Université de Carthage
FSEG Nabeul
3eme LF IAG

TP4_ Android

Implémentation des Widgets de sélection (liste) sous Android

1. Traitement des Spinners:

La contrainte d'affichage et taille de l'écran ont imposé à Android de fournir une solution pour les listes déroulantes. La première solution concerne les Spinners. Le Spinner d'Android est l'équivalent des boites déroulantes que l'on trouve dans certains kits de développement (JComboBox en Java par exemple). On cliquant sur le Spinner une boite de sélection apparait permettant à l'utilisateur de faire son choix. Android permet aisément de reconnaitre les choix proposés dans les Spinners grâce à un ensemble d'adaptateurs permettant de fournir une interface commune à toutes les listes de choix, que ce soient des listes déroulantes, des tableaux statiques ou encore des bases de données. Les adaptateurs d'Android se chargent de fournir la liste des données d'un widget de sélection et de convertir les différents éléments en vues spécifiques pour qu'elles s'affichent dans les widgets de sélection. L'adaptateur le plus simple est ArrayAdapter puisqu'il suffit d'envelopper dans un tableau une boites de sélection pour le rendre disponible pour un traitement.

Créer un nouveau projet nommée Spinner_Demo et changer son layout à un layout linéaire. Dans la palette du layout, choisir dans le répertoire From widgets, les widgets TextView puis Spinner et insérer les dans le layout. Le fichier XML du layout doit être comme suit :

```
<?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/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
    <Spinner
        android:id="@+id/spinner1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Dans le code Java de l'activité du projet, il faut enlever tous les imports et les codes Java concernant les menus et les fragments pour ne pas encombrer le projet. Dans la partie importe de l'activité, il faut qu'il ait uniquement les imports suivants :

```
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Spinner;          pour le traitement du spinner
import android.widget.AdapterView;
import android.widget.TextView;         pour le traitement de l'adaptateur du
import android.widget.ArrayAdapter;        spinner (ArrayAdapter)
```

Comme le Spinner est une liste de sélection donc nous devons placer des écouteurs sur chaque élément de la liste, or la liste est gérée par l'adaptateur par la suite les écouteurs vont être placés sur l'adaptateur. C'est pourquoi l'activité doit implémenter l'interface AdapterView et plus précisément la fonction onItemSelectedListener. La classe l'activité doit alors avoir le code suivant:

```
public class Demo_spinner extends ActionBarActivity implements
AdapterView.OnItemSelectedListener {
    // declaration des variables de la classe
   TextView Selection;
    String[] item={"Tunisie","France","Italie","Canada","Amerique"};   la liste
                                          qu'on va afficher au niveau de notre spinner

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        // création de l'activité et affectation du Layout
        super.onCreate(savedInstanceState);
        setContentView(R.layout.demo_spinner);

        // récupération des identificateurs des widgets TextView et Spinner pour leur
         //traitement et affectation des écouteurs
        Selection = (TextView) findViewById(R.id.textView1);
        Spinner spin = (Spinner) findViewById(R.id.spinner1);
        Spin.setOnItemSelectedListener(this);

        // création de l'adaptateur ArrayAdapter dont les éléments de son tableau
        //sont de type chaine de caractères à prendre à partir du variable    item
       //pour remplier les items du spinner se trouvant dans le layout.

//Contexte d'affichage
        ArrayAdapter<String> aa= new ArrayAdapter<String>
(this,android.R.layout.simple_spinner_item,item);

      //Remplissage des différents éléments du spinner de la vue par les éléments du
     //la variable item
       aa.setDropDownViewResource(android.R.layout.simple_spinner_item);

    // affectation de l'adaptateur au spinner pour gérer les écouteurs

       spin.setAdapter(aa);
```

```
    }
    // le traitement à faire si un choix est fait par l'utilisateur


    public void onItemSelected (AdapterView<?> parent, View v, int position, long id)
    {
       Selection.setText(item[position]);
    }

// le traitement à faire si aucun choix n'est fait

    public void onNothingSelected(AdapterView<?> parent)
    {
       Selection.setText("");
    }


    }
```

La fonction onItemSelected définit le traitement à faire lorsqu'un choix proposé par le spinner est sélectionné. Les paramètres de cette fonction sont les suivants :
Parent : définit l'AdapterView ou la sélection va se faire.
V : la vue dans laquelle on va faire la sélection.
Position : la position de l'article sélectionné dans l'adapteur
Id : l'identificateur de la ligne sélectionnée dans le spinner.

2. Traitement des AutoCompleteTextView

AutoCompleteTextView est une sorte d'hybride d'EditText et de Spinner. Avec l'auto-complétion, le texte saisi par l'utilisateur est traité comme un préfixe de filtrage : il est comparé à une liste de choix qui ressemble à un spinner. L'utilisateur peut alors continuer sa saisie (si le mot n'est pas dans la liste) ou choisir une entrée de celle-ci pour qu'elle devienne la valeur du champ.

Créer un nouveau projet nommée Autocomplete_Demo et changer son layout à un layout linéaire. Dans la palette du layout, choisir dans le répertoire From widgets, le TextView et inserer le dans le layout. Ensuite, choisir dans le repertoire Texts Fields le widget AutoCompleteTextView et ajouter le dans le layout. Le fichier XML du layout doit être comme suit :

```
<?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/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView" />
```

```
    <AutoCompleteTextView
        android:id="@+id/autoCompleteTextView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:completionThreshold="1"
       >
                                    Une ligne qu'il faut ajouter manuellement
        <requestFocus />
    </AutoCompleteTextView>

</LinearLayout>
```

Dans la balise de l'AutoCompleteTextView, ajouter la ligne :

android :completetionThreshold="1"

Cette propriété de l'AutoCompleteTextView permet d'indiquer le nombre minimal de caractères à entrer avant que la liste de proposition n'apparaisse.
Le traitement de l'AutoCompleteTextView se fait par l'enregistrement d'un TextWatcher qui prévient la vue lorsqu'un texte a été saisi ou modifié. Ce type d'évènement est déclenché par la saisie manuelle ou par une sélection dans la liste des propositions.

Dans le code Java de l'activité du projet, il faut enlever tous les imports et les codes Java concernant les menus et les fragments pour ne pas encombrer le projet. Dans la partie importe de l'activité, il faut qu'il ait uniquement les imports suivants :

```
import android.support.v7.app.ActionBarActivity;
import android.text.Editable;
import android.text.TextWatcher;          traitement de la saisie
import android.os.Bundle;
import android.widget.AutoCompleteTextView;   traitement du filtrage des entrées
import android.widget.TextView;
import android.widget.ArrayAdapter;
```

La classe l'activité doit alors avoir le code suivant:

```
public class MainActivity extends ActionBarActivity implements TextWatcher {

   //déclaration des variables de la classe
   TextView Selection;
   AutoCompleteTextView edit;

   String[] items={"Tunisie","Turqui","Tailand","France","Felande","Algerie","Arabie
                Saoudite", "Eriteria","Etat Unis","Maroc","Mauritanie"};

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

```
        Selection = (TextView) findViewById(R.id.textView1);
        edit = (AutoCompleteTextView) findViewById(R.id.autoCompleteTextView1);

        //affectation des écouteurs de Texte qui s'activent lors d'une saisie ou
        //sélection d'un texte

        edit.addTextChangedListener(this);

       // création et affectation d'un adaptateur du type ArrayAdapter dont les
       //éléments de son tableau sont des chaines de caractères et qui seront
       //affectées aux différentes lignes de l'AutoCompleteTextView a partir de la
       //variable item

           edit.setAdapter(new ArrayAdapter<String>(this,
                          android.R.layout.simple_dropdown_item_1line,items));
    }

       // Le traitement à faire si un texte est saisi ou sélectionné à partir de la
       // liste proposée

    @Override
       public void onTextChanged(CharSequence arg0, int arg1, int arg2, int arg3) {

             Selection.setText(edit.getText());
       }


    public void beforeTextChanged(CharSequence s, int start, int count, int after)

    {
       //impose par l'interface mais non utilise
    }

    public void afterTextChanged (Editable s)
    {
       // impose par l'interface mais non utilisee
    }

}
```

La fonction onTextChanged modifie le label de sélection pour qu'il reflète le choix courant du champ AutoCompleteTextView. La fonction afterTextChanged est employée pour notifier la vue dans laquelle elle est appelée que dans une partie de la variable s, le texte a été modifié.
La fonction beforeTextChanged permet de notifier la vue dans laquelle elle est appelée que dans la séquence de chaine 's', il y a 'count' caractères, commençant à partir de l'indice 'start' sont en train d'être modifiés par un texte de longueur 'after'.

3. Traitement des ListView

L'humble ListView est l'un des widgets les plus importants et les plus utilisés d'Android. Que l'on choisisse un contact téléphonique, un courrier à faire suivre ou un ebook à lire, c'est ce widget dont on se servira le plus souvent.

Dans notre cas, nous allons créer une liste de contacts et dès que l'utilisateur choisi un nom de ce contacte, on va lui afficher son numéro de téléphone. Pour ce faire, nous allons suivre les étapes suivantes :

Etape 1. Créer un nouveau projet nommé Contact_list

Application name : Contact_list
Activity name : Contact_list

Etape 2. Créer une classe nommée Contact.java

1. Faire **New / Class**.
2. Choisir ensuite **Contact** comme nom de classe.

On peut définir de manière très simple la classe contact avec uniquement un constructeur et une méthode de classe permettant de créer une liste de contacts. Le code de cette classe est comme suit :

```
package com.example.contact_list;
import java.util.ArrayList;

public class Contact {

        // déclaration des attributs de la classe Contact
           public String nom;
              public String prenom;
              public String telephone;

              // le constructeur de la classe
              public Contact(String aNom, String aPrenom, String aTelephone) {
              nom = aNom;
              prenom = aPrenom;
              telephone = aTelephone;
              }
               // création de l'adaptateur qui initialise et permet de gérer une liste
                //de contact

              public static ArrayList<Contact> Initialiser ()
              {
              ArrayList<Contact> listContact = new ArrayList<Contact>();
              Contact MonContact = new Contact("Dupont", "Thierry", "0124524521");
              listContact.add(MonContact);
              MonContact = new Contact("Tournesol", "Philippe", "054878569");
              listContact.add(MonContact);
              MonContact = new Contact("Martin", "Pecheur", "048578544");
              listContact.add(MonContact);
```

```
            MonContact = new Contact("Castafiore", "Helene", "08985785");
            listContact.add(MonContact);
            MonContact = new Contact("Dalton", "Joe", "0356898547");
            listContact.add(MonContact);
            MonContact = new Contact("Dalton", "Ma", "9874587444");
            listContact.add(MonContact);
            MonContact = new Contact("Obelix", "Gros", "025445836");
            listContact.add(MonContact);
            return listContact;

}
}
```

Etape 3. Changer le layout

Créer un nouveau layout linéaire appelé liste_contact et insérer deux TextView dont les identificateurs seront nommés Nom et Prénom. Dans la palette du layout, choisir le widget ListView se trouvant dans le répertoire Composite et insérer le dans le layout. Le code XML du fichier doit être comme suit :

```
<?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/Nom"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nom" />

    <TextView
        android:id="@+id/Prenom"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Prenom" />

    <ListView
        android:id="@+id/listView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>

</LinearLayout>
```

Etape 4. Créer une classe nommée ContactAdapter

Pour pouvoir gérer la structure de la classe Contact, il faut créer un objet de type adaptateur qui se chargera de faire le mapping entre nos données et le layout. Cet objet sera basé sur le principe d'Adapter, seulement cette classe n'offre qu'un type d'adaptateur prédéfini (ArrayAdapter) dont les éléments sont de type simple (entier, caractère, chaine). Or l'élément de la ListeView est une

structure (contact), par la suite, il faut créer la classe ContactAdapter qui hérite de la classe BaseAdapter et gère la structure Contact. Il faut procéder comme suit :

1. Faire **New / Class**
2. Nommer la classe ContactAdapter

Le code généré par défaut est le suivant :

```
package com.example.contact_list;

public class ContactAdapter {

}
```

Ce code doit être changé comme suit :

```
package com.example.contact_list;
import java.util.List;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.LinearLayout;
import android.widget.TextView;

public class ContactAdapter extends BaseAdapter {
	// Une liste de contact
	private List<Contact> mListP;
	//Le contexte dans lequel est présent notre adapter
	private Context mContext;
	//Un mécanisme pour gérer l'affichage graphique depuis un layout XML
	private LayoutInflater mInflater;


	// le constructeur
	public ContactAdapter(Context context, List<Contact> aListP) {
	mContext = context;
	mListP = aListP;
	//Le LayoutInflater permet de parser un layout XML et de le transcoder en IHM
Android.
	mInflater = LayoutInflater.from(mContext);
	}


	//Pour respecter l'interface BaseAdapter, il nous faut spécifier la méthode
"getcount()".
	public int getCount() {
	return mListP.size();
	}

	//retirer un élément de la liste
	public Object getItem(int position) {
	return mListP.get(position);
	}
```

```
        // fournir la position d'un élément de la liste
        public long getItemId(int position) {
        return position;
        }

        // Maintenant il faut surcharger la méthode pour renvoyer une "View"
        // en fonction d'une position donnée. le View pour notre cas est un élément de
la listeView
        //et qui correspond a un contact

        public View getView(int position, View convertView, ViewGroup parent) {

              LinearLayout layoutItem;
        //(1) : Réutilisation des layouts

        if (convertView == null) {
        //Initialisation de notre item à partir du layout XML layout.xml"
        layoutItem = (LinearLayout) mInflater.inflate(R.layout.liste_contact, parent,
false);
        }
        else
        {
        layoutItem = (LinearLayout) convertView;
        }
        //(2) : Récupération des TextView de notre layout
        TextView tv_Nom = (TextView)layoutItem.findViewById(R.id.Nom);
        TextView tv_Prenom = (TextView)layoutItem.findViewById(R.id.Prenom);

        //(3) : Renseignement des valeurs

        tv_Nom.setText(mListP.get(position).nom);
        tv_Prenom.setText(mListP.get(position).prenom);

        //On retourne l'item créé.
        return layoutItem;
        }

}
```

Etape 5. Créer une interface ContactAdapter pour le traitement du numéro de téléphone

Comme l'espace d'affichage du téléphone portable est toujours réduit, nous avons décidé d'afficher uniquement le nom et le prénom de notre structure Contact et dès que l'utilisateur veut savoir le numéro de téléphone d'une personne, il n'a qu'à cliquer dessus et une boite d'affichage apparait contenant le téléphone de la personne. Pour ceci nous devons créer des Listener sur notre structure Contact. Java permet ceci moyennant les interfaces. Ainsi, nous allons dans la même classe ContactAdapter ajouter une interface pour gérer les Listener.

1. Ajouter import java.util.ArrayList ;
   Import android.view.View.OnClickListener ; dans la partie import de la classe.

2. Ajouter à la suite du code la classe et avant l'}' qui ferme la classe le code suivant :
3.

```
//Interface pour écouter les évènements sur le nom d'un contact

public interface ContactAdapterListener {

// la fonction qui sera implémentée par toute classe qui utilise l'interface

public void onClickNom(Contact item, int position);
}

//création d'une liste d'écouteurs

private ArrayList<ContactAdapterListener> mListListener = new
ArrayList<ContactAdapterListener>();

// ajouter un listener sur notre liste de contacte
public void addListener(ContactAdapterListener aListener)
 {
mListListener.add(aListener);
}
//permet de activer tous les listeners de notre liste
private void sendListener(Contact item, int position) {
for(int i = mListListener.size()-1; i >= 0; i--) {
mListListener.get(i).onClickNom(item, position);
}
// le traitement a effectuer lorsque l'utilisateur clique sur le nom
OnClickListener ClickListener = new OnClickListener() {
@Override
 public void onClick(View v) {

//Lorsque l'on clique sur le nom, on récupère la position de la "Personne"
Integer position = (Integer)v.getTag();

//On vas réactiver les listeners pour qu'ils re-pointent sur le TextView "Nom".
 sendListener(mListP.get(position), position);
}
};
```

Etape 6. Modification de la méthode getView.

Une fois les listeners sont definis sur notre ListView des contacts, ajouter à la fin de la fonction getView et juste apres :

```
tv_Nom.setText(mListP.get(position).nom);
tv_Prenom.setText(mListP.get(position).prenom);
```

les deux lignes suivantes :

```
// (4) : On mémorise la position de la "Contact" dans le composant textview
tv_Nom.setTag(position);

//On ajoute un listener
tv_Nom.setOnClickListener(ClickListener);
```

Etape 7. Modification de la classe Contact_list

Dans le fichier Contact_list.java, comme d'habitude enlever tous les imports concernant les menus et fragment ainsi que le code correspondant. Dans la partie import de votre code , vous devez avoir tous les packages suivants :

```
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import java.util.ArrayList;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.widget.ListView;
import com.example.contact_list.Contact;
import com.example.contact_list.ContactAdapter;
import com.example.contact_list.ContactAdapter.ContactAdapterListener;
```

Modifier la classe Contact_list.java pour le rendre comme suit (sans oublier le changer le nom du layout dans l'activité) :

```
public class Contact_List extends ActionBarActivity implements ContactAdapterListener
{

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.liste_contact);

      //Récupération de la liste des contacts

        ArrayList<Contact> listP = Contact.Initialiser();

        //Création et initialisation de l'Adaptateur pour les contacts

        ContactAdapter adapter = new ContactAdapter(this, listP);

        adapter.addListener(this);

        //Récupération du composant ListView

        ListView list = (ListView)findViewById(R.id.listView1);

        //Initialisation de la liste avec les données

        list.setAdapter(adapter);


    }
```

```
    public void onClickNom(Contact item, int position) {

       // création d'une boite de dialogue dont le contexte d'affiche est l'activité
       //en cours pointée par this

       Builder builder = new AlertDialog.Builder(this);

      // écrire le titre de de la boite de dialogue

       builder.setTitle("Information sur le contact");

    // affichage du téléphone du contact sélectionné par l'utilisateur

       builder.setMessage("Le telephone est : " + item.telephone);

    // creation d'un bouton ok dans la boite de dialogue
       builder.setPositiveButton("OK", null);

      // affichage de la boite de dialgue
       builder.show();
       }


}
```