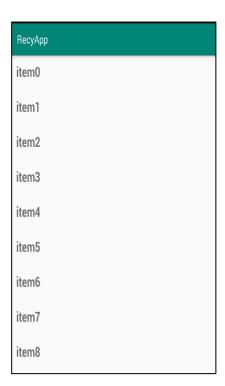


# TP N°5

# RecyclerView et CardView

## 1. Création de RecyclerView simple

Il s'agit de développer un exemple simple affichant un RecyclerView de chaines de caractères comme suit :



#### Q1. Créer un projet nommé TPRecycler

### Etape1: préparer les vues XML:

Q2. Ajouter un RecyclerView à votre layout de l'activité principale

```
<androidx.recyclerview.widget.RecyclerView
android:layout_width="match_parent"
android:layout_height="match_parent"
android:id="@+id/recyclerview" />
```

**Q3.** Créer le layout destiné pour une ligne du recycler **ligne.xml** contenant un Textview (utiliser la mise en forme suivante)

LinearLayout attribute	Value
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:orientation	"vertical"
android:padding	"6dp"

TextView Attribute	Value
android:id	"@+id/word"
android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:textSize	"24sp"
android:textStyle	"bold"

## **Etape2: Les classes Adapter et ViewHolder:**

Q4. Définir une classe MyAdapter qui hérite de la classe RecyclerView. Adapter

```
class MyAdapter(private val myDataSet: ArrayList<String>):
    RecyclerView.Adapter<MyAdapter.ViewHolder>()
```

Q5. Créer la classe interne MyViewHolder comme suit :

```
class ViewHolder(val itemview: View):
    RecyclerView.ViewHolder(itemview) {
    val vText = itemView.findViewById(R.id.word) as TextView
}
```

#### Q6. Définir les 3 méthode de la classe de notre adapter

a) Définir la méthode on Create View Holder comme suit :

b) Définir la méthode onBindViewHolder comme suit:

```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    var current = myDataSet[position]
    holder.vText.text = current.toString()
}
```

c) Définir la méthode getItemCount() comme suit:

```
override fun getItemCount(): Int {
   return myDataSet.size
}
```

## **Etape3 : Créer la source de données et lier le recycler aux classes Adapter**

#### et LayoutManager:

#### Q7. Dans la classe MainActivity.java

a) Déclarer les données à afficher dans notre RecycleView dans une liste de String comme suit :

```
var values= arrayListOf<String>("item1", "item2", "item3", "item4", "item5")
```

b) déclarer le RecyclerView, l'adaptateur et le LayoutManager comme suit :

```
private lateinit var recyclerView: RecyclerView
private lateinit var manager: RecyclerView.LayoutManager
private lateinit var myAdapter: RecyclerView.Adapter<*>
```

c) instancier la classe adapter, layoutManager et recyclerView et lier l'adapter et le layoutManager au recyclerView comme suit :

```
manager = LinearLayoutManager(this)
myAdapter = MyAdapter(values)

recyclerView = findViewById<RecyclerView>(R.id.recyclerView).apply {
    layoutManager = manager
    adapter = myAdapter
}
```

## 2. Ajouter un élément à la liste

- a) Ajouter un bouton au layout de l'activité principale
- b) Dans l'activité main ajouter l'évènement onclickLisner du bouton permettant d'ajouter l'élément suivant de la liste comme suite:

```
var n:Int = values.size + 1
values.add("item$n") //Ajout dans la source de données
myAdapter.notifyItemInserted(values.size)//Rafraichissement de l'adapter
```

## 3. Programmer le clic sur un élément de la liste

L'objectif est d'implémenter le click sur le ViewHolder : Si un élément de la liste est cliqué, un toast est affiché en précisant l'élément cliqué et dans la liste cet élément est changé « cliqued »

#### **Etape1 : Création de l'interface**

Q1. Dans la classe MyAdapter, déclarer une interface OnItemClickListener avec une méthode onItemClick

```
interface OnItemClickListener {
    fun onItemClick(position: Int)
}
```

#### **Etape 2 : Au niveau du Viewholder**

Q2. Modifier la déclaration de la classe ViewHolder comme suite :

```
inner class ViewHolder(val itemview: View):
RecyclerView.ViewHolder(itemview), View.OnClickListener {
```

- N.B! Une classe interne déclaré inner peut accéder aux membres de sa classe externe
- Q3. Dans le ViewHolder, ajouter le code suivant :

```
init {
    itemView.setOnClickListener(this)
}
```

```
override fun onClick(v: View?) {
   val position = adapterPosition
   if (position != RecyclerView.NO_POSITION) {
        listener.onItemClick(position)
   }
}
```

#### Etape 3 : Au niveau de l'Adapter

**Q4.** Ajouter à la classe **MyAdapter** un paramètre « listener » ayant comme type l'interface créée. La déclaration devient alors :

```
class MyAdapter(
    private val myDataSet: ArrayList<String>,
    private val listener: OnItemClickListener):
    RecyclerView.Adapter<MyAdapter.ViewHolder>()
{
```

## Etape 4 : Au niveau de l'activité principale

**Q5.** La classe **MainActivity** doit implémenter (implements en java) l'interface que nous avons créée

```
class MainActivity : AppCompatActivity(), MyAdapter.OnItemClickListener {
```

Q6. Dans MainActivity, implémenter la méthode onItemClick de l'interface

Q7. Changer l'appel à la classe MyAdapter

```
myAdapter = MyAdapter(values, this)
```

# 4. Travail à faire : RecyclerView et CardView

- **Q1**. Implémenter une Activité qui affiche la liste de 24 gouvernorats de la Tunisie chacun avec une photo
- **Q2.** Définir les lignes comme étant des cardview au lieu du layout utilisé. Faites les changements nécessaires.
- **Q3.** Implémenter l'événement click sur un élément de la liste qui affiche le nom du pays et sa photo dans une autre activité, qu'on nommera DetailActivity
- **Q4.** Ajouter un FloatingActionButton dans l'interface principale de l'application permettant de supprimer une ligne de votre choix dans la liste.