Séance 1 : `RecyclerView` - Introduction et Affichage Simple

Dr. GUEDDES Abdelweheb

Objectifs de la Séance

— Comprendre les bases du `RecyclerView` et son rôle dans l'affichage de listes, notamment pour des applications techniques.
— Savoir créer un `Adapter` simple pour une liste de données, en utilisant des exemples concrets incluant des images.
— Afficher une liste statique dans une activité en utilisant `RecyclerView`, en visualisant des données issues de capteurs ou d'expériences, avec des images représentatives.

Cours Théorique (30 minutes)

Introduction au `RecyclerView`

Le `RecyclerView` est un composant essentiel d'Android pour afficher des listes et des grilles de manière efficace. Il remplace avantageusement le `ListView`, souvent moins performant. En tant que futurs ingénieurs, vous serez amenés à traiter des ensembles de données importants, que ce soit des relevés de capteurs, des résultats d'expériences, des simulations, etc. Le `RecyclerView` permet d'afficher ces informations de manière fluide, même avec un grand nombre d'éléments. L'affichage d'images et de données ensemble rend l'application plus conviviale et plus riche en information.

Concepts Clés

— **Adapter** : L'adaptateur est responsable de créer les vues d'un élément de la liste et de les relier aux données. Il permet de faire le lien entre vos données et l'affichage.

— **ViewHolder** : Un ViewHolder est un objet qui contient des références aux vues d'un élément de la liste, améliorant ainsi les performances en évitant les recherches inutiles de vues. Imaginez un outil qui garde les accès prêts pour un travailleur, lui évitant de chercher ses outils à chaque fois.
— **LayoutManager** : Le LayoutManager gère la disposition des éléments dans le `RecyclerView` (par exemple, en liste verticale ou en grille). Il définit comment les données seront organisées sur l'écran, comme une organisation efficace de vos données dans un tableau ou un graphique.

TP Guidé Étape par Étape (2h30)

Étape 1 : Création d'un Nouveau Projet Android

1. Lancez Android Studio et créez un nouveau projet en choisissant le modèle "Empty Activity". Nommez-le "EPSSensorsApp" ou un nom similaire.
2. Assurez-vous d'avoir sélectionné Java comme langage et la version minimale de l'API souhaitée.

Étape 2 : Ajout de la Dépendance `RecyclerView`

Ajoutez la dépendance du 'RecyclerView' dans le fichier 'build.gradle (Module : app)' :

```
dependencies {
    implementation 'androidx.recyclerview:recyclerview:1.2.1'
    //Autres dépendances...
}
```

Listing 1 – build.gradle (Module: app)

Après la modification, cliquez sur "Sync Now" pour synchroniser votre projet.

Étape 3 : Création du Layout de l'Item - Avec Image

Créez un nouveau fichier de layout XML, 'item_sensor.xml', pour afficher des données de capteur avec une image :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/
    res/android"
    android:layoutwidth="match_parent"
```

```
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="16dp"
    android:gravity="center_vertical">

    <ImageView
        android:id="@+id/sensorImageView"
        android:layout_width="48dp"
        android:layout_height="48dp"
        android:layout_marginEnd="16dp"
        android:src="@drawable/ic_default_sensor" />

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:orientation="vertical">

        <TextView
            android:id="@+id/sensorNameTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/sensorValueTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="16sp" />

        <TextView
            android:id="@+id/sensorUnitTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="14sp"
            android:textColor="#808080" />
    </LinearLayout>
</LinearLayout>
```

Listing 2 – item_sensor.xml

Ce layout contient :

- Un `ImageView` pour afficher une image du capteur.
- Un `TextView` pour le nom du capteur.
- Un `TextView` pour la valeur du capteur.
- Un `TextView` pour l'unité de mesure.

Important : Ajoutez une image dans le répertoire "res/drawable" de votre projet. Vous pouvez créer une image simple ou utiliser une image de

base. Nommez cette image “ic_default_sensor.png“ ou “ic_default_sensor.xml“ pour qu’elle corresponde au code ci-dessus.

Étape 4 : Création de l’Adaptateur `SensorAdapter`

Modifiez la classe Java, ‘SensorAdapter‘, pour prendre en charge l’image. Les données seront représentées par une classe `SensorData` (à créer ensuite), modifiée pour y inclure l’id de l’image.

```
1  import android.view.LayoutInflater;
2  import android.view.View;
3  import android.view.ViewGroup;
4  import android.widget.ImageView;
5  import android.widget.TextView;
6  import androidx.annotation.NonNull;
7  import androidx.recyclerview.widget.RecyclerView;
8  import java.util.List;
9
10 public class SensorAdapter extends RecyclerView.Adapter<
       SensorAdapter.ViewHolder> {
11
12     private List<SensorData> data;
13
14     public SensorAdapter(List<SensorData> data) {
15         this.data = data;
16     }
17
18     @NonNull
19     @Override
20     public ViewHolder onCreateViewHolder(@NonNull ViewGroup
       parent, int viewType) {
21         View view = LayoutInflater.from(parent.getContext())
22                 .inflate(R.layout.itemsensor, parent, false);
23         return new ViewHolder(view);
24     }
25
26     @Override
27     public void onBindViewHolder(@NonNull ViewHolder holder,
       int position) {
28         SensorData item = data.get(position);
29         holder.sensorImageView.setImageResource(item.
       getImageResId());
30         holder.sensorNameTextView.setText(item.getName());
31         holder.sensorValueTextView.setText(String.valueOf(
       item.getValue()));
32         holder.sensorUnitTextView.setText(item.getUnit());
33     }
34
```

```
35      @Override
36      public int getItemCount() {
37          return data.size();
38      }
39
40      public static class ViewHolder extends RecyclerView.
    ViewHolder {
41          ImageView sensorImageView;
42          TextView sensorNameTextView;
43          TextView sensorValueTextView;
44          TextView sensorUnitTextView;
45
46          public ViewHolder(@NonNull View itemView) {
47              super(itemView);
48              sensorImageView = itemView.findViewById(R.id.
    sensorImageView);
49              sensorNameTextView = itemView.findViewById(R.id.
    sensorNameTextView);
50              sensorValueTextView = itemView.findViewById(R.id.
    sensorValueTextView);
51              sensorUnitTextView = itemView.findViewById(R.id.
    sensorUnitTextView);
52          }
53      }
54  }
```

Listing 3 – SensorAdapter.java

Explication du code :

- Le constructeur reçoit la liste de données `SensorData` à afficher.
- `onCreateViewHolder` crée une nouvelle vue pour un item, en utilisant le layout "item_sensor.xml".
- `onBindViewHolder` remplit la vue avec les données de l'objet `SensorData` de la position courante et définit la source de l'image.
- `getItemCount` retourne la taille de la liste.
- La classe `ViewHolder` contient les références vers les vues du layout item.

Modifiez ensuite la classe Java `SensorData` pour y inclure l'identifiant de l'image :

```
1  public class SensorData {
2      private String name;
3      private double value;
4      private String unit;
5      private int imageResId;
6
7      public SensorData(String name, double value, String unit,
       int imageResId){
```

```
        this.name = name;
        this.value = value;
        this.unit = unit;
        this.imageResId = imageResId;
    }
    public String getName(){
        return this.name;
    }
    public double getValue(){
        return this.value;
    }
    public String getUnit(){
        return this.unit;
    }
    public int getImageResId(){
        return this.imageResId;
    }
}
```

Listing 4 – SensorData.java

Étape 5 : Affichage de la Liste dans l'Activité

Modifiez la classe 'MainActivity.java' :

```
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

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

        RecyclerView recyclerView = findViewById(R.id.
    recyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager
    (this));

        List<SensorData> data = new ArrayList<>();
        data.add(new SensorData("Température", 25.5, "C", R.
    drawable.icdefaultsensor));
        data.add(new SensorData("Pression", 1013.25, "hPa",R.
    drawable.icdefaultsensor));
```

```
        data.add(new SensorData("Humidité", 60.7, "%", R.
    drawable.icdefaultsensor));
        data.add(new SensorData("Tension", 3.3, "V", R.
    drawable.icdefaultsensor));
        data.add(new SensorData("Courant", 1.2, "A", R.
    drawable.icdefaultsensor));

        SensorAdapter adapter = new SensorAdapter(data);
        recyclerView.setAdapter(adapter);
    }
}
```

Listing 5 – MainActivity.java

Explication du code :

— On crée une liste statique d'objets `SensorData` avec l'identifiant de l'image.

— Le reste du code est identique au précédent, à l'exception de l'ajout des images.

Étape 6 : Exécution de l'Application

Exécutez l'application, vous devriez voir une liste de données de capteurs affichée à l'écran avec les images associées.

Partie à Faire par l'Étudiant

Modifiez le layout de chaque élément de la liste pour y ajouter une barre de progression qui correspond à la valeur du capteur. Par exemple, si la valeur de l'humidité est 60.7%, la barre de progression doit indiquer 61%.

Annexe : Code Kotlin Correspondant

SensorData.kt

```
data class SensorData(val name: String, val value: Double,
    val unit: String, val imageResId: Int)
```

Listing 6 – SensorData.kt

SensorAdapter.kt

```kotlin
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

class SensorAdapter(private val data: List<SensorData>) :
    RecyclerView.Adapter<SensorAdapter.ViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup,
    viewType: Int): ViewHolder {
        val view = LayoutInflater.from(parent.context)
            .inflate(R.layout.item\sensor, parent, false)
        return ViewHolder(view)
    }

    override fun onBindViewHolder(holder: ViewHolder,
    position: Int) {
        val item = data[position]
        holder.sensorImageView.setImageResource(item.
    imageResId)
        holder.sensorNameTextView.text = item.name
        holder.sensorValueTextView.text = item.value.toString
    ()
        holder.sensorUnitTextView.text = item.unit
    }

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

    class ViewHolder(itemView: View) : RecyclerView.
    ViewHolder(itemView) {
        val sensorImageView: ImageView = itemView.
    findViewById(R.id.sensorImageView)
        val sensorNameTextView: TextView = itemView.
    findViewById(R.id.sensorNameTextView)
        val sensorValueTextView: TextView = itemView.
    findViewById(R.id.sensorValueTextView)
        val sensorUnitTextView: TextView = itemView.
    findViewById(R.id.sensorUnitTextView)
    }
}
```

Listing 7 – SensorAdapter.kt

MainActivity.kt

```
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activitymain)

        val recyclerView = findViewById<RecyclerView>(R.id.recyclerView)
        recyclerView.layoutManager = LinearLayoutManager(this)
        val data = mutableListOf<SensorData>()
        data.add(SensorData("Température", 25.5, "C", R.drawable.icdefaultsensor))
        data.add(SensorData("Pression", 1013.25, "hPa", R.drawable.icdefaultsensor))
        data.add(SensorData("Humidité", 60.7, "%", R.drawable.icdefaultsensor))
        data.add(SensorData("Tension", 3.3, "V", R.drawable.icdefaultsensor))
        data.add(SensorData("Courant", 1.2, "A", R.drawable.icdefaultsensor))
        val adapter = SensorAdapter(data)
        recyclerView.adapter = adapter
    }
}
```

Listing 8 – MainActivity.kt