Leica captivate user manual

SimpleAdapter's public class expands BaseAdapter implements a Filterable, ThemedSpinnerAdapter Simple Adapter for static data maps for views identified in the XML file. You can specify data supporting the list as ArrayList cards. Each entry in ArrayList corresponds to one line in the list. Maps contain data for each line. You also specify an XML file that identifies the views used to display a line and displays from the keys on the map to certain views. The data is tied to views in two stages. First, if SimpleAdapter.ViewBinder is available, ViewBinder-setViewValue (android.view.View, Object, String) is called. If the value returned is correct, there is a binding. If the returned value is false, the following representations are judged in the order: if no appropriate binding can be found, the Illegal State Exclusion is thrown out. SimpleAdapter.ViewBinder this class can be used by SimpleAdapter's external customers to link values to views. From the java.lang.Object Object class, the clone creates and returns a copy of this object. boolean (Object obj) indicates whether any other object is equal to this. invalid completion () Is called by the garbage collector at the facility when the garbage collection determines that there are no more references to the object. The final getClass class returns the time class of the subject. int hashCode () Returns the hash code value to the object. the final notification of the void () will wake up one stream that waits on the monitor of this zlt;?? The object. ToString returns the view of the object line. The final expectation of emptiness (long time out, int nanos) triggers anticipation of the current thread until another thread triggers the notification method () or the notifyAll method for that object, or some other thread interrupts the current thread, or a certain amount of real time has passed. The final expectation of emptiness (long time) triggers the wait for the current thread until another notification method () or notifyAll method has been triggered for that object, or a certain amount of time has passed. the final expectation of emptiness () causes the current thread to wait until another thread triggers the notification method () or the notifyAll method for that object. Public SimpleAdapter (context context, zlt; zlt;String, ?gt; list data, int-resource, string from the context of int parameters) Context: Context, where the view associated with this SimpleAdapter, runs a list of data: Map List. Each entry on the List corresponds to one line in the list. Maps contain data for each line and must include all entries listed in the int resource from: View layout resource ID, identifying for this item on the list. The layout file should include at least those named views identified in the to line: a list of column names that will be added to the Map. with every item. int: Views that should appear in the column in the set from. All this has to be TextViews. The first N views in this list are given the values of the first N columns by parameter. How many items are in the dataset provided by this adapter. Returns int Graph of The Elements. See also: GetDropDownView (int position, ConvertView ViewView, ViewGroup Parent View) Receives a view that appears in the fall of the data pop-up in the specified position in the dataset. Int position settings: the item index we want. convert View View: old look for reuse if possible. Note: You should check that this view is not zero and of the appropriate type before use. If you can't convert this view to display the right data, this method can create a new view. Parent ViewGroup: The parent that this view will eventually be attached to Returns View View, relevant data in the specified position. the getFilter public filter returns a filter that can be used to limit data with a filter pattern. This method is usually implemented by Adapter classes. Returns the filter used to restrict the data of the public object getItem (int position) Get the data element associated with the specified position in the dataset. Int Position Options: Position of the item we want in the adapter dataset. Returns the Data object in this position. See also: Public Long GetItemId (int position) Get a line ID associated with the specified position in the list. Int position options: Position position position in the adapter dataset that we want to do. Returns a long item ID in the specified position. See also: Public View getView (position int, View convertView, ViewGroup parent) Gets a view by displaying data in a specified position in the dataset. You can create a view manually or inflate it from the XML layout file. When the view is overpriced, the parent view (GridView, ListView...) will apply the default layout settings if you don't use LayoutInflater.inflate (int, android.view.ViewGroup, boolean) to indicate the root look and prevent the root attachment. Int position settings: Position position position position in the element adapter dataset that we want. convert View View: Old view for reuse if possible. Note: You should check that this view is not zero and of the appropriate type before use. If you can't convert this view to display the right data, this method can create a new view. Heterogeneous lists can their number of view types, so this kind is always the right type (see BaseAdapter.getViewTypeCount () and BaseAdapter.getItemViewType (int)). Parent ViewGroup: Parent that this view will eventually be attached to Returns View A View, relevant data in the specified position. Seeing See Adapter.getView (int, View, ViewGroup) public void setDropDownViewTheme (Theme Resources.Theme) sets Resources.Theme against which the types of fall are overstated. By default, the view of the fall is inflated in relation to the Context theme transmitted to the adapter constructor. Topics Options Resources.Theme: a theme against which to inflate a drop down opinion or zero use the theme from the context of the adapter See also: getDropDownView (int, View, ViewGroup) public blank setViewBinder (SimpleAdapter.ViewBinder viewBinder) Installs a binder used to link data to views. ViewBinder SimpleAdapter.ViewBinder Options: The binding value used to link data to views may be invalid to remove the existing link see also: public void setViewImage (ImageView v, Line Value), is called bindView () to customize the image for ImageView, but only if there is no existing ViewBinder or if the existing ViewBinder can't handle the imageView. By default, the value will be considered as an image resource. If the value cannot be used as an image resource, the value is used as a Uri image. This method is called instead of setViewImage (android.widget.ImageView, int) unless the data provided is int or Integer. Options v ImageView: ImageView for ImageView: Value Derived from the DataSet See also: setViewImage (ImageView, int) public blank setViewText (TextView v, String text) is called bindView () to customize text for TextView, but only if there is no existing ViewBinder or if the existing ViewBinder can't handle textView. Options v TextView: TextView for Text Text Line: Text That Will Be Installed for TextView Welcome to Android Simpleadapter Tutorial with an example in Android Studio. We learn about basic information about a simple adapter in depth today. Let me explain a little information about the adapter in the first place. The Android adapter is a bridge between the data source and the UI dynamic views, such as ListView, GridView, or Spinner. For example, you have one array of strings that contains the names of different animals. Now you want to create a list view with this array of lines. Here, the adapter will receive names from an array of lines and then it will provide it to the list. Without an adapter, it is not possible to make dynamic views in android. Android SimpleAdapter If you want to map the data into XML submissions, then a simple adapter is the best option. You can make ArrayList cards to service the data in a simple adapter. Typically, every developer uses An arrayList hash cards. Each hash card a single number of lists views. See the designer below for a simple adapter for more links. SimpleAdapter (Context Context, zlt; zlt;String, ?'gt; data list, int resource, string, int) Above the constructor have five parameters that play a significant role in compiling the list. Simple adapter methods Children's class of basic adapter. This way, it can use all the methods of the basic adapter. Some of these methods are like below one. It will return the number of datasets available in the data source. In other words, it will give the number of lines on the list. 2. getItem (int position) Get a data element associated with this position in the dataset. 3. getView (position int, ConvertView ViewView, ViewGroup Parent) Get a view that displays the data in a specified position in the dataset. We will inflate the special XML layout file in this method. Each listview series will have the same look as this XML file. Values for text viewing, image viewing, etc. in the listview line are also given in this method. 4. getDropDownView (int, View convertView, ViewGroup parent) gets a view that is displayed when pop-up data falls in a specified position in the dataset. 5. getFilter () Returns a filter that can be used to limit data with a filter pattern. 6. The DropDownViewResource set (int resource) installs a layout resource to create drop views. The layout resource means a special XML file to view the fall. 7. setViewBinder (SimpleAdapter.ViewBinder viewBinder) Installs a link used to link data to views. The Simple Adapter Following is the designer of a simple adapter. SimpleAdapter (Context Context, zlt; zlt;String, ?'gt; data list, int resource, string, int) Above the constructor have five parameters that play a significant role in compiling the list. Now let's see how all of the above options work. Context context gives you a link to the class in which we write the designer. We can use this, ActivityName.this, getApplicationContext (, getActivity) to show context. getActivity is commonly used in fragment. this, ActivityName.this is commonly used in action. 2. <? This is a data source. It's a list or a massive list of cards. For example, we can use a massivesiflist of hash cards in this setting. Each massiveist hash card will provide data for each item on the list line. 3. int resource These options refer to the XML layout file that we use to create a view of each item on the list. In the list, gridview or spinner, when we create a custom adapter, we have to make a separate XML file for line items. If you're viewing a list with text and an image in the XML file, we should add one text view and one view of the image. Each listview series will have one text view and one image, because it is inflating this file. this paragraph (int resource) in the example I gave at the end of this tutorial. 4. Line from and int's to String 'from name,image;//line of the array int's to'R'R.id.name,R.id.imgView;//int array of views id from and up to qlt;/String, ? linked to each other. Let's look at the first values of the array of rows from array and a number to. Below is the structure of the hashkarta. Hashmap'lt;String'gt; hashMap'new HashMap ();to create a hash card to store data in a pair of key values of hashMap.put (name, animalName)); hashMap.put (image, animalImages'i)); Hashmap has two types of keys: name and image. Both rows array values are also names and images. Therefore, when the compiler is going to set the value of the hash card name key, it must set the value for R.id.name. R.id.name is the first value of a whole array to and refers to text browsing. Similarly, for a key image of a hash card, the compiler will set its value to R.id.imgView. R.id.imgView is Imageview, and this second option is an array of listView with a simple adapter let's review using a simple adapter. Prepare a new project in the android studio. Step 1. XML Code First of all, we will write the source code for the XML layout files. In activity_main.xml add below the line of the qlt??xml version?1.0 encoding?utf-8?'lt; x/android xmlns:app/ xmlns:tools/ android:layout_width/match_parent android:layout_height'match_parent tools:context. MainActivity'gt; qlt'listView android:id'id/listview android:layout_width'match_parent android:layout_height'match_parent android:divider'#e20fc6 android:divideright'3dp'gt;'lt;listView'gt; qlt;android.support.constraint.ConstraintLayout'gt; We'll attach a simple adapter to this listview. Now make a new XML layout file under the catalog. Give this name, simple_item.xml Write down below a snippet of code at simple_item.xml'lt;xml version?1.0 encoding?utf-8?gt;gt;Linearout xmlns:android' ss/android android:layout_width'match_parent android:layout_height'match_parent android:orientation'vertical'gt;This file of the linearlayout android:layout_width'match_parent android:layout_height'wrap_content android:orientation'horizontal'gt; ImageView android:id'id/imgView android:layout_width'100dp android:layout_height'100dp android:layout_marginleft'10dp android:layout_margintop'10dp'gt;lt;lt/ImageView TextView android:id'id/name android:layout_width'match_parent android:layout_height'110dp android:gravity'center_vertical android:textappearance??android:attr/textAppearanceMedium android: paddingleft'10dp android: text'name'gt; :layout_width=match_parent android:layout_height=1dp android:layout_margintop=10dp android:layout_marginleft=10dp android:layout_marginright=10dp android:background=@color/colorAccent></View> view for each item on the list line. I added one Textview, one Imageview and one simple view in the aforementioned file. The compiler will write the name of the car in text viewing and the image of the car will be installed in the image. A simple look will work as a divider. Step 2. Car Images Download images of the car by clicking on the below link download car demonuts_images_listview After receiving the above images of the car, add them to the catalog. Step 3. The main code in the MainActivity.java file, add the following import source code android.support.v7.app.AppCompatActivity; Import android.os.Bundle; import android.widget.ListView; import android.widget.SimpleAdapter; import java.util.ArrayList; import java.util.HashMap; MainActivity's public class expands AppCompatActivity / //to initialize the view in the private listView listView; private int'myImageList - new int'R.drawable.benz, R.drawable.bike, R.drawable.car,R.drawable.carrera,R.drawable.ferrari,R.drawable.harly, R.drawable.lamborghini. private string myImageNameList - new string Benz, Bicycle, Car, Carrera Ferrari, Harley, Lamborghini,Silver; @Override protected void onCreate (Bundle savedInstanceState) - super.onCreate (savedInstanceState); setContentView (R.layout.activity_main); ArrayList <> <String,String>> arrayList-new ArrayList <> (); for (int i'0; i hashMap-new HashMap ();/create a hash card to store data in a pair of key values of zlt; 8 q (i) hashMap.put (name, myImageNameList-i); hashMap.put (image, myImageList arrayList.add);/add a hash card to arrayList (String) from name, image;//line of array int's to'R.id.name,R.id.imgView;/int array of views id's SimpleAdapter simpleAdapter-new SimpleAdapterAdapter (it's, arrayList, R.layout.simple_item,from,to);/Create an object and set the settings for a simpleAdapter listView.setAdapter (simpleAdapter);//sets the adapter for theView list - As you can read, I created one line, One whole array contains whole values that point to drawable images. ,String> </String,String>