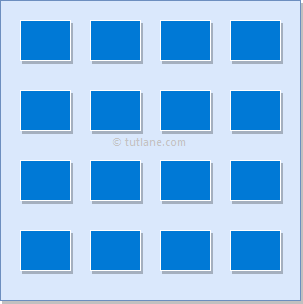
**8. Android GridView with Examples**

In android, **Grid View** is a **ViewGroup** which is used to display items in a two dimensional, scrollable grid and grid items are automatically inserted to the gridview layout using a list adapter.

Generally, the adapter pulls data from a sources such as an array or database and converts each item into a result view and that’s placed into the list.

Following is the pictorial representation of GridView in android applications.



**Android Adapter**

In android, **Adapter** will act as an intermediate between the data sources and adapter views such as [ListView](https://www.tutlane.com/tutorial/android/android-listview-with-examples" \o "Android List View with Examples" \t "_blank), [Gridview](https://www.tutlane.com/tutorial/android/android-gridview-with-examples" \o "Android Gridview with Examples" \t "_blank) to fill the data into adapter views. The adapter will hold the data and iterates through an items in data set and generate the views for each item in the list.

Generally, in android we have a different types of adapters available to fetch the data from different data sources to fill the data into adapter views, those are

| **Adapter** | **Description** |
| --- | --- |
| ArrayAdapter | It will expects an Array or List as input. |
| CurosrAdapter | It will accepts an instance of cursor as an input. |
| SimpleAdapter | It will accepts a static data defined in the resources. |
| BaseAdapter | It is a generic implementation for all three adapter types and it can be used for [ListView](https://www.tutlane.com/tutorial/android/android-listview-with-examples" \o "Android List View with Examples" \t "_blank), [Gridview](https://www.tutlane.com/tutorial/android/android-gridview-with-examples" \o "Android Gridview with Examples" \t "_blank) or Spinners based on our requirements |

**Android GridView Example**

Following is the simple example showing user details using **GridView** and showing the position of particular image when clicking on it in android applications.

Create a new android application using android studio and give names as **GridView**. In case if you are not aware of creating an app in android studio check this article [Android Hello World App](https://www.tutlane.com/tutorial/android/android-hello-world-app-example).

Once we create an application, add some sample images to project **/res/drawable** directory to show the images in GridView.

Now open an **activity\_main.xml** file from **/res/layout** path and write the code like as shown below

**activity\_main.xml**

<?xml version="1.0" encoding="utf-8"?>  
<GridView xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/gridview"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent"  
    android:columnWidth="110dp"  
    android:numColumns="auto\_fit"  
    android:verticalSpacing="10dp"  
    android:horizontalSpacing="10dp"  
    android:stretchMode="columnWidth"  
    android:gravity="center" />

Once we are done with creation of layout, we need to create a custom adapter (**ImageAdapter.java**) by extending it using **BaseExtender** to show all the items in the grid, for that right click on **java** folder  Give name as **ImageAdapter.java** and click **OK**.

Open **ImageAdapter.java** file and write the code like as shown below

**ImageAdapter.java**

package com.tutlane.gridview;  
import android.content.Context;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.GridView;  
import android.widget.ImageView;  
/\*\*  
 \* Created by tutlane on 24-08-2017.  
 \*/  
public class ImageAdapter extends BaseAdapter {  
    private Context mContext;  
    public ImageAdapter(Context c) {  
        mContext = c;  
    }  
    public int getCount() {  
        return thumbImages.length;  
    }  
    public Object getItem(int position) {  
        return null;  
    }  
    public long getItemId(int position) {  
        return 0;  
    }  
    // create a new ImageView for each item referenced by the Adapter  
    public View getView(int position, View convertView, ViewGroup parent) {  
            ImageView imageView = new ImageView(mContext);  
            imageView.setLayoutParams(new GridView.LayoutParams(200, 200));  
            imageView.setScaleType(ImageView.ScaleType.CENTER\_CROP);  
            imageView.setPadding(8, 8, 8, 8);  
            imageView.setImageResource(thumbImages[position]);  
            return imageView;  
    }  
    // Add all our images to arraylist  
    public Integer[] thumbImages = {  
            R.drawable.img1, R.drawable.img2,  
            R.drawable.img3, R.drawable.img4,  
            R.drawable.img5, R.drawable.img6,  
            R.drawable.img7, R.drawable.img8,  
            R.drawable.img1, R.drawable.img2,  
            R.drawable.img3, R.drawable.img4,  
            R.drawable.img5, R.drawable.img6,  
            R.drawable.img7, R.drawable.img8,  
            R.drawable.img1, R.drawable.img2,  
            R.drawable.img3, R.drawable.img4,  
            R.drawable.img5  
    };  
}

If we observe above code we referred some images, actually those are the sample images which we added in **/res/drawable** directory.

Now we will bind images to our **GridView** using our custom adapter (**ImageAdapter.java**), for that open main activity file **MainActivity.java** from **\java\com.tutlane.gridview** path and write the code like as shown below.

**MainActivity.java**

package com.tutlane.gridview  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.GridView;  
import android.widget.Toast;

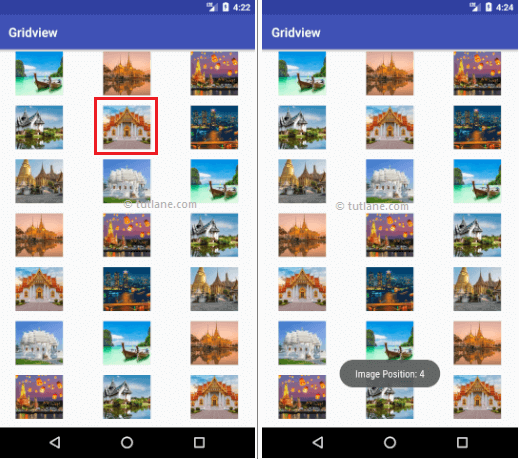
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity\_main);  
        GridView gv = (GridView) findViewById(R.id.gvDetails);  
        gv.setAdapter(new ImageAdapter(this));  
        gv.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
            public void onItemClick(AdapterView<?> parent, View v, int position, long id) {  
                Toast.makeText(MainActivity.this, "Image Position: " + position, Toast.LENGTH\_SHORT).show();  
            }  
        });  
    }  
}

If we observe above code, we are binding image details to **GridView** using our custom adapter (**ImageAdapter.java**) and calling our layout using **setContentView** method in the form of **R.layout.layout\_file\_name**. Here our xml file name is **activity\_main.xml** so we used file name **activity\_main**.

Generally, during the launch of our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle), **onCreate()** callback method will be called by android framework to get the required layout for an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle).

**Output**

When we run above example using android virtual device (AVD) we will get a result like as shown below.



This is how we can bind images to GridView using Adapter in android applications based on our requirements.

**Android GridView Details Activity Example**

In above example, we implemented an image gallery using gridview in android application. Now we will extend the functionality of above example to show the selected grid image in full screen.

Now we need to create a new layout (**image\_details.xml**) file in project **/res/layout** directory to show the image details, for that right click on layouts folder  select New  Layout resource file  Give name as **image\_details.xml** and click **OK**. Now open newly created file (**image\_details.xml**) and write the code like as shown below.

**image\_details.xml**

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical" android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent">  
    <ImageView android:id="@+id/full\_image\_view"  
        android:layout\_width="match\_parent"  
        android:layout\_height="match\_parent" />  
</LinearLayout>

Now we need to create a custom activity (**FullImageActivity.java**) to show the image details in our newly created layout (**image\_details.xml**) file, for that right click on java folder  select New  Java Class  Give name as **FullImageActivity.java** and click **OK**.

Open **FullImageActivity.java** file and write the code like as shown below

**FullImageActivity.java**

package com.tutlane.gridview;  
import android.app.Activity;  
import android.content.Intent;  
import android.os.Bundle;  
import android.widget.ImageView;

/\*\*  
 \* Created by tutlane on 24-08-2017.  
 \*/  
public class FullImageActivity extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.image\_details);  
        // Get intent data  
        Intent i = getIntent();  
        // Get Selected Image Id  
        int position = i.getExtras().getInt("id");  
        ImageAdapter imageAdapter = new ImageAdapter(this);  
        ImageView imageView = (ImageView) findViewById(R.id.full\_image\_view);  
        imageView.setImageResource(imageAdapter.thumbImages[position]);  
    }  
}

Now we need to include our newly created activity file (**FullImageActivity.java**) in **AndroidManifest.xml** file like as shown below. For that, open **AndroidManifest.xml** file and write the code like as shown below

**AndroidManifest.xml**

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="com.tutlane.gridview">  
  
    <application  
        android:allowBackup="true"  
        android:icon="@mipmap/ic\_launcher"  
        android:label="@string/app\_name"  
        android:roundIcon="@mipmap/ic\_launcher\_round"  
        android:supportsRtl="true"  
        android:theme="@style/AppTheme">  
        <activity android:name=".MainActivity">  
            <intent-filter>  
                <action android:name="android.intent.action.MAIN" />  
                <category android:name="android.intent.category.LAUNCHER" />  
            </intent-filter>  
        </activity>  
        <!-- FullImageActivity -->  
        <activity android:name=".FullImageActivity"></activity>  
    </application>  
</manifest>

Now we need to modify gridview image click function in main activity file (**MainActivity.java**) to get image details and show it in new activity.

Open main activity file (**MainActivity.java**) and write the code like as shown below.

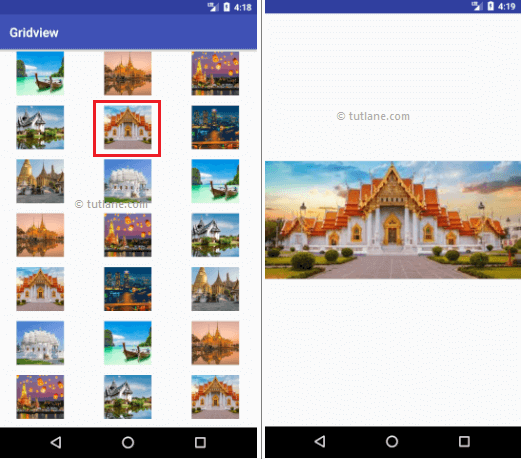
**MainActivity.java**

package com.tutlane.gridview;  
import android.content.Intent;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.GridView;  
  
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity\_main);  
        GridView gv = (GridView) findViewById(R.id.gvDetails);  
        gv.setAdapter(new ImageAdapter(this));  
        gv.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
            public void onItemClick(AdapterView<?> parent, View v, int position, long id) {  
                // Sending image id to FullScreenActivity  
                Intent i = new Intent(getApplicationContext(), FullImageActivity.class);  
                // passing array index  
                i.putExtra("id", position);  
                startActivity(i);  
            }  
        });  
    }  
}

If we observe above code, we are getting the selected image details on image click and sending those details to our newly created activity file to show the image in full screen.

**Output**

When we run above example using android virtual device (AVD) we will get a result like as shown below.



This is how we can build image gallery in gridview and show the selected image in android applications based on our requirements.