VII. ListView

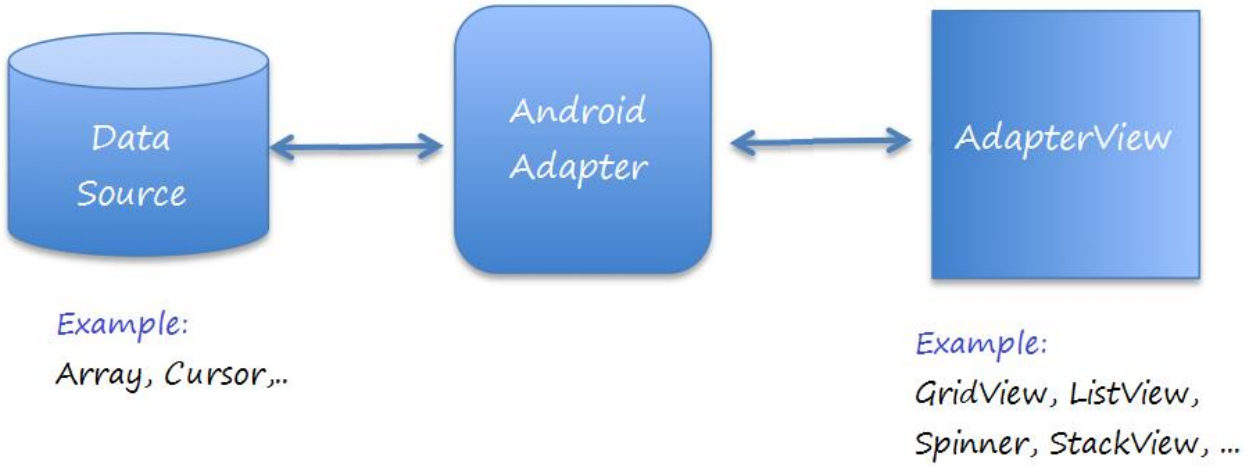
Un **ListView** est fait à partir de **ListItem**. Le ListItem est une ligne individuelle où les données seront affichées.

Le ListItem

Le **ListItem** est une ligne dans la **ListView** qui sert de modèle où les données sont organisées. Nous pourrons créer des **Layout** pré-défini pour les ListItem.

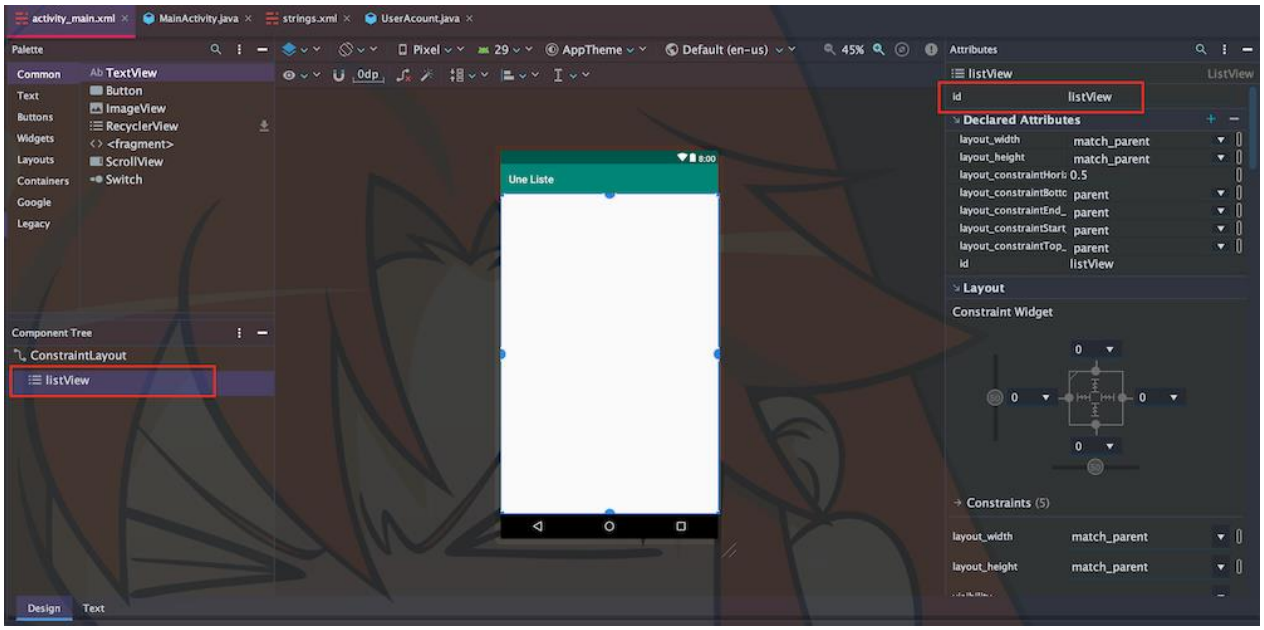
Les Adapter

Pour alimenter une **ListView** nous devons définir un **Adapter** qui gère les données et les adapte pour la ListView

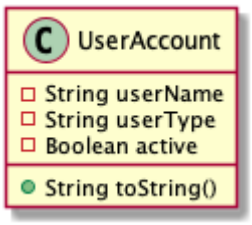


Construisons ensemble un premier exemple

- Tout d'abord construisez un layout avec une `ListView` que vous appellerez listview dans votre `activity_main.xml`



* Créez une classe `UserAccount`



La méthode `toString()` est une surcharge de `toString()`

```
@Override
public String toString(){
    return this.userName+"("+this.userType+")";
}
```

Cette surcharge va convertir notre objet en String lisible par la ListView.

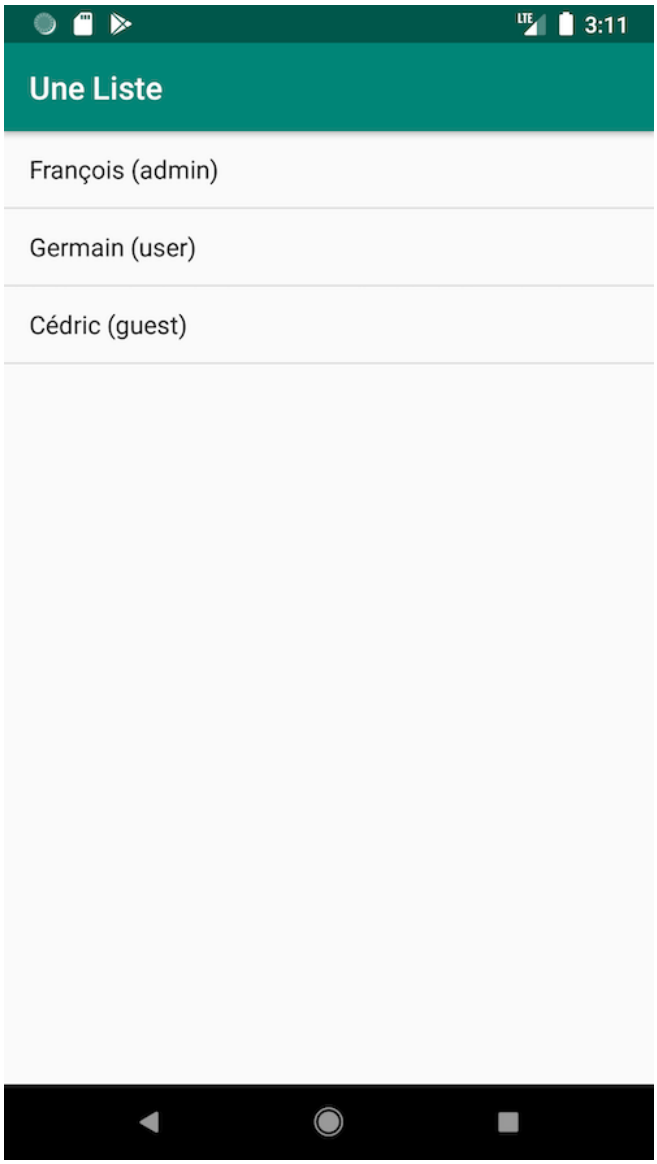
- Dans le `MainActivity` nous allons récupérer la listView
- Créer des instances de `UserAccount`
- Alimenter une liste de `UserAccount`
- Et les mettre dans un adapter

```
 // Récupération de listview
ListView listView = (ListView)findViewById(R.id.listView);
//Création des users
UserAccount francois = new UserAccount("François","admin");
UserAccount germain = new UserAccount("Germain","user");
UserAccount cedric = new UserAccount("Cédric","guest",false);
//Création de la liste
UserAccount[] users = new UserAccount[]{francois,germain,cedric};
```

```
//Création de l'adapter
ArrayAdapter<UserAccount> arrayAdapter = new ArrayAdapter<UserAccount>(this,
android.R.layout.simple_list_item_1,users);
//Affectation de l'adapter
listView.setAdapter(arrayAdapter);

for (int i = 0; i < users.length; i++) {
      listView.setItemChecked(i, users[i].isActive());
}
```

Vous devez obtenir



Personnalisation de ListView en utilisant BaseAdapter

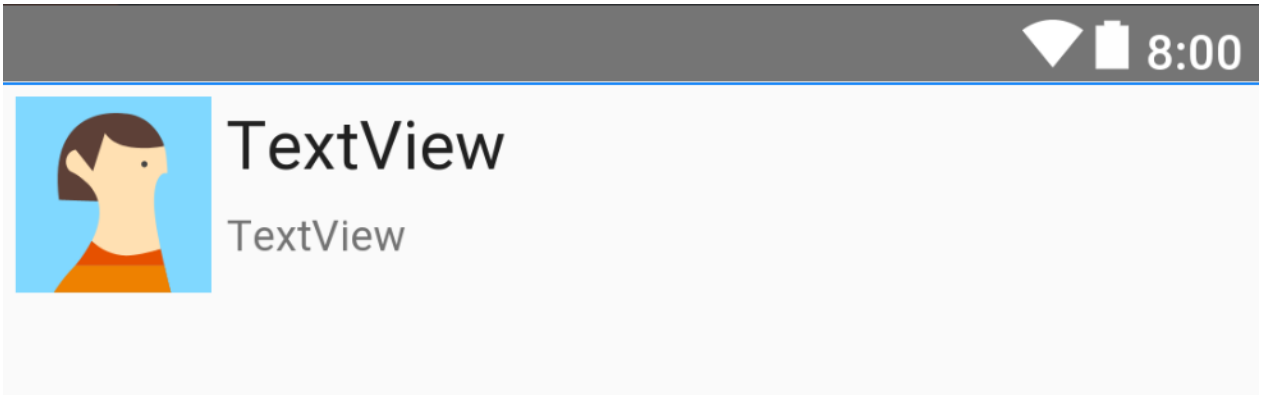
- Commençons par créer notre propre `Adapter`

Tout d'abord concevons un layout

New->Android Resource File et choisissez layout

Nommez la `liste_item_layout`

Vous devrez ajouter deux `TextField` et une `ImageView` pour obtenir le résultat suivant :



Votre xml devrait ressembler à ça:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/linearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/imageView_role"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:layout_margin="5dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:srcCompat="@tools:sample/avatars" />

    <TextView
        android:id="@+id/textView_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:text="TextView"
        android:textAppearance="@style/TextAppearance.AppCompat.Large"
        app:layout_constraintStart_toEndOf="@+id/imageView_role"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/textView_role"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:layout_marginStart="4dp"
        android:layout_marginTop="16dp"
        android:text="TextView"
        android:textAppearance="@style/TextAppearance.AppCompat.Small"
        app:layout_constraintStart_toEndOf="@+id/imageView_role"
        app:layout_constraintTop_toBottomOf="@+id/textView_name" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Construisons maintenant notre CustomAdapter.

Créez une classe `CustomListAdapter` qui va hériter de `BaseAdapter`. Puis implémentez les méthodes héritées.

- getCount():

```
public int getCount() {
        return listUser.size();
    }
```

- getItem(int i)

```
public Object getItem(int position) {
        return listUser.get(position);
    }
```

- getItemId(int i)

```
public long getItemId(int position) {
        return position;
    }
```

- getView(int i, View view, ViewGroup viewGroup) que vous laissez vide pour le moment.

Notre Adapter a 3 attributs :

```
private List<UserAccount> listUser;
private LayoutInflater layoutInflater;
private Context context;
```

Son constructeur :

```
public CustomListAdpater(List<UserAccount> listUser, Context context) {
        this.listUser = listUser;
        layoutInflater = LayoutInflater.from(context);
        this.context = context;
    }
```

le LayoutInflater sert à déployer le customlayout

Une classe statique pour le ViewHolder qui va contenir les infos de la vue :

```
static class ViewHolder{
        ImageView rolePicView;
        TextView userNameView;
        TextView roleView;
    }
```

Une méthode pour charge les images dans le Mipmap en fonction du nom de l'image :

```
public int getMipmapResIdByName(String resName)  {
        String pkgName = context.getPackageName();
        // Return 0 if not found.
        int resID = context.getResources().getIdentifier(resName , "mipmap",
pkgName);
        Log.i("unliste", "Res Name: "+ resName+"==> Res ID = "+ resID);
        return resID;
    }
```

vous pouvez utiliser les images suivantes que vous mettrez dans le dossier res/mipmap

Enfin créons la méthode de chargement getView():

```
public View getView(int positon, View convertView, ViewGroup parent) {
        ViewHolder holder;
        if(convertView == null){
            convertView =
layoutInflater.inflate(R.layout.list_item_layout,null);
            holder = new ViewHolder();
            holder.rolePicView = (ImageView)
convertView.findViewById(R.id.imageView_role);
            holder.userNameView = (TextView)
convertView.findViewById(R.id.textView_name);
            holder.roleView = (TextView)
convertView.findViewById(R.id.textView_role);
            convertView.setTag(holder);
        } else {
            holder = (ViewHolder) convertView.getTag();
        }

        UserAccount user = this.listUser.get(positon);
        holder.userNameView.setText(user.getUserName());
        holder.roleView.setText("Role: "+ user.getUserType());

        int imageId = this.getMipmapResIdByName(user.getUserType());

        holder.rolePicView.setImageResource(imageId);
        return convertView;
    }
```

Utilisation du CustomAdpater dans le MainActivity

Nous allons simplement ajouter

`listView.setAdapter(new CustomListAdpater(listCli, this));` dans la méthode `onCreate` .

Vous devriez obtenir : lien github branche `master` pour le premier exemple et `etape2` pour le second.