RecyclerView

Le widget ListView est déprécié à partir de l'API 30 (Android 11). Google recommande l'utilisation de RecyclerView à la place.

RecyclerView permet de répéter dynamiquement la vue dans une fenêtre réduite d'une large quantité de données qui changent fréquemment.

Architecture d'un RecyclerView

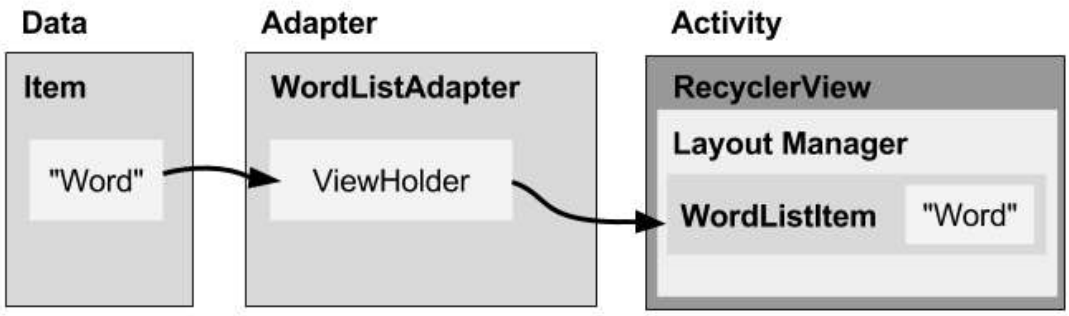
L'activité va fournir une donnée qu'elle aimerait afficher dans sa vue. Cette vue est un RecyclerView. Un RecyclerView va représenter une liste d'éléments. La vue de chaque élément doit-être préalablement définie.

Nous avons donc besoin de :

- ViewHolder : il permet de représenter un élément de la liste.

- Adapter : Il permet de faire la connexion entre les données et la vue.

- LayoutManager : il permet de positionner les éléments dans un RecyclerView.



Dépendances

Nous avons besoin d'ajouter cette dépendance dans le fichier « build.gradle » du module associé au projet.

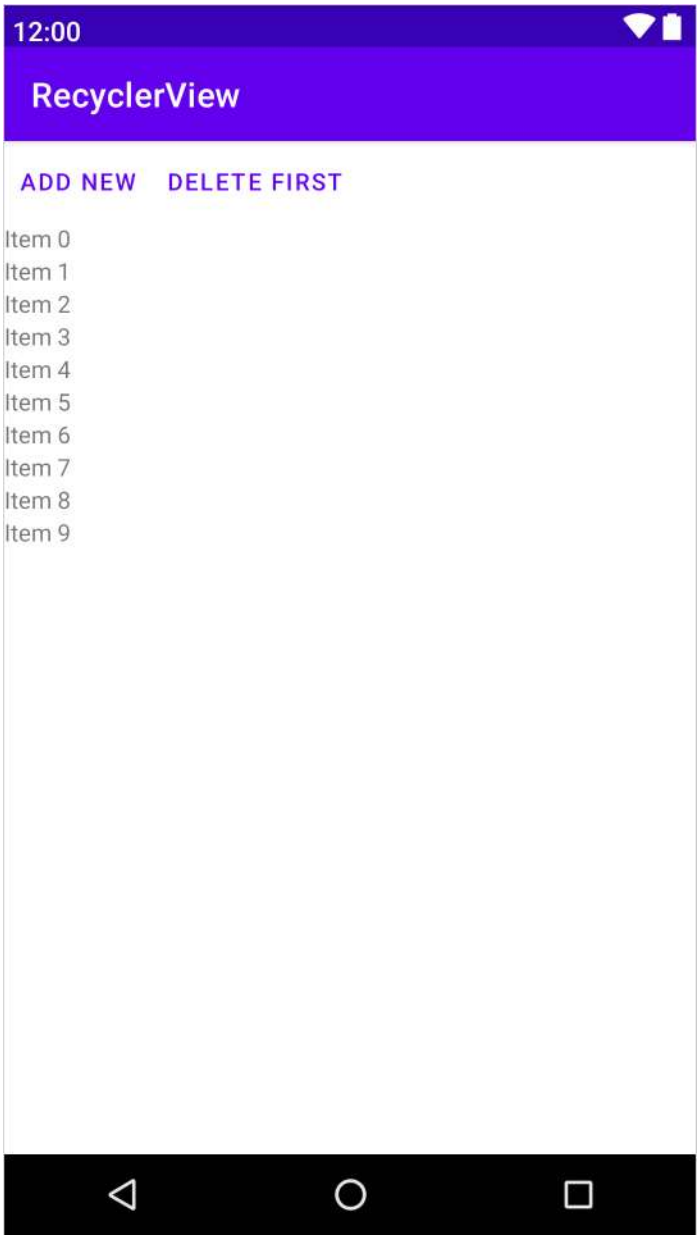
Nom_du_projet/app/build.gradle

```
dependencies {
    implementation 'androidx.recyclerview:recyclerview-selection:1.1.0'
}
```

La vue de l'interface

Nous allons utiliser l'exemple du chapitre 7 simplifié de sa base de données. L'interface finale est sous cette forme :

Les éléments seront disposés verticalement comme suit :



Dans l'interface principale, nous devons définir deux boutons avec respectivement les deux actions : ajouter un item à la liste et effacer un item de la liste. Par la suite, nous devons définir un « RecyclerView » qui va contenir l'ensemble des items.

```
<?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"
    android:id="@+id/linearLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">


    <Button
        android:id="@+id/add"
        style="?android:attr/buttonBarButtonStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="@string/Add_New"
        app:layout_constraintBaseline_toBaselineOf="@+id/delete"
        app:layout_constraintStart_toStartOf="@+id/recyclerview" />

    <Button
        android:id="@+id/delete"
        style="?android:attr/buttonBarButtonStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="@string/Delete_First"
        app:layout_constraintStart_toEndOf="@+id/add"
        app:layout_constraintTop_toTopOf="parent" />


    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerview"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/add" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

La vue d'un item

Chaque item doit avoir une certaine forme dans la liste. Nous allons définir cette forme dans le fichier « recyclerviewitem.xml » comme suit :

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

    <TextView
        android:id="@+id/item_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:padding="8dp"
        android:textSize="25sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

On constate que chaque item est représenté par un « TextView » ayant une suite de caractéristiques.

Un commentaire

Chaque item de la liste va contenir un commentaire. Nous devons définir le contenu d'un commentaire. Nous allons le faire à travers la classe « Commentaire.java ». Cette classe va contenir tous les get/set nécessaires pour afficher/modifier les éléments de la classe.

Chaque commentaire est caractérisé par ces deux variables :

id : une variable du type long qui identifie le commentaire.

commentaire : une variable du type String qui représente le commentaire.

```
package ca.umontreal.iro.ift1155.recyclerview;

import androidx.annotation.NonNull;

public class Commentaire {

    private long id;
    private String commentaire;

    long getId() {
        return id;
    }
    void setId(long id) {
        this.id = id;
    }
    public String getComment() {
        return commentaire;
    }
    Commentaire(long _id, String _commentaire){
        id = _id;
        commentaire = _commentaire;
    }
    void setComment(String comment) {
        this.commentaire = commentaire;
    }

    // Il est nécessaire pour  ArrayAdapter dans RecyclerView
    @Override
    @NonNull
    public String toString() {
        return commentaire;
    }
}
```

La classe d'adaptateur et association à la vue

La classe adaptateur a besoin d'hériter de la classe « RecylerView.Adapter ». Nous avons par la même occasion défini dans la même classe, la classe « ViewHolder » qui va afficher les données de l'élément (le commentaire).

Nous aurions pu séparer ces deux classes et les définir donc dans des fichiers à part.

La classe « AdapterClass » a besoin de définir ces 3 méthodes :

`onCreateViewHolder` : cette méthode crée les « ViewHolder » en fonction des besoins.

onBindViewHolder : cette méthode met à jour l'affichage de la vue d'un élément (en tenant compte de sa position dans la liste).

getItemCount : cette méthode retourne le nombre d'éléments dans la liste.

Nous avons ajouté dans cette classe un constructeur. Ce dernier permet d'initialiser la variable associée au tableau de commentaires.

La classe « ViewHolder » hérite de la classe « RecyclerView.ViewHolder ». Elle va permettre d'afficher les données de l'élément (ici un commentaire) dans la vue (la variable layout).

```
package ca.umontreal.iro.ift1155.recyclerview;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class AdapterClass extends RecyclerView.Adapter<AdapterClass.ViewHolder> {

    ArrayList<Commentaire> data;

    public AdapterClass(ArrayList<Commentaire> data) {
        this.data = data;
    }

    @NonNull
    @Override
    public AdapterClass.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return new ViewHolder(LayoutInflater.from(parent.getContext()).inflate(R.layout.recyclerviewitem, parent, false));
    }

    @Override
    public void onBindViewHolder(@NonNull AdapterClass.ViewHolder holder, int position) {
        holder.name.setText(data.get(position).getComment());
    }

    @Override
    public int getItemCount() {
        return data.size();
    }

    public static class ViewHolder extends RecyclerView.ViewHolder{
        TextView name;
        ConstraintLayout layout;
        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            name = itemView.findViewById(R.id.item_name);
            layout = itemView.findViewById(R.id.layout);
        }
    }
}
```

La classe principale

Nous allons définir dans cette classe les différents éléments et faire l'association entre eux.

Nous allons commencer par définir une instance de « RecyclerView », de la classe « AdapterClass » que nous avons définie, et un tableau de commentaires, initialisé avec une liste vide.

```
RecyclerView mRecyclerView;
ArrayList<Commentaire> data = new ArrayList<>();
AdapterClass adapter;
```

Dans la méthode « onCreate », nous allons faire l'association entre les différents éléments :

On associe la vue d'un item du « RecyclerVue » :

```
mRecyclerView = findViewById(R.id.recyclerview);
```

On associe l'adaptateur à une nouvelle instance, initialisée avec « data ». La variable « data » est un tableau vide pour le moment, sous la forme d'une liste.

```
adapter = new AdapterClass(data);
```

On connecte le « RecyclerVue » à son adaptateur.

```
mRecyclerView.setAdapter(adapter);
```

On fixe la mise en page des items. Nous avons choisi de les disposer verticalement.

```
mRecyclerView.setLayoutManager(new LinearLayoutManager(this, RecyclerView.VERTICAL, false));
```

L'action des boutons est définie pour réaliser la tâche d'ajouter un nouveau commentaire dans la liste ou d'en retirer le dernier commentaire inséré de la liste. Pour chacune des situations, nous allons soit utiliser la méthode « add » ou « remove ». Ces deux méthodes sont transparentes et sont définies par défaut avec la classe « AdapterClass ».

Suite à chaque changement, nous devons rafraîchir la liste. Pour éviter de rafraîchir toute la liste, nous pouvons spécifier un item en particulier à l'aide de la méthode « adapter.notifyItemInserted() » pour l'ajout et « adapter.notifyItemRemoved() » pour le retrait. On peut aussi mentionner les changements par « adapter.notifyDataSetChanged() ». Sauf que cette approche va rafraîchir toute la liste. Si la liste est trop longue, cette opération risque de prendre un certain temps. Elle est donc déconseillée si la position de l'item est connue.

```
package ca.umontreal.iro.ift1155.recyclerview;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;
import android.util.Log;
import android.view.View;

import java.util.ArrayList;
import java.util.Random;

public class MainActivity extends AppCompatActivity {

    RecyclerView mRecyclerView;
    ArrayList<Commentaire> data = new ArrayList<>();
    AdapterClass adapter;
    long id = 0;
    String TAG = "DIRO";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mRecyclerView = findViewById(R.id.recyclerview);
        adapter = new AdapterClass(data);
        mRecyclerView.setAdapter(adapter);
        mRecyclerView.setLayoutManager(new LinearLayoutManager(this, RecyclerView.VERTICAL, false));
    }

    public void onClick(View view) {
        Commentaire comment;
        if (view.getId() == R.id.add) {
            String[] comments = new String[] { "Cool", "Very nice", "Hate it" };
            int nextInt = new Random().nextInt(3);
            comment = new Commentaire(id,comments[nextInt]);
            id++;
            data.add(comment);
            adapter.notifyItemInserted((int) id);
            Log.i(TAG,"add: " + comment.getComment() + " size: " + adapter.getItemCount());
        }
        if (view.getId() ==  R.id.delete){
            if (adapter.getItemCount() > 0){
                comment = data.get(0);
                data.remove(comment);
                adapter.notifyItemRemoved(0);
                Log.i(TAG,"del: " + comment.getComment() + " size: " + adapter.getItemCount());
            }
        }
    }
}
```

Bibliographie

RecyclerView | Android Developers
https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.html

Create dynamic lists with RecyclerView | Android Developers
https://developer.android.com/guide/topics/ui/layout/recyclerview

views-widgets-samples/RecyclerView at main · android/views-widgets-samples (github.com)
https://github.com/android/views-widgets-samples/tree/main/RecyclerView/