



Today I would like to start new series about tips for RecyclerView. While you might be thinking, well, I'll just create a list and update it whenever new objects come in," FirebaseUI does exactly that for you so you don't have to write boilerplate code. If you're using simple POJO objects to hold data in your adapter, you can completely avoid writing the binding code in onBindViewHolder by using the Data Binding library.

Create the view holder for view items, connect the data source of the RecyclerView and handle the www.freshbytelabs.com view logic by creating a RecyclerView Adapter. If you use the provided code, there should be no errors whenever you are instructed to build the app. The aim is to keep the code concise and setup RecyclerView adapter without any boilerplate code.

Think about it. If you're making database queries constantly as the user scrolls through your RecyclerView, you're taking an unnecessary performance hit. Instead of accessing a ViewModel in this second activity, we send the input back to the MainActivity and insert the data into the Room database there.

Attach the adapter to the RecyclerView. The method that gets called to inflate the appropriate row is onBindViewHolder. Declare the RecyclerView in an activity layout and reference it in the activity Kotlin file. Each of the enclosing boxes (except for the SQLite database) represents a class that you will create.

An alternative is to use the findViewHolderForAdapterPosition(int position) method to locate the view (by way of its ViewHolder) for a specific position. Parts of the Android framework and libraries, such as RecyclerView , contain trace markers. Android UI does work in two phases - Record View#draw, on the UI thread, and DrawFrame on the RenderThread.