Unit 6 : ListView, GridView and RecyclerView

#Contnets

- Listview (Introduction, Features, Implementing listview in an application)
- Gridview (Introduction, Features, Implementing gridview in an application)
- Recyclerview (Introduction, Features, Implementing recyclerview in an application)

#Past Questions

1. How does ListView differ from RecyclerView? Explain with an example. [5, 2020]
2. Develop an Android application to display 8 programming languages in a ListView. [5, 2021]
3. What is the difference between ArrayAdapter and CursorAdapter? Develop an Android application to display the application name with logo (e.g., YouTube, Facebook, Twitter, TikTok, Snapchat, Telegram, etc.) in GridView and display their site in another activity when a GridView item is clicked. [3+7, 2023]
4. Develop an Android application to display a list of students (student_name) in a ListView and display the student details (roll, name, and address) in a fragment when a ListView item is selected. [10, 2021]

1. ListView

Introduction

A **ListView** in Android is a view that displays a vertically scrollable list of items. It's one of the most commonly used components for displaying large sets of data in a list format. Each item in the list is represented by an instance of a view, often based on a predefined XML layout.

Features

- **Scrolling**: Automatically handles vertical scrolling.
- **Customizable Items**: Supports custom item layouts using `ArrayAdapter` or custom adapters.
- **Click Events**: Supports item click and long-click listeners.
- **Optimized for Large Data**: Efficiently manages memory by reusing views (view recycling).

Implementing ListView in an Application

1. XML Layout for ListView:

```
<!-- res/layout/activity_main.xml -->
```

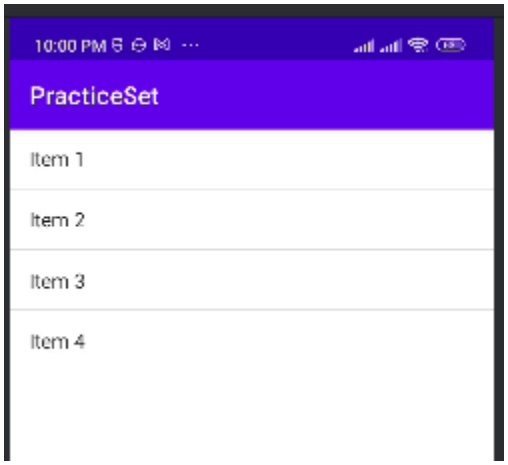
```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

2. Adapter for ListView:

```
MainActivity.java
```

```
package com.example.practiceset;
```

```java
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

        ListView listView = findViewById(R.id.listView);

        String[] values = { "Item 1", "Item 2", "Item 3", "Item 4" };

        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
                android.R.layout.simple_list_item_1, android.R.id.text1, values);

        listView.setAdapter(adapter);
    }
}
```



In this example, an `ArrayAdapter` is used to map an array of strings to the `ListView`. Each item is displayed in a simple text item layout provided by Android.

---

2. GridView

Introduction

A **GridView** is a view that displays items in a two-dimensional, scrollable grid. It's useful when you want to display data in a grid pattern, such as in a photo gallery or a list of products.

Features

- **Grid Layout**: Items are displayed in a grid with a fixed number of columns.
- **Customizable Items**: Supports custom item layouts via adapters.
- **Scrolling**: Automatically handles vertical scrolling.
- **Item Click Handling**: Supports click listeners for individual items.

Implementing GridView in an Application

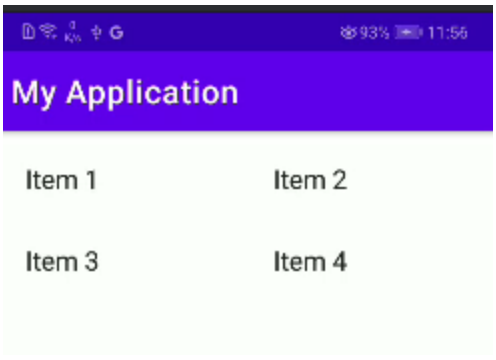
1. XML Layout for GridView:

```
<!-- res/layout/activity_main.xml -->
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <GridView
        android:id="@+id/gridView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:numColumns="2"
        android:padding="10dp" />
</LinearLayout>
```

2. Adapter for GridView:

MainActivity.java

```
package com.example.practiceset;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.GridView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

        GridView gridView = findViewById(R.id.gridView);

        String[] values =  { "Item 1", "Item 2", "Item 3", "Item 4" };

        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
                android.R.layout.simple_list_item_1, android.R.id.text1, values);

        gridView.setAdapter(adapter);
    }
}
```



This example uses an `ArrayAdapter` to map an array of strings to a `GridView`, displaying them in a grid with two columns.

---

3. RecyclerView

Introduction

RecyclerView is a more advanced and flexible version of `ListView` and `GridView`. It is designed to handle large amounts of data efficiently and provides more features, such as item animations and custom layout management. `RecyclerView` is now the recommended way to display large lists or grids of data.

Features

- **ViewHolder Pattern**: Improves performance by reducing the number of `findViewById` calls.
- **Layout Managers**: Supports different layouts like linear, grid, and staggered grid.
- **Item Animations**: Supports custom item animations.
- **Flexibility**: Allows customization of every aspect, from layout to item decoration.
- **Click Handling**: Supports item click listeners and interactions.

Implementing RecyclerView in an Application

1. Adding Dependencies:

Ensure that you have the necessary dependencies in your `build.gradle` file.

```
dependencies {
    implementation 'androidx.recyclerview:recyclerview:1.2.1'
}
```

You should add the dependencies in the `build.gradle.kts` file located inside the `app` directory. This file is responsible for managing the dependencies and configurations specific to the app module

2. XML Layout for RecyclerView:

```
<!-- res/layout/activity_main.xml -->
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

3. Creating a Layout for Items:

```
<!-- res/layout/item_layout.xml -->
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/textViewItem"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp"
    android:textSize="18sp" />
```

4. Creating an Adapter for RecyclerView:

```
// MyAdapter.java
package com.example.myfirstapp;

import android.view.LayoutInflater;
import android.view.View;
```

```
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import java.util.List;

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

    private List<String> dataList;

    public MyAdapter(List<String> dataList) {
        this.dataList = dataList;
    }

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

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        String data = dataList.get(position);
        holder.textViewItem.setText(data);
    }

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

    public static class ViewHolder extends RecyclerView.ViewHolder {
        public TextView textViewItem;

        public ViewHolder(View view) {
            super(view);
            textViewItem = view.findViewById(R.id.textViewItem);
        }
    }
}
```

5. Setting Up RecyclerView in Activity:

```
// MainActivity.java
package com.example.myfirstapp;

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

public class MainActivity extends AppCompatActivity {

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

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

        List<String> dataList = Arrays.asList("Item 1", "Item 2", "Item 3", "Item 4");

        MyAdapter adapter = new MyAdapter(dataList);
        recyclerView.setAdapter(adapter);
    }
}
```

In this example, the `RecyclerView` is set up with a linear layout manager, and a custom adapter is used to populate the items in the list.

Conclusion

- **ListView** is ideal for simple, scrollable lists with minimal customization.
- **GridView** is used when you need to display items in a grid format.
- **RecyclerView** is the most flexible and efficient option for displaying large datasets, offering more customization and better performance. It is now the preferred choice for implementing lists and grids in Android applications.