Name : Mujahid Ullah

Subject: Android Development

Submitted To Sir Junaid

Date : 27-Dec-20

Assignment No 5

List View

List of scrollable items can be displayed in Android using ListView. It helps you to displaying the data in the form of a scrollable list. Users can then select any list item by clicking on it. ListView is default scrollable so we do not need to use scroll View or anything else with ListView.
ListView is widely used in android applications. A very common example of ListView is your phone contact book, where you have a list of your contacts displayed in a ListView and if you click on it then user information is displayed.

Adapter: To fill the data in a ListView we simply use adapters. List items are automatically inserted to a list using an Adapter that pulls the content from a source such as an arraylist, array or database.

ListView in Android Studio: Listview is present inside Containers. From there you can drag and drop on virtual mobile screen to create it. Alternatively you can also XML code to create it.

Here is Android ListView XML Code:

```
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/simpleListView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:context="abhiandroid.com.listexample.MainActivity">
</ListView>
```

Attributes of ListView:

Lets see some different attributes of ListView which will be used while designing a custom list:

1. **id**: id is used to uniquely identify a ListView.
2. **divider**: This is a drawable or color to draw between different list items.
3. **dividerHeight**: This specify the height of the divider between list items. This could be in **dp**(density pixel),sp(scale independent pixel) or px(pixel).

4. **listSelector**: listSelector property is used to set the selector of the listView. It is generally orange or Sky blue color mostly but you can also define your custom color or an image as a list selector as per your design.

```
<!-- List Selector Code in ListView -->

<ListView
android:id="@+id/simpleListView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:divider="#f00"
        android:dividerHeight="1dp"
        android:listSelector="#0f0"/> <!--list selector in green color-->
```

Adapters Use in ListView:

An adapter is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to adapter view then view can takes the data from the adapter view and shows the data on different views like as list view, grid view, spinner etc.

ListView is a subclass of AdapterView and it can be populated by binding to an Adapter, which retrieves the data from an external source and creates a View that represents each data entry.

In android commonly used adapters are:

- **Array Adapter**
- **Base Adapter**

Now we explain these two adapter in detail:

1.Array Adapter: Whenever you have a list of single items which is backed by an array, you can use ArrayAdapter. For instance, list of phone contacts, countries or names.

Important Note: By default, ArrayAdapter expects a Layout with a single TextView, If you want to use more complex views means more customization in list items, please avoid ArrayAdapter and use custom adapters.

Below is Array Adapter code:

```
ArrayAdapter adapter = new
ArrayAdapter<String>(this,R.layout.ListView,R.id.textView,StringArray);
```

Example

Step 1: Create a new project Listexample and activity Main Activity. Here we will create a ListView in LinearLayout. Below is the code of activity_main.xml or content_main.xml:

```
    <?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"
        android:orientation="vertical">

<ListView
android:id="@+id/simpleListView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:divider="@color/material_blue_grey_800"
        android:dividerHeight="1dp" />
</LinearLayout>
```

Step 2: Create a new activity name Listview and below is the code of activity_listview.xml

```
<?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"
    android:orientation="vertical">

<TextView
android:id="@+id/textView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:padding="@dimen/activity_horizontal_margin"
    android:textColor="@color/black" />
</LinearLayout>
```

Step 3: Now in this final step we will use ArrayAdapter to display the country names in UI. Below is the code of MainActivity.java

```
    package abhiandroid.com.listexample;

    import android.os.Bundle;
```

```
    import android.app.Activity;
    import android.view.Menu;
    import android.widget.ArrayAdapter;import android.widget.ListView;

public class MainActivity extends Activity
{
  // Array of strings...
  ListView simpleList;
  String countryList[] = {"India", "China", "australia", "Portugle", "America", "NewZealand"};

  @Override  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);    setContentView(R.layout.activity_main);
    simpleList = (ListView)findViewById(R.id.simpleListView);
    ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this, R.layout.activity_listview,
R.id.textView, countryList);
    simpleList.setAdapter(arrayAdapter);
  }
}
```

2.Base Adapter: BaseAdapter is a common base class of a general implementation of an Adapter that can be used in ListView. Whenever you need a customized list you create your own adapter and extend base adapter in that. Base Adapter can be extended to create a custom Adapter for displaying a custom list item. ArrayAdapter is also an implementation of BaseAdapter.

Example

Step 1: Create a new project Listebasexample and activity Main Activity. Here we will create a ListView in LinearLayout. Below is the code of activity_main.xml or content_main.xml:

```
    <?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"
        android:orientation="vertical">

<ListView
android:id="@+id/simpleListView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:divider="@color/material_blue_grey_800"
        android:dividerHeight="1dp"
        android:footerDividersEnabled="false" />
</LinearLayout>
```

Step 2: Create a new activity name Listview and below is the code of activity_listview.xml

```
    <?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"
        android:orientation="horizontal">

<ImageView
android:id="@+id/icon"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:src="@drawable/ic_launcher" />

<TextView
android:id="@+id/textView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:padding="@dimen/activity_horizontal_margin"
        android:textColor="@color/black" />
</LinearLayout>
```

Step 3: In third step we will use custom adapter to display the country names in UI by coding MainActivity.java. Below is the code of MainActivity.java

```
public class MainActivity extends Activity {

    ListView simpleList;
    String countryList[] = {"India", "China", "australia", "Portugle", "America",
"NewZealand"};
    int flags[] = {R.drawable.india, R.drawable.china, R.drawable.australia,
R.drawable.portugle, R.drawable.america, R.drawable.new_zealand};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        simpleList = (ListView) findViewById(R.id.simpleListView);
        CustomAdapter customAdapter = new CustomAdapter(getApplicationContext(),
countryList, flags);
        simpleList.setAdapter(customAdapter);
    }
}
```

Step 4: Now create another class Custom Adapter which will extend BaseAdapter. Below is the code of CustomAdapter.java

```
public class CustomAdapter extends BaseAdapter {
    Context context;
    String countryList[];
    int flags[];
    LayoutInflater inflter;

    public CustomAdapter(Context applicationContext, String[] countryList, int[]
flags) {
```

```
        this.context = context;
        this.countryList = countryList;
        this.flags = flags;
        inflter = (LayoutInflater.from(applicationContext));
    }

    @Override
    public int getCount() {
        return countryList.length;
    }

    @Override
    public Object getItem(int i) {
        return null;
    }

    @Override
    public long getItemId(int i) {
        return 0;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        view = inflter.inflate(R.layout.activity_listview, null);
        TextView country = (TextView)           view.findViewById(R.id.textView);
        ImageView icon = (ImageView) view.findViewById(R.id.icon);
        country.setText(countryList[i]);
        icon.setImageResource(flags[i]);
        return view;
    }
```

Grid View:

In Android GridView is a view group that display items in two dimensional scrolling grid (rows and columns), the grid items are not necessarily predetermined but they are automatically inserted to the layout using a ListAdapter. Users can then select any grid item by clicking on it. GridView is default scrollable so we don't need to use ScrollView or anything else with GridView.

GridView is widely used in android applications. An example of GridView is your default Gallery, where you have number of images displayed using grid.

Adapter Is Used To Fill Data In Gridview: To fill the data in a GridView we simply use adapter and grid items are automatically inserted to a GridView using an Adapter which pulls the content from a source such as an arraylist, array or database. You can read full Adapter tutorial here.

GridView in Android Studio: Gridview is present inside Containers. From there you can drag and drop on virtual mobile screen to create it. Alternatively you can also XML code to create it.

```
<GridView
android:id="@+id/simpleGridView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:numColumns="3"/>
```

Attributes of GridView:

Lets see different attributes of GridView which will be used while designing a custom grid view:

1. 1.id: id is used to uniquely identify a GridView.
2. 2.numColumns: numColumn define how many columns to show. It may be a integer value, such as “5” or auto_fit.

auto_fit is used to display as many columns as possible to fill the available space on the screen.

```
<!-- numColumns example code -->
<GridView
android:id="@+id/simpleGridView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:numColumns="4"/> <!-- numColumns set to 4-->
```

3. horizontalSpacing: horizontalSpacing property is used to define the default horizontal spacing between columns. This could be in pixel(px),density pixel(dp) or scale independent pixel(sp).

Below is the horizontalSpacing example code with explanation included where horizontal spacing between grid items is 50 dp.

```
<!--Horizontal space example code in grid view-->>
<GridView
android:id="@+id/simpleGridView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:numColumns="3"
        android:horizontalSpacing="50dp"/><!--50dp horizontal space between grid
items-->
```

4. verticalSpacing: verticalSpacing property used to define the default vertical spacing between rows. This should be in px, dp or sp.

Below is the verticalSpacing example code with explanation included, where vertical spacing between grid items is 50dp.

```
<!-- verticalSpacing example code in grid view-->>
<GridView
android:id="@+id/simpleGridView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:numColumns="3"
        android: verticalSpacing ="50dp"/><!--50dp verticalSpacing between grid items-->
```

5. **columnWidth:** columnWidth property specifies the fixed width of each column. This could be in px, dp or sp.

Below is the columnWidth example code. Here column width is 80dp and selected item’s background color is green which shows the actual width of a grid item.

```
<!-- columnWidth example code in grid view-->>
<GridView
android:id="@+id/simpleGridView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:numColumns="3"
        android: columnWidth ="80dp"
        android: verticalSpacing ="50dp"/><!--50dp columnWidth between grid items-->
```

Example of GridView using Custom Adapter :

- Step 1: Create a new project and name it GridViewExample.
- Step 2: Now open app -> res -> layout -> activity_main.xml (or) main.xml and add following code :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/activity_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        tools:context="abhiandroid.com.gridviewexample.MainActivity">

<GridView
        android:id="@+id/simpleGridView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:numColumns="3"/>

</RelativeLayout>
```

- **Step 3:** Create a new layout Activity in app -> res-> layout-> new activity and name it grid_view_items.xml and add following code:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/grid_view_items"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        tools:context="abhiandroid.com.gridviewexample.GridViewItems">

<ImageView
```

```
        android:id="@+id/imageView"
                android:layout_width="150dp"
                android:layout_height="150dp"
                android:padding="5dp"
                android:scaleType="fitXY"
                android:src="@drawable/ic_launcher"
                android:layout_alignParentTop="true"
                android:layout_centerHorizontal="true" />

    <TextView
            android:id="@+id/textView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="@dimen/activity_horizontal_margin"
                android:text="Demo"
                android:textColor="#000"
                android:layout_below="@+id/imageView"
                android:layout_centerHorizontal="true"
                android:layout_marginTop="13dp" />
    </RelativeLayout>
```

- **Step 4:** Now open app -> java -> package -> MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    GridView simpleList;
    ArrayList birdList=new ArrayList<>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        simpleList = (GridView) findViewById(R.id.simpleGridView);
        birdList.add(new Item("Bird 1",R.drawable.b1));
        birdList.add(new Item("Bird 2",R.drawable.b2));
        birdList.add(new Item("Bird 3",R.drawable.b3));
        birdList.add(new Item("Bird 4",R.drawable.b4));
        birdList.add(new Item("Bird 5",R.drawable.b5));
        birdList.add(new Item("Bird 6",R.drawable.b6));

        MyAdapter myAdapter=new MyAdapter(this,R.layout.grid_view_items,birdList);
        simpleList.setAdapter(myAdapter);

    }
}
```

- **Step5 :** Create a new Class src -> package -> MyAdapter.java and add the following code:

```
public class MyAdapter extends ArrayAdapter {

    ArrayList birdList = new ArrayList<>();

    public MyAdapter(Context context, int textViewResourceId, ArrayList objects) {
        super(context, textViewResourceId, objects);
        birdList = objects;
    }

    @Override
    public int getCount() {
        return super.getCount();
```

```
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        View v = convertView;
        LayoutInflater inflater = (LayoutInflater)
getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        v = inflater.inflate(R.layout.grid_view_items, null);
        TextView textView = (TextView) v.findViewById(R.id.textView);
        ImageView imageView = (ImageView) v.findViewById(R.id.imageView);
        textView.setText(birdList.get(position).getbirdName());
        imageView.setImageResource(birdList.get(position).getbirdImage());
        return v;

    }

}
```

- **Step 6:** Create a new Class src -> package -> Item.java and add the below code:

```
public class Item {

    String birdListName;
    int birdListImage;

    public Item(String birdName,int birdImage)
    {
        this.birdListImage=birdImage;
        this.birdListName=birdName;
    }
    public String getbirdName()
    {
        return birdListName;
    }
    public int getbirdImage()
    {
        return birdListImage;
    }
}
```