
Layout	android.R.layout.simple_list_item_1	R.layout.layout_perso	R.layout.layout_perso
Id ...		R.id.etiquette	
Données	String[]	String[]	List de HashMap

**ListActivity et Activity :**

	ListActivity	Activity
Id de la liste dans le layout	@android:id/list	@+id/listView1
Implémentation d'une interface	Aucune	OnItemClickListener
Liaison widget-événement	Aucun	Via setOnItemClickListener()
Événement	onListItemClick	onItemClick
Arguments de l'événement	ListView parent, View v, int position, long id (*)	AdapterView<?> parent, View v, int position, long id (*)
Remplissage de la ListView	this.setListAdapter(Adapter)	ListView.setAdapter(Adapter)

(\*) Le parent est la ListView, la View est l'item de la ListView, la position dans la ListView.

Rappel pour le remplissage d'un Spinner : ArrayAdapter(contexte, android.R.layout.simple\_spinner\_item, tableau de valeurs) et pour l'événement onItemClick.

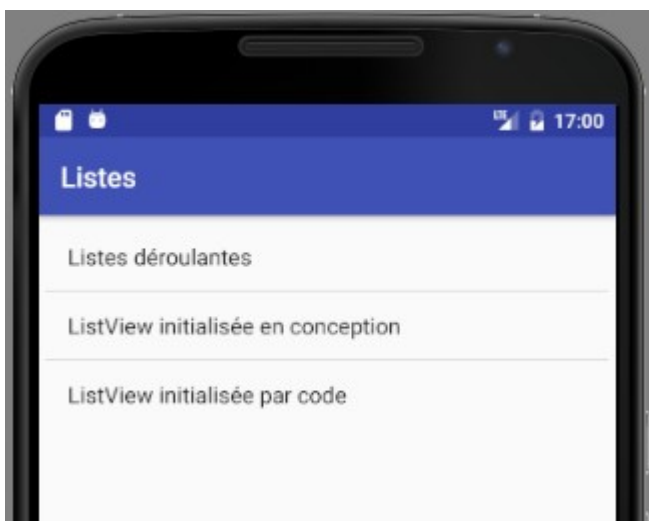
### 1.5.3 - Menu sous forme de ListView ... pour la suite

#### 1.5.3.1 - L'objectif

Un menu principal sous forme de ListView ... Lorsque l'on clique sur un item de la liste une **intention** – les intentions seront examinées en détails dans le document multiples\_activites\_android\_java.odt - est créée et une activité s'affiche. Le retour au menu principal est réalisé au moyen du bouton



ou parfois d'un bouton dans une activité avec l'appel à la méthode finish().



### 1.5.3.2 - Le layout

#### main.xml

pour une Activity ...

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">
    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listViewMain" />
</LinearLayout>
```

pour une Activity avec un remplissage via un arrays.xml ...

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">
    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listViewMain"
        android:entries="@array/items_main" />
</LinearLayout>
```

pour une ListActivity ...

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp">
    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@android:id/list"
    />
</LinearLayout>
```

### 1.5.3.3 - L'activité (version générique)

```
package fr.pb.listes;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Set;

public class Main extends AppCompatActivity implements
    AdapterView.OnItemClickListener {

    private ListView listViewMain;
    private Map<String, String> mapMenu;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mapMenu = new LinkedHashMap();
        mapMenu.put("Listes déroulantes", "ListesDeroulantes");
        mapMenu.put("ListView initialisée en conception", "ListView1");
        mapMenu.put("ListView initialisée par code", "ListView2");

        Set<String> cles = mapMenu.keySet();
        String[] tCles = cles.toArray(new String[cles.size()]);

        listViewMain = findViewById(R.id.listViewMain);
        listViewMain.setOnItemClickListener(this);
        ArrayAdapter<String> adaptateur = new ArrayAdapter(this,
            android.R.layout.simple_list_item_1, tCles);
        listViewMain.setAdapter(adaptateur);

    } // onCreate

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
        position, long id) {
        Intent intention = new Intent();

        String lsSelection =
            parent.getItemAtPosition(position).toString();
        String lsNomClasse = mapMenu.get(lsSelection);

        try {
            Class<?> classe = Class.forName("fr.pb.listes." +
                lsNomClasse);
            intention.setClass(this, classe);
            startActivity(intention);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

```
        }  
    } /// onItemClick  
} /// class
```

#### 1.5.3.4 - L'activité (version spécifique)

Cf 1.3 (ListView dans une classe de type Activity).

## 1.5.4 - Personnalisation d'une ListView : colorisation

### 1.5.4.1 - Coloriser les lignes

Cf ListeColorisee du projet AndroidListes.

Chaque item à un fond noir et un texte rouge.



Il existe plusieurs techniques : une utilisant un layout texte et une autre utilisant les styles. Voyons la première.

### Démarche

Il faut pour obtenir ce résultat créer un layout pour la ligne puis l'appliquer dans l'activité comme cela a été fait pour la liste avec une image et un texte.



**ligne\_texte\_colorisee.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/textViewColorisee"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:textColor="@color/rouge"
    android:background="@color/noir"
/>
```

et dans l'activité

```
ArrayAdapter<String> aaListe = new ArrayAdapter<String>(this,
R.layout.ligne_texte_colorisee, tVilles);
```

#### 1.5.4.2 - Coloriser la sélection

L'item sélectionné a un fond bleu et un texte noir.

Là encore il existe plusieurs techniques, dont une par code Java ; c'est celle que nous allons voir.



#### Démarche

On utilise la méthode `setBackgroundColor(couleur)` appliquée à la ligne courante récupérée avec `parent.getChildAt(position)`.

On conserve dans une variable d'instance la position courante lors du clic dans la liste.

## Code

cf la classe ListeAvecSelectionColorisee du projet AndroidListes.

La variable `iiAnciennePosition` est déclarée comme attribut de classe.

La variable `iiAnciennePosition` est initialisée dans le constructeur à -1.

```
// -----
public void onListItemClick(ListView parent, View v, int position, long
id) {

    // Affiche le texte de la selection

    textViewSelection.setText(parent.getAdapter().getItem(position).toString()
);

    // Modifie la couleur du fond de la ligne selectionnee
    //parent.getChildAt(position).setBackgroundColor(Color.BLUE);
    //parent.getChildAt(position).setBackgroundColor(Color.rgb(169, 234,
254)); // AZURIN
    parent.getChildAt(position).setBackgroundColor(Color.LTGRAY);
    // Modifie la couleur du fond de l'ancienne ligne selectionnee
    if (iiAnciennePosition != -1 && iiAnciennePosition != position) {

parent.getChildAt(iiAnciennePosition).setBackgroundColor(Color.WHITE);
    }

    // Conserve la position
    iiAnciennePosition = position;

} /// onListItemClick()
```

Note : cela ne modifie que sur la longueur du texte !

#### 1.5.4.3 - Changer la taille de la police (ou une autre propriété) des items de la liste

Avec un layout de ligne personnalisé.

Par exemple :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textViewLigne"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Small Text"
        android:textAppearance="?android:attr/textAppearanceSmall" />

</LinearLayout>
```

Pour le code de l'activité cf 1.2.2 ; c'est le même type de code.

Cf un exemple d'utilisation dans NON\_SQL\_android\_java.odt dans le chapitre sur les préférences.

### 1.5.5 - Appui long sur un item

#### Démarche et Syntaxe

Déclarer un objet de type `OnItemLongClickListener`.  
Instancier l'objet.  
Coder la méthode `onItemLongClick`.  
Associer la liste au listener via la méthode `setOnItemClickListener`.

#### Imports

```
import android.widget.AdapterView;  
import android.widget.AdapterView.OnItemClickListener;
```

#### Code

dans le `onCreate` de l'activité (qui hérite de `ListActivity`).

```
/*  
 * Le clic long  
 */  
OnItemLongClickListener oilcl;  
  
oilcl = new OnItemLongClickListener() {  
    @Override  
    public boolean onItemLongClick(AdapterView<?> parent, View v,  
        int position, long id) {  
        String lsSelection =  
parent.getItemAtPosition(position).toString();  
        Toast.makeText(getBaseContext(), "long clic",  
Toast.LENGTH_SHORT).show();  
        return true;  
    } // / onItemLongClick  
};  
  
ListView liste = this.getListView();  
  
liste.setOnItemLongClickListener(oilcl);
```

.../...

Ou (et c'est peut-être mieux !) :

```
import android.widget.AdapterView;  
import android.widget.AdapterView.OnItemClickListener;
```

```
public class UneClasse extends ActionBarActivity implements  
OnItemLongClickListener {
```

```
listViewXXX.setOnItemClickListener(this);
```

```
@Override  
public boolean onItemLongClick(AdapterView<?> parent, View view, int  
position, long id) {  
    //Toast.makeText(this, "onItemLongClick", Toast.LENGTH_SHORT).show();  
    // Code ...  
    return false;  
} /// onItemLongClick
```

## 1.5.6 - Simuler un click dans une ListView

**Objectif** : sélectionner un item d'une ListView.

### 1.5.6.1 - Quand l'activité est de type ListActivity

```
ListView lv = getListView();  
onListItemClick(lv, null, 2, 2);
```

Note : l'index commence à 0.

```
public void onListItemClick(ListView parent, View vue, int position, long  
id) {  
    /// Le code  
} // / onListItemClick
```

### 1.5.6.2 - Quand l'activité est de type Activity

```
onItemClick(listView1, null, 2, 2);
```

```
public void onItemClick(AdapterView<?> av, View vue, int position, long  
id) {  
    /// Le code  
} /// onItemClick
```

## 1.5.7 - Récupérer le contenu d'une ListView

### 1.5.7.1 - Une ListView simple

Une boucle sur l'adapter.

```
StringBuilder lsbContenu = new StringBuilder();  
  
ListAdapter la = listView1.getAdapter();  
int liCount = la.getCount();  
for (int i = 0; i < liCount; i++) {  
    lsbContenu.append(la.getItem(i).toString());  
    lsbContenu.append("\n");  
}
```



## 1.5.8 - Pré-sélectionner un élément d'une ListView

### 1.5.8.1 - Une ListView simple

Une boucle sur l'adapter.

```
String lsElementASelectionner = "Ajout";
ListAdapter la = listView1.getAdapter();
int liCount = la.getCount();
for (int i = 0; i < liCount; i++) {
    if (la.getItem(i).toString().equals(lsElementASelectionner)) {
        listView1.setSelection(i);
    }
}
```

### 1.5.9 - Webographie

<http://developer.android.com/guide/topics/ui/layout/listview.html>

<http://www.vogella.com/tutorials/AndroidListView/article.html>