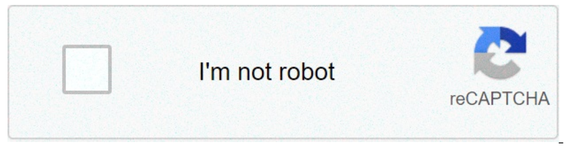

Continue

How to get last position of recyclerview in android

Get the last visible location of Recycler displays the IMPORT Android.support.v7.widget.vportLayoutManager; IMPORT Android.support.v7.widget.Recycuncycle; Main public class {/ ** / * from w w w. JA VA2S.com * / * Get the last visible Vista Recyclage position * * @Param RV * @return * / Static Public Int GetlastvisiblesiBlesposition (RICICKCLERVIEW RV) {if (RV! = NULL) {FINAL RECYCLEY.LAYOUTMANAGER LAYOUTMANAGER = RV .Getlayoutmanager (); IF (LayoutManager ISTANCEOF LINEARLAYOUTMANAGER) {RETURN ((LINEARLAYOUTMANAGER) LAYOUTMANAGER) .FINDLASTVISIBLESITEMPOSITION ();}} Return 0;}} Related Tutorial Public Static Abstract Class RECYCLERVIEW.ViewHolder Extend Object Java.lang.Object Ã, Ã, Ã, Ã , Â³ Androidx.RecyclerView.Widget.RecyclerView.ViewHolder A viewholder describes an element and metadata view of its place inside the Recyclerview. Recyclerviserview.adapter Implementations should pay the subclasses and add fields for cached storage prohibits potentially expensive.Findviewbyid (Int) Results. While Recyclew.layoutParams belongs to recycling ..layoutManager, Versholders belong to the adapter. Adapters should feel free to use their own custom viewholder implementations to store data that makes it easier to view the content. Implementations they should assume that individual views of the OBS object Ieda Strong references to objects of view and that recycling instances can hold strong references to the views of extra off-screen elements for cache storage purposes Final view Rated viewView Viewholder (View Voice) from Class Java.lang.Object Object ( ) Boolean (object arg0) void finalizes () final class Getclass () int hashcode () final voidy notify () final void notifyall () string tostring () final void wait (long arg0, intr1) final void is expected (LONG ARG0) Void Final Void Waiting () Public View Fields View ItemView Constructors Public Viewholder (View Artoloview) Public methods Final Int getabsoluteadapterposition () Returns the position of the adapter of the item represented by this viewfinder with respect to recycling recycling. . If the Recyclerviserview.adatter who bounded this Recyclerviserview.Viewolder is within another adapter (eg ConcatAdapter), this position may be different and will include offsets caused by other adapters in the Concatadapter. Note that this could be different from GetLayOutposition () if there are updates waiting for adapters, but a new layout pass has not yet happened. Recycling does not manage adapter updates up to the next Traversal layout. This can create temporary inconsistencies between which user sees on the screen and which contents of the adapter they have. This inconsistency is not important because it will be less than 16 ms, but it could be a problem if you want to use the position of the payment to access the adapter. Sometimes, you may need to get the exact position of the adapter to perform some actions in response to the user's events. In this case, it is necessary to use this method that will calculate the viewer adapter position. Note that if you called recyclerviserview.adapter.notifydatasetitchanged (), up to the next layout step, the return value of this method will be recycled.no_position. Note that if you are questioning the position as you see recycling, you need to use getabsoluteadapterposition () (for example. You want to use to save the scroll status). If you are questioning the position to access the recycling content .Adapter, you need to use GetBindingAdatterposition (). See also: getbindingadatterposition () GetLayoutPosition () Public Final Int GetbindingaDapterPosition () The position of the object adapter represented by this viewfinder regarding recycling. Note that this could be different from GetLayOutposition () if there are updates waiting for adapters, but a new layout pass has not yet happened. Recycling does not manage adapter updates up to the next Traversal layout. This can create temporary inconsistencies between which user sees on the screen and which contents of the adapter they have. This inconsistency is Important as it will be less than 16 ms, but it could be a problem if you want to use the position of the payment to access the adapter. Sometimes, you may need to get the exact position of the adapter to perform some actions in response to the user's events. In this case, it is necessary to use this method that will calculate the viewer adapter position. Note that if you called recyclerviserview.adapter.notifydatasetitchanged (), up to the next layout step, the return value of this method will be recycled.no_position. If the Recyclerviserview.adapter that has bounded this recycling recycling. Viewholder is within another recycling recycling.adapter (eg Concatadapter), this location may be different from GetabbsoluteadatterPervosition (). If you want to know the location that RecyclerVEW considers (for example for the saved status), you need to use GetabsoluteaDatterPosition (). See also: getabsoluteadapterposition () GetlayoutPosition () Public Final Long Getititemid () Returns the object represented by this viewfinder. PUBLIC FINAL INT GETETEMViewType () Returns int The kind of view of this viewfinder. Public Final Int GetlayoutPosition () Returns the viewfinder's position in terms of passing the most recent layout. This position is used for the most used by recycling components to be consistent during recycling recycling for recycling for processes in panorama of adapter updates. For reasons of performance and animation, recycling recycling is beating all adapter updates until the next layout passage. This could cause misalignments between the position of the article adapter and the position he had in the calculations of the most recent layout. LayoutManagers should always call this method while you make calculations based on the locations of the article. All methods in Recycleview.LayoutManager, Recyclerview.State, Recycleview.Recycler receiving a position expects to be the position of the item layout. If LayoutManager must call an external method that requires the position of the article adapter, can use GetabbsoluteadAdapterPosition () or recycling recycling.Recycler.convertPrelayoutPositionTopostlayout (int). Int returns returns the viewfinder adapter position in the last layout pass. See also: getbindingadatterposition () Getabbsoluteadapterposition () Public Final Int GetoldPosition () When LayoutManager supports animations, recycling tracks 3 positions for experts to perform animations. If a viewfinder has been arranged in the previous call onlayout, the old position will keep its adapter index in the previous layout. Returns int The index of the previous adapter of the object represented by this viewfinder or recyclew.no_position If the old position does not exist or canceled (pre-layout is complete). Public Final Boolean Isrecyclication () Returns Boolean True if this article is available to be recycled, false otherwise. See also: Public Final Void Setislecicload (Boolean Riciclable) informs the recycler if this voice can be recycled. The views that are not recyclable will not be reused for other elements until Setislecycyclication () later set to true. Calls to Setrecyclication () should always be paired (a call to Setisrecyclabe (false) should always be combined with a subsequent call to Setislecycyclible (True))). Calls of calls can be nested, since the status is counted internally of reference. Boolean recyclable parameters: if this item is available to be recycled. The default value is true. Public String Tostring () Recycling is a view group that makes any view based on adapter Similar way. It should be the successor of ListView and GridView. One of the reasons is that the Recyclervivise has a more extensible picture, especially because it provides the possibility of implementing both horizontal and vertical layouts. Use the Recyclerview widget when you have data collections whose items change in runtime based on user action or network events. If you want to use a recyclvisciew, you need to work with the following: reconickciew.adapter - To manage data collection and tie it to the view layout: Helps position the items Itemanimator - helps to animate the items for Operations such as addition or removal of element also provides animation support for Recyclerview elements when added or removed, which had been extremely difficult to do with ListView. Recyclerview also starts to enforce the viewholder model too, which was already a recommended practice, but now deeply integrated with this new picture. For more details, see this detailed overview. Compared to the ListView Recyclerview differs from its predecessor ListView mainly due to the following features: required Viewholder in adapters - ListView adapters do not require the use of the Viewholder model to improve performance. In contrast, the implementation of a recyclerview adapter requires the use of the model for which Viewholder uses RecyclerView.Viewholder. Customizable article Layout - ListView can only layout elements in a vertical linear arrangement and this cannot be customized. On the contrary, the RecyclerView has a recyclerview.LayoutManager that allows any article layout including horizontal lists or staggered grids. Simple object animations - ListView does not contain special provisions through which you can animate the addition or cancellation of elements. On the contrary, the Recyclerview has the RECYCLERVIEW.Itemanimator class for voice animation management. Data Source Manual - ListView had adapters for different sources, such as arrayadapter and cursoradapter respectively for arrays and database results. On the contrary, the RecyclerView.Adapter requires a custom implementation to provide data for the adapter. Article Decoration Manual - ListView has the Android: property divider for easy dividers between the elements of the list. On the contrary, RecyclerView requires the use of a RECYCLERVIEW.ItemDecoration object to install more manual divider decorations. Manual Click Detection - ListView has an AdapterView.onitemClickListener interface for linked events for the individual list items. On the contrary, Recyclerview has only support for RecyclerView.onitemtouchListener that manages individual touch events, but is not equipped with click management. Components of a RECYCLERVIEW LAYOUTMANAGERS A RECYCLERVIEW to have a layout manager and an adapter to be instantiated. A view layout manager locations in a recyclerview and determines when reusing views of the article that is no longer visible to the user. Recyclerview provides these embedded layout managers: linearlayoutmanager shows items in a vertical or horizontal sliding list. GridlayoutManager shows items in a grid. StagledGridlayoutManager shows elements in a staggered grid. To create a custom layout manager, extend the RECYCLERVIEW.LAYOUTMANAGER class. Here we talk about Dave Smith on the personalized layout notes manager: in the recent version of the support library, if you do not explicitly set the layoutmanager, the Recyclerview will not show! There is a logcat error if and / recyclerview: no attached layout manager; Layout Jump RECYCLERVIEW.Adapter RecyclerView includes a new type of adapter. It is a similar approach to those that already used, but with some peculiarities, such as a requested viewholder. You will have to replace two main methods: one to inflate the view and its own titrator, and another for the data bind to sight. The good thing about this is that the first method is called only when we really need to create a new view. There is no need to check if it's been recycled. ITemanimator Recyclerview.Itemanimator AnimerÃâGroup changes, how to add / delete / select that are notified to the card. DefaultTitematorImator can be used for basic predefined animations and pretty well. See the section of this guide for more information. Using the RecyclerView using a RecyclerView has the following key steps: Define a model class to use as a data source Add a RecyclerView to your activity to view items Create a custom XML layout file to view the item Create a recyclerview.adapter and Viewholder to make the voice Bind the data data adapter To populate the recycling, the steps are explained in more detail below. Definition of a model recycling Each is supported by a source for data. In this case, we will define a contact class that represents the data model displayed by recycling: public class Contact {private String MNAME; Monline private boolean; public contact (String name, boolean online) {MNAME = name; Monline = online; } Public String GetName () {RETURN MNAME; } Public boolean IsOnline () {return Monline; } Privatic Static Int LastContactid = 0; public static ArrayList CreateContactStalist (Int Numcontacts) {ArrayList = New arrayList (); for (int i = 1; i