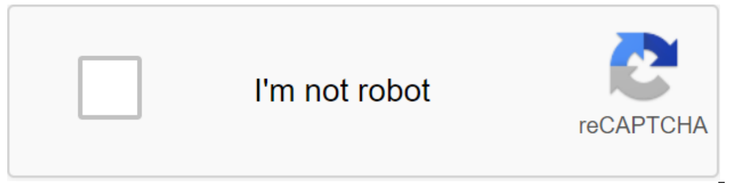
Lazy loading in android recyclerview example github



Continue

A simple library that helps the master download more features from RecyclerView. Written in kotlin, supported by Java Available through the jcenter repository ... jcenter () ... com.github.alkurop:updatinglist:0.2.5 clones the project for demo page 2 Simple Library, which helps the master download more features from RecyclerView. Written in kotlin, supported by Java Available through the jcenter repository ... jcenter () ... compilation com.github.alkurop:updatinglist:0.2.5 clone project to demonstrate This example shows how to effectively use Microstream storage with Android RecyclerView. RecyclerView is an effective way to implement a dynamic list of views. It is often used on various social networks to show comments or messages. first, you should add a Microstream repository to your project. So in the build.gradle file for the entire project, add these lines: allprojects - repositories - google () jcenter () maven - URL ' - and for the app, Add the following dependencies: implementation 'one.microstream:storage.embedded:02.02.00-MS-GA' implementation 'one.microstream:storage.embedded.configuration:02.02.00-MS-GA' implementation 'one.microstream:storage.embedded.configuration:02.02.00-MS-GA' implementation 'one.microstream:storage.embedded.configuration:02.02.00-MS-GA' implementation 'one.microstream:storage.embedded androidx.recyclerview:recyclerview:1.1.0' abstract Process one.microstream:base:02.02.00-MS-GA This example shows a list of fake customers with a name and address. Customers are created using the JavaFaker library and stored in Microstream. Scroll through the view, you dynamically create new records from either existing customer instances downloaded from storage or from newly created ones stored on the fly. This example shows the functionality of the Microstream download. package one.microstream.android.data; import java.util.HashMap; import java.util.Map; import one.microstream.reference.Lazy; CustomerRoot's public class, the customerMap. CustomerRoot - CustomerMap - the new HashMap Thus, the app doesn't need to download all customers from data storage at once, but just a lazy link pointing to them. Only when RecyclerView actually requires customers to show them on the display, will they be downloaded from data storage if they are not downloaded yet. RecyclerView also offers the option of removing items from memory that are no longer displayed. The app just needs to call a clear () method to a lazy link. This approach allows you to show thousands of items on the screen without destroying a bunch of apps. This project is ready to be built with the help of Android studio. Endless Scroll (Infinite Scroll) for in Android JavaDoc is available by Motivation of content For a long time I'm a customer. Find the correct implementation of the endless scroll AKA endless scroll for Android. A few solutions that I found weren't ready for production, didn't work properly, or had too many features. I wanted to have a small, simple and flexible solution to implement an endless scroll for Android that works with RecyclerView from the newest Android API. That's why this project was created. Examples of the app can be found in the app directory. Here's an animation of how the app's example works. You can also see the exam use of this library in the SearchTwitter app. Use Create the necessary fields in your activities: public recyclerView recyclerView; Private LinearLayoutManager mock-upManager; Create the new InfiniteScrollListener: private InfiniteScrollListener createInfiniteScrollListener -- bring back the new InfiniteScrollListener (maxItemsPerquestRequest, layoutManager) - @Override public void onScrolledToEnd (final int firstVisibleItemPosition) / // Download your items here // The logic of downloading items will vary depending on the case of use // Combine old and new items, transfer them to the adapter // and call refreshView (...) method from the Class InfiniteScrollListener to update RecyclerView Initiate RecyclerView and LinearLayoutManager in your activities: @Override protected void onCreate (Bundle SavedSation) - recyclerView (RecyclerView) findViewById (R.id.recycler_view); layoutManager - the new LinearLayoutManager (it); recyclerView.setHasFixedSize recyclerView.setLayoutManager (layoutManager); Install your custom recyclerView.setAdapter (new MyAdapter (elements); Add InfiniteScrollListener as OnScrollListener recyclerView.addOnScrollListener If you want to display download progress, you should add an extra look for it, show it during the start of download and hide it when the download is finished. Check out the exemplary app in this repository to see a specific solution. That's it! Download the latest version: replace x.y.z with the latest version, you can rely on the library through Maven: zlt.dependency.gt; zlt.groupId'gt;com.github.pwittchen/lt/groupId'gt; or via Gradle: Dependencies and compilations 'com.github.pwittchen:infinitescroll:x.y.z' Tests to perform a unit of tests: ./gradlew code style used in the project, called SquareAndroid Static code analysis to run static code analysis, v.: ./gradlew reports on analysis verification in the library/assembly/report/catalogue. Who uses this library? Do you use this library in your app and want to be listed here? Send me a Pull request or email to piotr@wittchen.io license license 2016 Piotr Wittchen License licensed by Apache, version 2.0 (License); You can't use this file unless it's in compliance with the License. You can obtain a copy of the license by phone If it is not required by applicable law or not agreed in writing, the software distributed under the License is distributed to AS IS BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. You can see the License for a specific language that regulates permits and restrictions under the License. EndlessRecyclerView allows you to download new pages when the user scrolls down to the bottom of the list. It expands RecyclerView and is fully customizable to meet various development needs. The library is compatible with Android 14. Dependence on use To use the library in your project write this code for your build.gradle: buildscript - Repository - jcenter() - dependency - compilation 'com.github.yasevich:endless-recycler-view:2.0.0' - or include it in the lib folder. Layout You can include EndlessRecyclerView to your layout as follows: zlt;com.github.yasevich.endlessrecyclerview.EndlessRecyclerView xmlns:android/ @android:id/list android:orientation 'vertical android:layout_width'match_parent android:layout_height'match_parent'lt;'lt;com.github.yasevich.endlessrecyclerview.EndlessRecyclerView'gt; Initialization Of The Settings Of EndlessRecyclerView As needed: EndlessRecyclerView list ...; /; initialization of list.setLayoutManager (new LinearLayoutManager); list.setProgressView (R.layout.item_progress); list.setAdapter(adapter); list.setAdapter(adapter); Also, when the page is downloaded, you can stop showing progress: list.setRefreshing (false); More information can be found in the samples and documentation. Page 2 EndlessRecyclerView allows you to download new pages when the user scrolls down to the bottom of the list. It expands RecyclerView and is fully customizable to meet various development needs. The library is compatible with Android 14. Dependence on use To use the library in your project write this code for your build.gradle: buildscript - Repository - jcenter() - dependency - compilation 'com.github.yasevich:endless-recycler-view:2.0.0' - or include it in the lib folder. Layout You can include EndlessRecyclerView to your layout as follows: zlt;com.github.yasevich.endlessrecyclerview.EndlessRecyclerView xmlns:android/ @android:id/list:match_parent layout_width android Initialization Setting Of EndlessRecyclerView Settings As Needed: EndlessRecyclerView List...; /; initialization of list.setLayoutManager (new LinearLayoutManager); list.setProgressView (R.layout.item_progress); list.setAdapter(adapter); list.setAdapter(adapter); In addition, if you download a page, you may want to stop showing progress: work: More information can be found in the samples and documentation. In this post I've included a simple approach to implementing Infinite Load more with download progress in RecyclerView. I made this project as easy as possible, the main kind of lists of movie titles rated IMDB (loads from the server). When you reach the bottom, scroll down the list, the app will download more movies, pointing to the download progress. Update - PHP Source added for network queries I used Retrofit/OkHttp in this project. Even the project included Retrofit, I will miss this part in this post as this post means RecyclerView loads more implementation. Full source code available in Github, you can download and use the code as you wish. RecyclerView is an extended and flexible version of ListView that effectively handles large datasets, supporting a limited number of views (such as the ViewHolder template used in ListView). You can use the same RecyclerView to present a list and grid type by installing RecyclerView.LayoutManager. As we implement RecyclerView load more I assume you have created the project and have enough knowledge on the layout of RecyclerView and the required dependencies. More to read here to get a basic understanding. OK let's start... Below is the final class of the RecyclerView adapter that I used in this project, look at the code, I'll explain the necessary blocks separately. - It has two types of viewing (movie, title, rating) and download progress) - MovieModel class data model that have a name, rating and type - boolean isLoading is used to customize download and download completed status. That will prevent the unnecessary back to the back loading more call. - You should call the custom method notifyDataChanged () to handle isLoading status - boolean isMoreDataAvailable is used to customize data server availablity status. That won't allow the app to request when the server no longer has data. Call the adapter.setMoreDataAcidable (false); notify the adapter that the server no longer has any data. - OnLoadMoreListener has a onLoadMore callback method that will be called whenever the scroll is reached at the bottom. Below is an example of the block code I used to call a load larger in the activity class. - In the main activity implements OnLoadMoreListener () to get a callback when the scroll reaches the bottom. - In the loadMore remote query method, I add an idea of the download progress to the bottom of the list by setting the MovieModel type to download. - In response to the data, the callback removes an additional load type from the MovieModel list and then adds the server's response to the current list. When the server returns an empty array of installs (false) to prevent further requests. That's it... For further help, show the full source code on GitHub. Github. Github.