FREE eBook

LEARNING recyclerview

Free unaffiliated eBook created from
Stack Overflow contributors.

#recyclervie

w

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: recyclerview

It is an unofficial and free recyclerview ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official recyclerview.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with recyclerview

Remarks

This section provides an overview of what recyclerview is, and why a developer might want to use it.

It should also mention any large subjects within recyclerview, and link out to the related topics. Since the Documentation for recyclerview is new, you may need to create initial versions of those related topics.

Examples

Installation & Setup

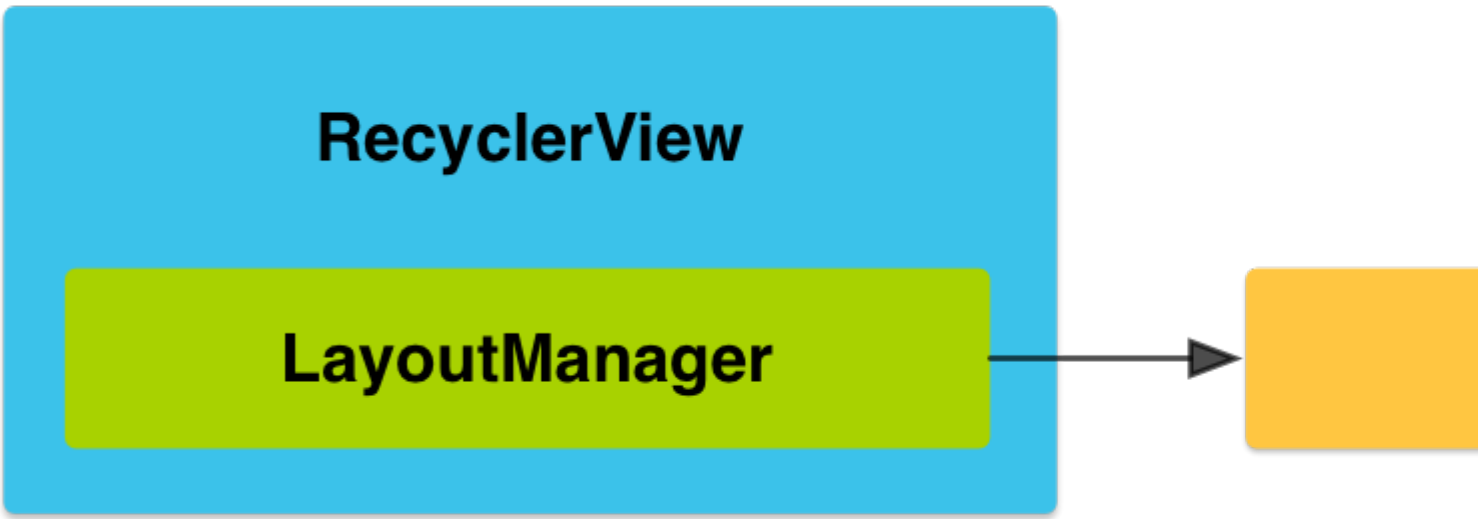
The display of elements in a list or grids is a very common pattern in mobile applications. The user sees a collection of items and can scroll through them. The collection of items can be a list, a grid or another structured representations of data.

The `RecyclerView` widget is a more advanced and flexible version of ListView. This widget is a container for displaying large data sets that can be scrolled very efficiently by maintaining a limited number of views. Use the RecyclerView widget when you have data collections whose elements change at runtime based on user action or network events.

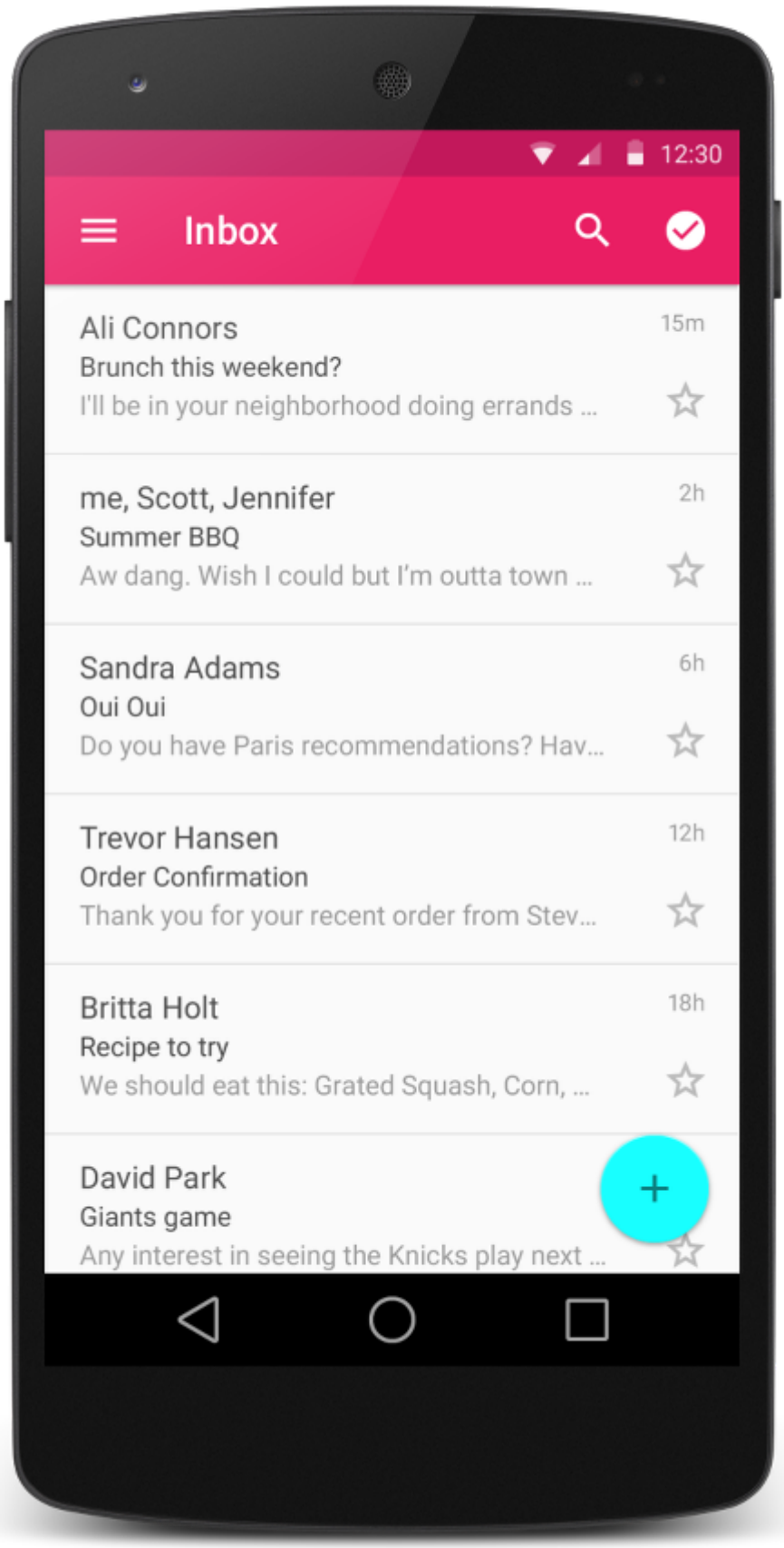
The `RecyclerView` class simplifies the display and handling of large data sets by providing:

- Layout managers for positioning items
- Default animations for common item operations, such as removal or addition of items

You also have the flexibility to define custom layout managers and animations for `RecyclerView` widgets.



To use the `RecyclerView` widget, you have to specify an adapter and a layout manager. To create an adapter, extend the `RecyclerView.Adapter` class. The details of the implementation depend on the specifics of your dataset and the type of views.



A layout manager positions item views inside a `RecyclerView` and determines when to reuse item views that are no longer visible to the user. To reuse (or recycle) a view, a layout manager may

ask the adapter to replace the contents of the view with a different element from the dataset. Recycling views in this manner improves performance by avoiding the creation of unnecessary views or performing expensive findViewById() lookups.

`RecyclerView` provides these built-in layout managers:

- LinearLayoutManager shows items in a vertical or horizontal scrolling list.
- GridLayoutManager shows items in a grid.
- StaggeredGridLayoutManager shows items in a staggered grid.

To create a custom layout manager, extend the RecyclerView.LayoutManager class.

Now to add a `RecyclerView` you need to add the following dependency in your `build.gradle`

```
dependencies {
    compile 'com.android.support:recyclerview-v7:23.4.0'
}
```

Now add a `RecyclerView` in your layout like this.

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:scrollbars="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Once you have added a `RecyclerView` widget to your layout, obtain a handle to the object, connect it to a layout manager, and attach an adapter for the data to be displayed.

```
public class MyActivity extends Activity {
    private RecyclerView mRecyclerView;
    private RecyclerView.Adapter mAdapter;
    private RecyclerView.LayoutManager mLayoutManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.my_activity);
        mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);

        // use this setting to improve performance if you know that changes
        // in content do not change the layout size of the RecyclerView
        mRecyclerView.setHasFixedSize(true);

        // use a linear layout manager
        mLayoutManager = new LinearLayoutManager(this);
        mRecyclerView.setLayoutManager(mLayoutManager);

        // specify an adapter (see also next example)
        mAdapter = new MyAdapter(myDataset);
        mRecyclerView.setAdapter(mAdapter);
    }
    ...
}
```

The adapter provides access to the items in your data set, creates views for items, and replaces the content of some of the views with new data items when the original item is no longer visible. The following code example shows a simple implementation for a data set that consists of an array of strings displayed using `TextView` widgets.

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {
    private String[] mDataset;

    // Provide a reference to the views for each data item
    // Complex data items may need more than one view per item, and
    // you provide access to all the views for a data item in a view holder
    public static class ViewHolder extends RecyclerView.ViewHolder {
        // each data item is just a string in this case
        public TextView mTextView;
        public ViewHolder(TextView v) {
            super(v);
            mTextView = v;
        }
    }

    // Provide a suitable constructor (depends on the kind of dataset)
    public MyAdapter(String[] myDataset) {
        mDataset = myDataset;
    }

    // Create new views (invoked by the layout manager)
    @Override
    public MyAdapter.ViewHolder onCreateViewHolder(ViewGroup parent,
                                                   int viewType) {
        // create a new view
        View v = LayoutInflater.from(parent.getContext())
                               .inflate(R.layout.my_text_view, parent, false);
        // set the view's size, margins, paddings and layout parameters
        ...
        ViewHolder vh = new ViewHolder(v);
        return vh;
    }

    // Replace the contents of a view (invoked by the layout manager)
    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        // - get element from your dataset at this position
        // - replace the contents of the view with that element
        holder.mTextView.setText(mDataset[position]);

    }

    // Return the size of your dataset (invoked by the layout manager)
    @Override
    public int getItemCount() {
        return mDataset.length;
    }
}
```

First add the support library to your project: Within the app's build.gradle file, add the dependency.

Read Getting started with recyclerview online:
https://riptutorial.com/recyclerview/topic/5426/getting-started-with-recyclerview

Credits

S. No	Chapters	Contributors
1	Getting started with recyclerview	Community, Princess Ruthie, Reaz Murshed