RecyclerView

Une RecyclerView est une nouvelle façon d'afficher une liste ou une grille de vues.

Elle peut s'apparenter à une ListView mais permet beaucoup plus de personnalisation.

Cette nouvelle vue est disponible dans la librairie de support v7 d'android, il vous faut donc commencer par importer dans votre fichier gralde :

```
    dependencies {
     implementation 'com.android.support:recyclerview-v7:28.0.0'
}
```

Ou bien :

```
    implementation "androidx.recyclerview:recyclerview:1.1.0"
```

Comme toutes vues, il faut la déclarer dans nos layout xml :

```
    <?xml version="1.0" encoding="utf-8"?>
<!-- A RecyclerView with some commonly used attributes -->
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:scrollbars="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

```
    <?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">

    <TextView
            android:text="TextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:id="@+id/tv_nom"
android:layout_weight="1"/>
    <TextView
            android:text="TextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:id="@+id/tv_num"
android:layout_weight="1"/>
</LinearLayout>
```

A la même façon que les ListView, il faut créer un Adapter, mais cette fois ci il doit étendre RecyclerView.Adapter<Votre type (içi Element)>. Cet adapter est typé pour un fonctionner avec une certaine classe de ViewHolder (objet qui va garder les références vers les vues de chaque cellule). Dans notre cas il va donc falloir créer un ViewHolder, nous le nommerons MyViewHolder.

```
EN KOTLIN
class Element ( var nom :String , numero :Int)
En JAVA
class Element {
    public String nom, numero;

    public Element(String nom, String numero) {
        this.nom = nom;
        this.numero = numero;
    }
}
```

EN KOTLIN

```kotlin
class MyAdapter (val con:Context,val myData: Array<Element>)
    : RecyclerView.Adapter<MyAdapter.MyViewHolder>()
{
    override fun onCreateViewHolder(p0: ViewGroup, position: Int): MyViewHolder {
        // Creation nouvelles vues
        val layout = LayoutInflater.from(con)
            .inflate(R.layout.layout, null) as LinearLayout

        return MyViewHolder(layout)// recuperation des sous composants
    }

    override fun getItemCount(): Int {
        // definier la taille
       return myData.size
    }

    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        // Mettre à jour un vue déjà crée

        var e=myData.get(position)
        // récupérer l'élément depuis la liste des données myData

        holder.tvnom.text=e.nom
        holder.tvnum.text=e.num.toString()
        // modifier le contenu d'un vue par les valeurs de cet élément
    }

    inner class MyViewHolder(parent: View) : RecyclerView.ViewHolder(parent)
    {
        //recuperation des sous composant du layout depuis son Context : parent
        var tvnom=parent.findViewById(R.id.tv_nom) as TextView
        var tvnum=parent.findViewById(R.id.tv_num) as TextView
    }
}
```

En JAVA

```java
class MonAdapter extends RecyclerView.Adapter <MonAdapter.MyViewHolder>{
    Context context;
    ArrayList<Element> data;

    public MonAdapter(Context context, ArrayList<Element> data) {
        this.context = context;
        this.data = data;
    }

    class MyViewHolder extends RecyclerView.ViewHolder
    {
        TextView t1;
        Button b;
        public MyViewHolder(@NonNull View itemView) {
            super(itemView);
            t1=itemView.findViewById(R.id.t1);
            b=itemView.findViewById(R.id.b);
        }
    }
```

```
    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        LayoutInflater inf=LayoutInflater.from(context);
        LinearLayout l= (LinearLayout) inf.inflate(R.layout.element,parent,false);
        return new MyViewHolder(l);
    }

    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
        Element c=data.get(position);
        holder.t1.setText(c.nom);
        holder.b.setText(c.numero);
    }

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

Utilisons le maintenant dans notre Activity

Une étape importante est d'affecter un LayoutManager à notre activity, sinon une erreur sera levée : RecyclerView : No layout manager attached; skipping layout

EN KOTLIN

```
        // in content do not change the layout size of the RecyclerView
        lv.setHasFixedSize(true)
//définit l'agencement des cellules, ici de façon verticale, comme une ListView
        lv.setLayoutManager(LinearLayoutManager(getApplicationContext()));

        // specify an viewAdapter (see also next example)
        lv.adapter = viewAdapter
```

En JAVA

```
        MonAdapter ad=new MonAdapter(this, data);
        rv.setAdapter(ad);
        rv.setLayoutManager(new LinearLayoutManager(this));
//pour adapter en grille comme une RecyclerView, avec 2 cellules par ligne
//recyclerView.setLayoutManager(new GridLayoutManager(this,2));
//rv.setLayoutManager(new
//LinearLayoutManager(this,LinearLayoutManager.HORIZONTAL,false));
```

Pour plus de détails :

https://guides.codepath.com/android/using-the-recyclerview