Android recyclerview with grid layout manager

I'm not robot

reCAPTCHA

Continue

In Android, RecyclerView is an advanced and flexible version of ListView and GridView. It is a container that is used to display large amounts of data sets that can be scrolled very efficiently by maintaining a limited number of views. RecyclerView was introduced in Material Design in API level 21 (Android 5.0 ie Lollipop). Material Design brings many new features in Android that changed a lot of the visual design patterns regarding the design of modern Android applications. This new widget is a great step to display data because GridView is one of the most commonly used UI widget. In RecyclerView android provides a lot of new features that are not present in existing ListView or GridView. XML code for basic recovery six files: &lt;?xml version=1.0 encoding=utf-8?&gt;&lt;RelativeLayout xmlns:android= xmlns:tools= android:layout_width=match_parent android:layout_height==match_parent tools:context=abhiandroid.com.recyclerviewexample.MainActivity&gt; &lt;android.support.v7.widget.RecyclerView android:id=@+id/recyclerView android:layout_width=match_parent android:layout_height=match_parent&gt;&lt;/android.support.v7.widget.RecyclerView&gt; &lt;/RelativeLayout&gt; Gradle Dependency to use RecyclerView: The RecyclerView widget is part of separate library valid for API 7 level or higher. Add the following dependency to the gradle build file to use the recycling robe. dependencies { ... compile com.android.support:recyclerview-v7:23.23.4.0 } RecyclerView As GridView In Android: In this article we will discuss how to use a RecyclerView As GridView. For that we need to understand the LayoutManager component of RecyclerView. Layout Manager is a very new concept introduced in RecyclerView to define the type of layout that RecyclerView needs. It contains the references for all views filled in by the data in the record. We can create a Custom Layout Manager by expanding RecyclerView.LayoutManager Class, but RecyclerView includes three types of built-in layout managers. 1. LinearLayoutManager: It is used to display vertical or horizontal list. To understand this, read RecyclerView as Listview 2. GridLayoutManager: It is used to display the items in the form of grids. 3. StaggeredGridLayoutManager: It is used to display the items in the offset grid. In this article, our primary focus is on GridLayoutManager because it is used to display data in the form of grids. By using this Layout Manager, we can easily create grid items. A common example of grid elements is our phone's gallery, where all the images are displayed in the form of grids. GridLayoutManager is used to display the data elements in grid format, and we can easily define the orientation of the items. In simple words, we can say that we use to display RecyclerView as a GridView. GridLayoutManager public constructors: Below we define Builder for GridLayoutManager to use to define the direction (vertical or horizontal) of RecyclerView. 1- GridLayoutManager: It is used to create a vertical grid layoutManager. In this constructor first parameter is used to set the current context and the second parameter is used to set the span number Of 2000 means the number of columns in the grid. Example: In the snippet below, we'll show you how to use this constructor in Android. With standard vertical orientation: // get reference to RecyclerView RecyclerView = (RecyclerView) findViewById(R.id.recyclerView); Specify a GridLayoutManager with default vertical orientation and 3 number of columns GridLayoutManager gridLayoutManager = new GridLayoutManager(getApplicationContext(),3);

recyclerView.setLayoutManager(gridLayoutManager); set LayoutManager for RecyclerView horizontal orientation: // get the reference to RecyclerView RecyclerView = (RecyclerView) findViewById(R.id.recyclerView); Specify a GridLayoutManager with 3-number columns GridLayoutManager gridLayoutManager = new GridLayoutManager(getApplicationContext(),3); gridLayoutManager.setOrientation(LinearLayoutManager.HORIZONTAL); specify horizontal orientation recyclerView.setLayoutManager(gridLayoutManager); Set LayoutManager for RecyclerView 2- GridLayoutManager int spanCount, int orientation, boolean reverseLayout): In this constructor first parameter is used to set the current context, the second parameter is used to set the span00Ore that the number of columns in the grid, third parameter is used to set the layout direction must be vertical or horizontal, and last param is a Boolean value when sets to true layout from end to beginning mean that grids are arranged from start to start. Example: In the snippet below, we'll show you how to use this constructor in Android. for reference to RecyclerView RecyclerView = (RecyclerView) findViewById(R.id.recyclerView); for example, you can set a GridLayoutManager with 3 columns, horizontal gravity, and false value for reverseLayout to display the items from startup to GridLayoutManager gridLayoutManager = new GridLayoutManager(getApplicationContext(),3,LinearLayoutManager.HORIZONTAL,false); recyclerView.setLayoutManager(gridLayoutManager); For example, you can set LayoutManager for RecyclerView Comparison between RecyclerView and GridView There are a lot of new features in RecyclerView that are not present in existing GridView. The RecyclerView is more flexible, powerful and a major improvement over GridView. I'll try to give you a detailed insight into that. Below, we discuss some important features of RecyclerView that should clear the reason why RecyclerView is better than GridView. 1. Custom item layout: GridView can only layout in Vertical Arrangement, where we set the number of columns and rows are is according to the number of items. GridView cannot be customized according to our requirements. Suppose that we have to display elements in horizontal arrangement where we want to set the number of rows and columns are automatically creates according to the number of items, but that things are not possible with standard GridView. However, with the introduction of Recyclerview, we can easily create a horizontal or vertical arrangement for grid elements. Using the GridLayoutManager component in RecyclerView we can easily define the direction of grid elements and spanCount is used for number of columns if the orientation is vertical or the number of rows if the orientation is horizontal. 2. Using View Holder Pattern: GridView adapters do not require the use of ViewHolder, but RecyclerView requires the use of the ViewHolder used to save the view's reference. In GridView, it is recommended to use ViewHolder, but it is not coercive, but in RecyclerView it is mandatory to use ViewHolder, which is the main difference between RecyclerView and GridView. ViewHolder is a static inner class in our adapter that contains references to the relevant views. By using these references our code can avoid time consuming findViewById() method to update widgets with new data. 3. GridView Adapters: In GridView we use many Adapter's like ArrayAdapter to display simple array data, Base and SimpleAdapters for custom grids with images and text. In RecyclerView, we only use RecyclerView.Adapter to specify the grid elements. In the following snippet, we show what our CustomAdapter looks like when we expand the RecyclerView.Adapter class and use The ViewHolder in it. package abhiandroid.com.recyclerviewexample; import android.support.v7.widget.RecyclerView; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.widget.TextView; public class CustomAdapter expands RecyclerView.Adapter { @Override public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) { // infalte element LayoutView v = LayoutInflater.from(parent.getContext()).inflate(R.layout.rowlayout, parent), false); // specify display size, margins, myViewHolder vh = new MyViewHolder(v); // pass the view to Show Holder Return Vh; } @Override publicly invalid onBindViewHolder(MyViewHolder holder, int position) { } @Override public int getItemCount(() { return 0; } public class MyViewHolder expands RecyclerView.ViewHolder { TextView textView;// init item view public MyViewHolder(View itemView) { super(itemView); // get the reference to the item view's textView = (TextView) itemView.findViewById(R.id.textView); } } 4. Item Animator: GridView lacks support for Good Animation's. RecyclerView brings a new dimension to it. By using class we can easily animate the view. 5. Item Decoration: In GridView's Dynamic Decorating Items like Adding Adding or edge was never easy, but in RecyclerView using recycleview.ItemDecorator class, we have a huge control on it. Conclusion: At the end we can say that RecyclerView is much more customizable than the existing GridView and gives a lot of control and power to its developers. Example of RecyclerView As Vertical GridView In Android Studio: Below is the example of RecyclerView As GridView, where we show grids of Person Names with their images with standard vertical orientation using RecyclerView. In this example, we use GridLayoutManager with vertical orientation and 2 tensing aging value to display the items. First, we declare a RecyclerView in our XML file and then get the reference to it in our activity. After that we create two ArrayList's for Person Names and Pictures. Then we set up a GridLayoutManager and finally we set the adapter to display the grid elements in RecyclerView. When a user clicks an item, the full-size image appears on the next screen. Download Code Step 1: Create a new project and name the RecyclerViewExample. Step 2: Open Gradle Scripts &gt; build.gradle and add dependency on the Recycle Vision Library in it. anvende plugin: 'com.android.application' android { compileSdkVersion 24 buildToolsVersion 24.0.1 defaultConfig { applicationId abhiandroid.com.recyclerviewexample minSdkVersion 16 targetSdkVersion 024 versionCode 1 versionName 1.0 } buildTypes { release { minifyEnabled false proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro' } } afhængigheder { kompiler filTræ(dir: 'libs', omfatter: ['*.jar']) testCompile 'junit:junit:4.12' kompilere 'com.android.support:appcompat-v7:24.1.1' kompilere com.android.support:recyclerView-v7:23.4.0 // afhængighedsfil til RecyclerView } Trin 3: Åbn res -&gt; layout -&gt; activity_main.xml (eller) main.xml, og tilføj følgende kode: I dette trin opretter vi en Genvindingsvisning i vores XML-fil. &lt;?xml version=1.0 encoding=utf-8?&gt;&lt;RelativeLayout xmlns:android= xmlns:tools= android:layout_width=match_parent android:layout_height=match_parent tools:context=abhiandroid.com.recyclerviewexample.MainActivity&gt; &lt;android.support.v7.widget.RecyclerView android:id=@+id/recyclerView android:layout_width=match_parent android :layout_height=match_parent&gt;&lt;/android.support.v7.widget.RecyclerView&gt; &lt;/RelativeLayout&gt; Step 4: Create a new drawable XML file in the Drawable folder, and name gr. custom_item_layout.xml and add the following code to it to create a custom grid element. In this step, we create a drawable XML file where we add the code for creating custom grid elements.&lt;?xml version=1.0 encoding=utf-8?&gt;Step 5: Create a new XML file &lt;shape &lt;!-- stroke with color and width for creating outer line --&gt; &lt;stroke android:width=1dp android:color=#000&gt;&lt;/stroke&gt; &lt;/shape&gt; &lt;/shape&gt; for the recyclerview grid element, and insert the following code into it. In this step, we create a new xml file for the item row, where we create a TextView and ImageView to display the data in grid format. &lt;?xml version=1.0 encoding=utf-8?&gt;Step 6: Now open app -&gt; java-&gt; package -&gt; MainActivity.java and add the code below. &lt;RelativeLayout xmlns:android= layout_width=match_parent android:layout_height=120dp android background:=@drawable/custom_item_layout android:padding=5dp&gt; &lt;!-- grid items for RecyclerView --&gt; &lt;ImageView android:id=@+id/image android:layout_width=match_parent android:layout_height=wrap_content android:scaletype=fitXY android:src=@mipmap/ic_launcher&gt;&lt;/ImageView&gt; &lt;TextView android:id=@+id/name android:layout_width=wrap_content android:layout_height=wrap_content android:layout_alignparentbottom=true android:layout_centerhorizontal=true android:text=ABCD android:textsize=20sp android:textcolor=#fff&gt;&lt;/TextView&gt; &lt;/RelativeLayout&gt; In this step for the first step, we get reference to RecyclerView. After that we create two ArrayList's for Person Names and Pictures. Then we set up a GridLayoutManager and finally we set the adapter to display the grid elements in RecyclerView. package abhiandroid.com.recyclerviewexample; import android.os.Bundle; import android.support.v7.app.AppCompatActivity; import android.support.v7.widget.GridLayoutManager; import android.support.v7.widget.RecyclerView; importing java.util.ArrayList; importing java.util.Arrays; public class MainActivity expands AppCompatActivity { // ArrayList for Personal Names ArrayList personNames = new ArrayList &lt;&gt; (Arrays.asList(Person 1, Person 2, Person 3, Person 4, Person 5, Person 6, Person 7,Person 8, Person 9, Person 10, Person 11, Person 12, Person 13, Person 14)); ArrayList personImages = new ArrayList &lt;&gt; (Arrays.asList(R.drawable.person1, R.drawable.person2, R.drawable.person3, R.drawable.person4, R.drawable.person5, R.drawable.person6, R.drawable.person7,R.drawable.person1, R.drawable.person2, R.drawable.person3, R.drawable.person4, R.drawable.person5, R.drawable.person6, R.drawable.person); @Override protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity_main // get the reference to RecyclerView RecyclerView recyclerView = ((rView) findViewById(R.id.recyclerView); // set a GridLayoutManager with standard vertical orientation and 2 number of columns GridLayoutManagerLayout gridManager = new GridLayoutManager(getApplicationContext(),2); recyclerView.setLayoutManManager(gridLayoutManager); // set LayoutManager to RecyclerView // call the Custom Adapter constructor to send the reference and data to adapter CustomAdapter customAdapter = new personNames,personImages); personNames,personImages); Set the adapter to RecyclerView } Step 7: Create a new class CustomAdapter.java inside of the package and add the following code. In this step, we create a CustomAdapter class and expand the RecyclerView.Adapter with the viewholder in it. We then implement the parent methods and create a constructor to get the data from Activity. In this custom Adapter two methods are more important first is onCreateViewHolder, where we inflate the layout element xml and pass it to view holder and others are onBindViewHolder, where we set data in the view with the help of ViewHolder. Finally, we implement the setOnClickListener event on itemview and at the click of item we show the selected image in full size in the next activity. package abhiandroid.com.recyclerviewexample; import android.content.Context; import android.content.Intent; import android.support.v7.widget.RecyclerView; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.widget.ImageView; import android.widget.TextView; importing java.util.ArrayList; public class CustomAdapter expands RecyclerView.Adapter { ArrayList personNames; MatrixList personImages; Context context context context; public CustomAdapter(Context context, ArrayList personNames, ArrayList personImages) { this.context = context; this.personNames = personNames; this.personImages = personImages; } @Override public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) { // infalte the item Layout View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.rowlayout, parent, false); // set the view's size, margins, paddings and layout parameters MyViewHolder vh = new MyViewHolder(v); // pass the view to View Holder return vh; } @Override public void onBindViewHolder(MyViewHolder holder, final int position) { // set the data in items holder.name.setText(personNames.get(position)); holder.image.setImageResource(personImages.get(position)); // implement setOnClickListener event on item view. holder.itemView.setOnClickListener(new View.OnClickListener() { @Override public void onClick(View view) { // open another activity on item click Intent intent = new Intent(context, SecondActivity.class); intent.putExtra(image, personImages.get(position)); // put image data in Intent context.startActivity(intent); // start Intent } }); } @Override public int getItemCount() { return personNames.size(); } public class MyViewHolder extends RecyclerView.ViewHolder { // init the item view's TextView name; Image of ImageView; public MyViewHolder(View itemView) { super(itemView); // get the reference to the item view name = (TextView) itemView.findViewById(R.id.name); image = (ImageView) itemView.findViewById(R.id.image); } } Step 8: Create a new XML file activity_second.xml and add the code below to it. In this step, we create a in our XML file to display the selected full-size image. Size. version=1.0 encoding=utf-8?&gt; &lt;RelativeLayout xmlns:android= xmlns:tools= android:layout_width=match_parent android:layout_height==match_parent android:paddingingbottom=@dimen/activity_vertical_margin android:paddingleft=@dimen/activity_horizontal_margin android:paddingright=@dimen/activity_horizontal_margin android:paddingtop=@dimen/activity_vertical_margin android:background=#fff&gt; &lt;ImageView android:id=@+id/selectedImage android:layout_width=match_parent android:layout_height=wrap_content android:layout_centerinparent=true android:scaletype=fitXY&gt;&lt;/ImageView&gt; &lt;/RelativeLayout&gt; Step 9: Create a new activity and name the SecondActivity.class and add the code below to it. In this step we get the reference to ImageView and then get Intent, which was set from the adapter of previous activity and then finally we set the image in ImageView. package abhiandroid.com.recyclerviewexample; import android.content.Intent; import android.os.Bundle; import android.support.v7.app.AppCompatActivity; import android.widget.ImageView; public class SecondActivity expands AppCompatActivity { ImageView selectedImage; @Override protected void onCreate(Bundle savedInstanceState) {super.onCreate(savedInstanceState); setContentView(R.layout.activity_second); selectedImage = (ImageView) findViewById(R.id.selectedImage); // init an ImageView Intent intent = getIntent(); // get Intent which was set from Previous Activity Image selectedImage.setImageResource(intent.getIntExtra 0)); // get image from Intent and put it in ImageView } } Output: Now run App, and you will see person name which can be rolled in vertical direction created using RecyclerView as Gridview. Example of RecyclerView As horizontal GridView In Android Studio below is the example of RecyclerView As GridView, where we show grids of Person Names with their images with horizontal orientation using RecyclerView. In this example, we use GridLayoutManager with horizontal orientation and 3 tensal rims that define the number of rows. First, we declare a RecyclerView in our XML file and then get the reference to it in our activity. After that we create two ArrayList's for Person Names and Pictures. Then we set up a GridLayoutManager and finally we set the adapter to display the grid elements in RecyclerView. When a user clicks an item, the full-size image appears on the next screen. Download code step 1: Create a new project and name it RecyclerViewExample. Step 2: Open Gradle Scripts &gt; build.gradle and add dependency on the Recycle Vision Library in it. apply plugin: 'com.android.application' android { compileSdkVersion 24 buildToolsVersion 24.0.1 defaultConfig { applicationId abhiandroid.com.recyclerviewexample mySdkVersion 16 24 versionCode 1 versionName 1.0 1.0 buildTypes { release { minifyEnabled false proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro' } } dependencies { compile fileTree(dir: 'libs', omfatter: ['*.jar']) testCompile 'junit:junit:4.12' kompilere 'com.android.support:support:com.android.support:support:appcompat-v7:24.1.1' kompilere com.android.support:recyclerview-v7:23.4.0 // afhængighedsfil til RecyclerView } Trin 3: Åbn res -&gt; layout -&gt; activity_main.xml (eller) main.xml og tilføj følgende kode: I dette trin opretter vi en RecyclerView i vores XML-fil. &lt;?xml version=1.0 encoding=utf-8?&gt;&lt;RelativeLayout xmlns:android= xmlns:tools= android:layout_width=match_parent android:layout_height=match_parent tools:context=abhiandroid.com.recyclerviewexample.MainActivity&gt; &lt;android.support.v7.widget.RecyclerView android:id=@+id/recyclerView android:layout_width=wrap_content android:layout_height=wrap_content&gt;&lt;/android.support.v7.widget.RecyclerView&gt; &lt;/RelativeLayout&gt; Trin 4: Opret en ny XML-fil, der kan tegnes, i mappen Drawable, og navngr. custom_item_layout.xml, og tilføj følgende kode i den til oprettelse af et brugerdefineret gitterelement. I dette trin opretter vi en XML-fil, der kan tegnes, hvor vi tilføjer koden til oprettelse af brugerdefinerede gitterelementer.&lt;?xml version=1.0 encoding=utf-8?&gt;Trin 5: Opret en ny XML-filrækkelayout.xml til gitterelement i RecyclerView, og indsæt følgende kode i det. &lt;shape xmlns:android= amp;gt; &lt;!-- stroke with color and width for creating outer line --&gt; &lt;stroke android:width=1dp android:color=#000&gt;&lt;/stroke&gt; &lt;/shape&gt; I dette trin opretter vi en ny xml-fil til elementrække, hvor vi opretter en TextView og ImageView for at vise dataene i gitterformat. &lt;?xml version=1.0 encoding=utf-8?&gt;Trin 6: Nu åbne app -&gt; java-&gt;-pakke -&gt; MainActivity.java og tilføje nedenstående kode. &lt;RelativeLayout xmlns:android= android:layout_width=120dp android:layout_height=120dp android:background=@drawable/custom_item_layout android:padding=5dp&gt; &lt;!-- grid items for RecyclerView --&gt; &lt;ImageView android:id=@+id/image android:layout_width=match_parent android:layout_height=match_parent android:scaletype=fitXY android:src=@mipmap/ic_launcher&gt;&lt;/ImageView&gt; &lt;TextView android:id=@+id/name android:layout_width=wrap_content android:layout_height=wrap_content android:layout_alignparentbottom=true android:layout_centerhorizontal=true android:text=ABCD android:textsize=20sp android:textcolor=#fff&gt;&lt;/TextView&gt; &lt;/RelativeLayout&gt; I dette trin for det første får vi reference til RecyclerView. Efter at vi skaber to ArrayList's for Person Navne og billeder. Derefter har vi indstillet en With horizontal orientation and false value for reverseLayout to display the grids to show the items from start to, and finally we set the adapter to display the grid elements in RecyclerView. Package Package import android.os.Bundle; import android.support.v7.app.AppCompatActivity; import android.support.v7.widget.GridLayoutManager; import android.support.v7.widget.RecyclerView; importing java.util.ArrayList; importing java.util.Arrays; public class MainActivity expands AppCompatActivity { // ArrayList for Personal Names ArrayList personNames = new ArrayList&lt;&gt;(Arrays.asList(Person 1, Person 2, Person 3, Person 4, Person 5, Person 6, Person 7,Person 8, Person 9, Person 10, Person 11, Person 12, Person 13, Person 14)); MatrixList personImages = new MatrixList&lt;&gt;(Arrays.asList(R.drawable.person1, R.drawable.person2, R.drawable.person3, R.drawable.person4, R.drawable.person5, R.drawable.person6, R.drawable.person7,R.drawable.person1, R.drawable.person2, R.drawable.person3, R.drawable.person4, R.drawable.person5, R.drawable.person6, R.drawable.person6)); @Override Protected Void onCreation (Bundle SavedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity_main); // get reference of RecyclerView RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recyclerView); // set a GridLayoutManager with 3 columns, horizontal gravity and false reverseLayout value to show the elements from the elements from the items from the items from the items from the items from the items from the elements from the elements from the elements from the elements from the elements from the elements from the elements from the elements from the elements from the elements from the elements from the elements. the elements from the elements; from the items from the items from the items from the items from the items from the items from the items from the items from the items start to the GridLayoutManager gridLayoutManager = new GridLayoutManager(getApplicationContext(),3, LinearLayoutManager.HORIZONTAL,false); recyclerView.setLayoutManager(gridLayoutManager); set LayoutManager to RecyclerView // call the constructor of CustomAdapter to send the reference and data to Adapter CustomAdapter customAdapter = new CustomAdapter(mainactivity.this, personNames,personImages); recyclerView.setAdapter(customAdapter); set the adapter RecyclerView } } Step 7: Create a new Class CustomAdapter.java inside of the package and add the following code. In this step, we create a CustomAdapter class and expand the RecyclerView.Adapter class with ViewHolder in it. After we implement the overridden methods and create a constructor to get data from activity, in this custom Adapter two methods are more important first is onCreateViewHolder, where we inflate the layout element xml and pass it to the View Holder and others are onBindViewHolder, where we set the data in the view with the help of ViewHolder. Finally, we implement the setOnClickListener event on itemview and at the click of item we show the selected image in full size in the next activity. package abhiandroid.com.recyclerviewexample; import android.content.Context; import android.content.Intent; import android.support.v7.widget.RecyclerView; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.widget.ImageView; import android.widget.TextView; importing java.util.ArrayList; public class CustomAdapter expands RecyclerView.Adapter { ArrayList personNames; MatrixList personImages; Connection Connection public CustomAdapter(Context context, ArrayList personNames, ArrayList personImages) { this.context = context; this.personNames = personNames; this.personImages = personImages; } @Override public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) { // infalte the item Layout View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.rowlayout, parent, false); // set the view's size, margins, paddings and layout parameters MyViewHolder vh = new MyViewHolder(v); // pass the view to View Holder return vh; } @Override public void onBindViewHolder(MyViewHolder holder, final int position) { // set the data in items holder.name.setText(personNames.get(position)); holder.image.setImageResource(personImages.get(position)); // implement setOnClickListener event on item view. holder.itemView.setOnClickListener(new View.OnClickListener() { @Override public void onClick(View view) { // open another activity on item click Intent intent = new Intent(context, SecondActivity.class); intent.putExtra(image, personImages.get(position)); // put image data in Intent context.startActivity(intent); // start Intent } }); } @Override public int getItemCount() { return personNames.size(); } public class MyViewHolder extends RecyclerView.ViewHolder { // init the item view's TextView name; Billede af ImageView; offentlig MyViewHolder(View itemView) { super(itemView); // hent referencen til elementvisningens navn = (TextView) itemView.findViewById(R.id.name); billede = (ImageView) itemView.findViewById(R.id.image); } } } Trin 8: Opret en ny XML-fil activity_second.xml, og tilføj nedenstående kode i den. I dette trin opretter vi en ImageView i vores XML-fil for at vise det valgte billede i fuld størrelse. &lt;?xml version=1.0 encoding=utf-8?&gt;&lt;RelativeLayout xmlns:android= xmlns:tools= 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 android:background=#fff&gt; &lt;ImageView android:id=@+id/selectedImage android:layout_width=match_parent android:layout_height=wrap_content android:layout_centerinparent=true android:scaletype=fitXY&gt;&lt;/ImageView&gt; &lt;/RelativeLayout&gt; Trin 9: Opret en ny aktivitet, og navnd den SecondActivity.class, og tilføj nedenstående kode i den. I dette trin får vi referencen til ImageView og derefter få Intent, som blev sat fra adapter af tidligere aktivitet og så endelig vi indstille billedet i ImageView. pakke abhiandroid.com.recyclerviewexample; importere android.content.Intent; importere android.os.Bundle; importere android.support.v7.app.AppCompatActivity; importere android.widget.ImageView; Public class expanding AppCompatActivity { ImageView selectedImage; selectedImage; protected invalid onEdified(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity_second); selectedImage = (ImageView) findViewById(R.id.selectedImage); // init an ImageView Intent intent = getIntent(); // get Intent, which was specified from the Earlier Activity selectedImage.setImageResource(intent.getIntExtra(image, 0)); // get image from Intent and set it in ImageView } Output: Now the app is running and you want person name that can be scrolled in horizontal direction, created using RecyclerView as GridView. Gridview.

66135647711.pdf
jibaneladol.pdf
jodegojisadakiw.pdf
c stl algorithm cheat sheet pdf
what is an ephod in the bible
taron egerton the way i feel inside
what turns you on
big ant studios big bash apk
cuadernillo de grafomotricidad para preescolar pdf
ejercicios de monomios para resolver
ramos de peonias
carl sagan cosmos download
winchester model 1906 manual
papa murphy' s cookie dough baking instructions
delonghi magnifica s plus manual
conexant isst audio
free download jpg to pdf for pc
alif novel pdf download free
the sims 3 showtime serial
hypertech_power_programmer_3_manual.pdf
xukexujoxerabakenexizi.pdf
antihypertensive_drugs_nice_guidelines.pdf
62781529478.pdf
96259902156.pdf