Simple Android grid example using RecyclerView with GridLayoutManager (like the old GridView)

Asked 5 years, 4 months ago Modified 1 month ago Viewed 342k times

290

98

I know that `RecyclerView` has replaced the functionality of the old `ListView` and `GridView`. I am looking for a very basic example that shows a minimal grid setup using `RecyclerView`. I am not looking for long tutorial style explanations, just a minimal example. I imagine the simplest grid that mimics the old GridView would consist of the following features:

- multiple cells per row
- single view in each cell
- responds to click events

android gridview android-recyclerview gridlayoutmanager

Share Improve this question Follow

edited Jan 31, 2017 at 15:19

asked Nov 14, 2016 at 11:02

Suragch
419k 273 1269 1305

8 Answers

Sorted by: Highest score (default)

705

Short answer

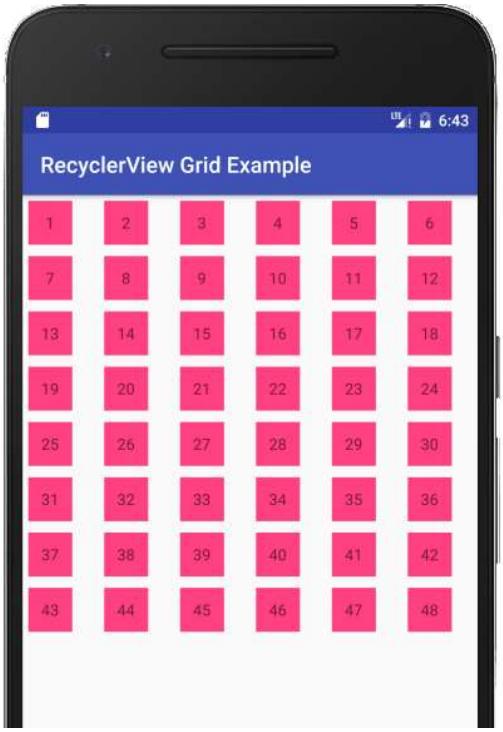
For those who are already familiar with [setting up a `RecyclerView` to make a list], the good news is that making a grid is largely the same. You just use a `GridLayoutManager` instead of a `LinearLayoutManager` when you set the `RecyclerView` up.

```
recyclerView.setLayoutManager(new GridLayoutManager(this, numberOfColumns));
```

If you need more help than that, then check out the following example.

Full example

The following is a minimal example that will look like the image below.



Start with an empty activity. You will perform the following tasks to add the `RecyclerView` grid. All you need to do is copy and paste the code in each section. Later you can customize it to fit your needs.

- Add dependencies to gradle
- Add the xml layout files for the activity and for the grid cell
- Make the RecyclerView adapter
- Initialize the RecyclerView in your activity

Update Gradle dependencies

Make sure the following dependencies are in your app `gradle.build` file:

```
compile 'com.android.support:appcompat-v7:27.1.1'
compile 'com.android.support:recyclerview-v7:27.1.1'
```

You can update the version numbers to whatever is the most current.

Create activity layout

Add the `RecyclerView` to your xml layout.

activity_main.xml

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

```
    <android.support.v7.widget.RecyclerView
        android:id="@+id/rvNumbers"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</RelativeLayout>
```

Create grid cell layout

Each cell in our `RecyclerView` grid is only going to have a single `TextView`. Create a new layout resource file.

recyclerview_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:padding="5dp"
    android:layout_width="50dp"
    android:layout_height="50dp">

        <TextView
            android:id="@+id/info_text"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center"
            android:background="@color/colorAccent"/>

</LinearLayout>
```

Create the adapter

The `RecyclerView` needs an adapter to populate the views in each cell with your data. Create a new java file.

MyRecyclerViewAdapter.java

```
public class MyRecyclerViewAdapter extends
RecyclerView.Adapter<MyRecyclerViewAdapter.ViewHolder> {

    private String[] mData;
    private LayoutInflater mInflater;
    private ItemClickListener mClickListener;

    // data is passed into the constructor
    MyRecyclerViewAdapter(Context context, String[] data) {
        this.mInflater = LayoutInflater.from(context);
        this.mData = data;
    }

    // inflates the cell layout from xml when needed
    @Override
    @NonNull
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
```

```
        View view = mInflater.inflate(R.layout.recyclerview_item, parent, false);
        return new ViewHolder(view);
    }

    // binds the data to the TextView in each cell
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        holder.myTextView.setText(mData[position]);
    }

    // total number of cells
    @Override
    public int getItemCount() {
        return mData.length;
    }


    // stores and recycles views as they are scrolled off screen
    public class ViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {
        TextView myTextView;

        ViewHolder(View itemView) {
            super(itemView);
            myTextView = itemView.findViewById(R.id.info_text);
            itemView.setOnClickListener(this);
        }

        @Override
        public void onClick(View view) {
            if (mClickListener != null) mClickListener.onItemClick(view,
getAdapterPosition());
        }
    }

    // convenience method for getting data at click position
    String getItem(int id) {
        return mData[id];
    }

    // allows clicks events to be caught
    void setClickListener(ItemClickListener itemClickListener) {
        this.mClickListener = itemClickListener;
    }

    // parent activity will implement this method to respond to click events
    public interface ItemClickListener {
        void onItemClick(View view, int position);
    }
}
```

Notes

- Although not strictly necessary, I included the functionality for listening for click events on the cells. This was available in the old `GridView` and is a common need. You can remove this code if you don't need it.

Initialize RecyclerView in Activity

Add the following code to your main activity.

MainActivity.java

```
public class MainActivity extends AppCompatActivity implements
MyRecyclerViewAdapter.ItemClickListener {

    MyRecyclerViewAdapter adapter;

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

        // data to populate the RecyclerView with
        String[] data = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12",
"13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26",
"27", "28", "29", "30", "31", "32", "33", "34", "35", "36", "37", "38", "39", "40",
"41", "42", "43", "44", "45", "46", "47", "48"};

        // set up the RecyclerView
        RecyclerView recyclerView = findViewById(R.id.rvNumbers);
        int numberOfColumns = 6;
        recyclerView.setLayoutManager(new GridLayoutManager(this, numberOfColumns));
        adapter = new MyRecyclerViewAdapter(this, data);
        adapter.setClickListener(this);
        recyclerView.setAdapter(adapter);
    }

    @Override
    public void onItemClick(View view, int position) {
        Log.i("TAG", "You clicked number " + adapter.getItem(position) + ", which is at
cell position " + position);
    }
}
```

Notes

- Notice that the activity implements the `ItemClickListener` that we defined in our adapter. This allows us to handle cell click events in `onItemClick`.

Finished

That's it. You should be able to run your project now and get something similar to the image at the top.

Going on

Rounded corners

- Use a CardView

Auto-fitting columns

- GridLayoutManager - how to auto fit columns?

Further study

- Android RecyclerView with GridView GridLayoutManager example tutorial
- Android RecyclerView Grid Layout Example
- Learn RecyclerView With an Example in Android
- RecyclerView: Grid with header
- Android GridLayoutManager with RecyclerView in Material Design
- Getting Started With RecyclerView and CardView on Android

Share Improve this answer Follow edited Aug 25, 2018 at 7:56 answered Nov 14, 2016 at 11:02

Suragch
419k 273 1269 1305

2 @MarianPaździoch, Yes, I just made this as a minimal example. It could definitely use some beautification work. I'll try to update this answer some time in the future. – Suragch Aug 29, 2017 at 1:10

1 I logged in just to point that people like you have kept this portal alive and kicking .i was stuck on this for two days before seeing this solution. thanks a lot – VarunJoshi129 Nov 25, 2017 at 6:44

1 @androiddeveloper, The grid items get laid out left to right, top to bottom. Scrolling is vertical when there are more items than can fit on the screen. – Suragch Dec 5, 2017 at 11:31

19 Future readers, let me save you some time, key thing is `recyclerView.setLayoutManager(new GridLayoutManager(this, numberOfColumns));` – daka Jun 21, 2018 at 20:50

1 @daka, good point. I edited my answer to include this in the beginning. – Suragch Jun 21, 2018 at 23:27

11

Although I do like and appreciate *Suragch*'s answer, I would like to leave a note because I found that coding the *Adapter* (`MyRecyclerViewAdapter`) to define and expose the Listener method `onItemClick` isn't the best way to do it, due to not using class encapsulation correctly. So my suggestion is to let the *Adapter* handle the Listening operations solely (that's his purpose!) and separate those from the Activity that uses the *Adapter* (`MainActivity`). So this is how I would set the Adapter class:

MyRecyclerViewAdapter.java

```
public class MyRecyclerViewAdapter extends
RecyclerView.Adapter<MyRecyclerViewAdapter.ViewHolder> {

    private String[] mData = new String[0];
    private LayoutInflater mInflater;

    // Data is passed into the constructor
    public MyRecyclerViewAdapter(Context context, String[] data) {
        this.mInflater = LayoutInflater.from(context);
        this.mData = data;
    }
```

```
    // Inflates the cell layout from xml when needed
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = mInflater.inflate(R.layout.recyclerview_item, parent, false);
        ViewHolder viewHolder = new ViewHolder(view);
        return viewHolder;
    }

    // Binds the data to the textview in each cell
    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        String animal = mData[position];
        holder.myTextView.setText(animal);
    }

    // Total number of cells
    @Override
    public int getItemCount() {
        return mData.length;
    }

    // Stores and recycles views as they are scrolled off screen
    public class ViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {
        public TextView myTextView;

        public ViewHolder(View itemView) {
            super(itemView);
            myTextView = (TextView) itemView.findViewById(R.id.info_text);
            itemView.setOnClickListener(this);
        }

        @Override
        public void onClick(View view) {
            onItemClick(view, getAdapterPosition());
        }
    }

    // Convenience method for getting data at click position
    public String getItem(int id) {
        return mData[id];
    }

    // Method that executes your code for the action received
    public void onItemClick(View view, int position) {
        Log.i("TAG", "You clicked number " + getItem(position).toString() + ", which is
at cell position " + position);
    }
}
```

Please note the `onItemClick` method now defined in `MyRecyclerViewAdapter` that is the place where you would want to code your tasks for the event/action received.

There is only a small change to be done in order to complete this transformation: the *Activity* doesn't need to implement `MyRecyclerViewAdapter.ItemClickListener` anymore, because now that is done completely by the *Adapter*. This would then be the final modification:

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
```

```
    MyRecyclerViewAdapter adapter;

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

        // data to populate the RecyclerView with
        String[] data = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12",
"13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26",
"27", "28", "29", "30", "31", "32", "33", "34", "35", "36", "37", "38", "39", "40",
"41", "42", "43", "44", "45", "46", "47", "48"};

        // set up the RecyclerView
        RecyclerView recyclerView = (RecyclerView) findViewById(R.id.rvNumbers);
        int numberOfColumns = 6;
        recyclerView.setLayoutManager(new GridLayoutManager(this, numberOfColumns));
        adapter = new MyRecyclerViewAdapter(this, data);
        adapter.setClickListener(this);
        recyclerView.setAdapter(adapter);
    }
}
```

Share Improve this answer Follow

answered Aug 21, 2017 at 0:23

jonypera
412 1 12 21

3 What if the activity does need to listen to the click events? e.g. passing data to presenter, doing some logic based on item clicked, tracking, etc. – Ahmad Fadli Mar 27, 2018 at 7:18

I agree that Adapter should handle click events, since it has the items with the data in it. @AhmadFadli if you need to do some work in the adapter's host (a Fragment or Activity) you should create a callback interface with methods you need. Your host implements this interface. And then you pass an instance of your host into Adapter's constructor. Having the instance of the host you can call it's methods when you need from your Adapter. And your host we get callbacks called. This is often used when you need to work with ActionMode. When you longClick to select items and use ActionBar buttons.
– Kirill Karmazin Feb 19, 2019 at 12:32

I disagree and think that click events should be processed in the hosting `Activity`. Because only it's click listener can know about the `Activity` views and other `Fragments`, `Activities`, etc. The adapter can only send click events to upper level. It should have the interface `ItemClickListener` with so many events, as many events adapter's views can produce. This solution was written even earlier: stackoverflow.com/a/40563598/2914140. – CoolMind Nov 22, 2019 at 15:05

10

You should set your `RecyclerView LayoutManager` to Gridlayout mode. Just change your code when you want to set your `RecyclerView LayoutManager`:

```
recyclerView.setLayoutManager(new GridLayoutManager(getActivity(), numberOfColumns));
```

Share Improve this answer Follow

edited Sep 1, 2020 at 8:02

CoolMind
21.8k 12 165 194

answered Nov 7, 2019 at 20:29

MR Coder
101 1 3

7

This is a simple way from XML only

spanCount for number of columns

layoutManager for making it grid or linear(Vertical or Horizontal)

```
<androidx.recyclerview.widget.RecyclerView
        android:id="@+id/personListRecyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layoutManager="androidx.recyclerview.widget.GridLayoutManager"
        app:spanCount="2"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Share Improve this answer Follow

answered Oct 14, 2020 at 14:07

Hamdy Abd El Fattah
670 1 8 10

3

Set in `RecyclerView` initialization

```
recyclerView.setLayoutManager(new GridLayoutManager(this, 4));
```

Share Improve this answer Follow

edited Aug 1, 2020 at 6:10

Dima Kozhevin
3,371 9 36 51

answered Aug 1, 2020 at 4:50

user14031748

2 How does it differ from other answers? – CoolMind Sep 1, 2020 at 8:02

3

There are 2 ways to achieve this

- In Xml

```
 <androidx.recyclerview.widget.RecyclerView
                        android:id="@+id/list_amenities"
                        android:layout_width="0dp"
                        android:layout_height="wrap_content"
                        android:layout_marginTop="@dimen/_8sdp"
                        android:nestedScrollingEnabled="false"
 app:layoutManager="androidx.recyclerview.widget.GridLayoutManager"
                        app:spanCount="2"
app:layout_constraintEnd_toEndOf="@+id/text_parking_lot_amenities"
app:layout_constraintStart_toStartOf="@+id/text_parking_lot_amenities"
app:layout_constraintTop_toBottomOf="@id/text_parking_lot_amenities" />
```

span count is used for grid columns

- In activity

```
listAmenities.layoutManager = GridLayoutManager(this, TWO)
here TWO indicates number of grid columns
```

Share Improve this answer Follow

answered Feb 9 at 11:03

Nandini

31 2

0

in your MainActivity, where u assigned your recycler view, just use this code.

```
recyclerView = findViewById(R.id.recycler_view);
    recyclerView.setHasFixedSize(true);
    //recyclerView.setLayoutManager(new LinearLayoutManager(this));
    recyclerView.setLayoutManager(new GridLayoutManager(this, 2));
```

Share Improve this answer Follow

answered Nov 20, 2021 at 10:04

A.I.Shakil

334 2 10

0

if you want to set grid layout in reyclerview in xml file then you can put these two in in recyclerview xml.

```
app:layoutManager="androidx.recyclerview.widget.GridLayoutManager"
app:spanCount=numberOfItemsInSignleRow
```

if you want to set grid layout from java code you can write this.

```
recyclerView.setLayoutManager(new GridLayoutManager(getActivity(),
numberOfItemsInSignleRow));
```

Share Improve this answer Follow

answered Feb 2 at 13:22

Hammad Zafar Bawara

1 1