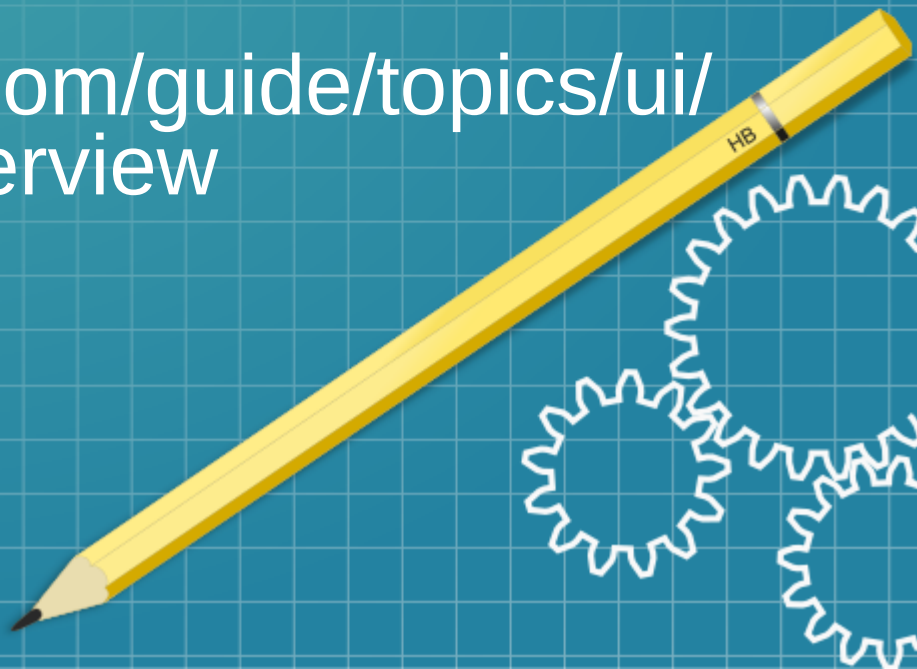
RecyclerView

https://developer.android.com/guide/topics/ui/layout/recyclerview



Cont ...

- If your app needs to display a scrolling list of elements based on large data sets (or data that frequently changes), you should use RecyclerView as described on this page.

Overview

- The RecyclerView widget is a more advanced and flexible version of ListView.
- In the RecyclerView model, several different components work together to display your data.
- The overall container for your user interface is a RecyclerView object that you add to your layout.
- The RecyclerView fills itself with views provided by a layout manager that you provide.

Cont ...

- You can use one of the standard layout managers (such as LinearLayoutManager or GridLayoutManager), or implement your own.
- The views in the list are represented by view holder objects.
- These objects are instances of a class you define by extending RecyclerView.ViewHolder.
- Each view holder is in charge of displaying a single item with a view.
- For example, if your list shows music collection, each view holder might represent a single album.

Cont ...

- The RecyclerView creates only as many view holders as are needed to display the on-screen portion of the dynamic content, plus a few extra.
- As the user scrolls through the list, the RecyclerView takes the off-screen views and rebinds them to the data which is scrolling onto the screen.

Cont ...

- The view holder objects are managed by an adapter, which you create by extending **RecyclerView.Adapter**
- The adapter creates view holders as needed.
- The adapter also binds the view holders to their data.
- It does this by assigning the view holder to a position, and calling the adapter's onBindViewHolder() method.
- That method uses the view holder's position to determine what the contents should be, based on its list position.

Cont ...

- This RecyclerView model does a lot of optimization work so you don't have to:
  - When the list is first populated, it creates and binds some view holders on either side of the list. For example, if the view is displaying list positions 0 through 9, the RecyclerView creates and binds those view holders, and might also create and bind the view holder for position 10. That way, if the user scrolls the list, the next element is ready to display.

Cont ...

- As the user scrolls the list, the RecyclerView creates new view holders as necessary. It also saves the view holders which have scrolled off-screen, so they can be reused. If the user switches the direction they were scrolling, the view holders which were scrolled off the screen can be brought right back.
- When the displayed items change, you can notify the adapter by calling an appropriate RecyclerView.Adapter.notify…() method. The adapter's built-in code then rebinds just the affected items.

Add RecyclerView to your layout

Now you can add the RecyclerView to your layout file. For example, the following layout uses RecyclerView as the only view for the whole layout:

- <?xml version="1.0" encoding="utf-8"?>

  <!-- A RecyclerView with some commonly used attributes -->

  <android.support.v7.widget.RecyclerView

  android:id="@+id/my_recycler_view"

  android:scrollbars="vertical"

  android:layout_width="match_parent"

  android:layout_height="match_parent"/>

Cont ...

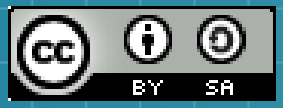
- Once you have added a RecyclerView widget to your layout, obtain a handle to the object, connect it to a layout manager, and attach an adapter for the data to be displayed:

Cont ...

```
@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);
}
```

Add a list adapter

- To feed all your data to the list, you must extend the RecyclerView.Adapter class.
- This object 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.



RecyclerView

https://developer.android.com/guide/topics/ui/layout/recyclerview

Cont ...

- If your app needs to display a scrolling list of elements based on large data sets (or data that frequently changes), you should use RecyclerView as described on this page.

Overview

- The RecyclerView widget is a more advanced and flexible version of ListView.
- In the RecyclerView model, several different components work together to display your data.
- The overall container for your user interface is a RecyclerView object that you add to your layout.
- The RecyclerView fills itself with views provided by a layout manager that you provide.

Cont ...

- You can use one of the standard layout managers (such as LinearLayoutManager or GridLayoutManager), or implement your own.
- The views in the list are represented by view holder objects.
- These objects are instances of a class you define by extending RecyclerView.ViewHolder.
- Each view holder is in charge of displaying a single item with a view.
- For example, if your list shows music collection, each view holder might represent a single album.

Cont ...

- The RecyclerView creates only as many view holders as are needed to display the on-screen portion of the dynamic content, plus a few extra.
- As the user scrolls through the list, the RecyclerView takes the off-screen views and rebinds them to the data which is scrolling onto the screen.

Cont ...

- The view holder objects are managed by an adapter, which you create by extending **RecyclerView.Adapter**
- The adapter creates view holders as needed.
- The adapter also binds the view holders to their data.
- It does this by assigning the view holder to a position, and calling the adapter's onBindViewHolder() method.
- That method uses the view holder's position to determine what the contents should be, based on its list position.

Cont ...

- This RecyclerView model does a lot of optimization work so you don't have to:
  - When the list is first populated, it creates and binds some view holders on either side of the list. For example, if the view is displaying list positions 0 through 9, the RecyclerView creates and binds those view holders, and might also create and bind the view holder for position 10. That way, if the user scrolls the list, the next element is ready to display.

Cont ...

- As the user scrolls the list, the RecyclerView creates new view holders as necessary. It also saves the view holders which have scrolled off-screen, so they can be reused. If the user switches the direction they were scrolling, the view holders which were scrolled off the screen can be brought right back.
- When the displayed items change, you can notify the adapter by calling an appropriate RecyclerView.Adapter.notify…() method. The adapter's built-in code then rebinds just the affected items.

Add RecyclerView to your layout

Now you can add the RecyclerView to your layout file. For example, the following layout uses RecyclerView as the only view for the whole layout:

- <?xml version="1.0" encoding="utf-8"?>

  <!-- A RecyclerView with some commonly used attributes -->

  <android.support.v7.widget.RecyclerView

  android:id="@+id/my_recycler_view"

  android:scrollbars="vertical"

  android:layout_width="match_parent"

  android:layout_height="match_parent"/>

Cont ...

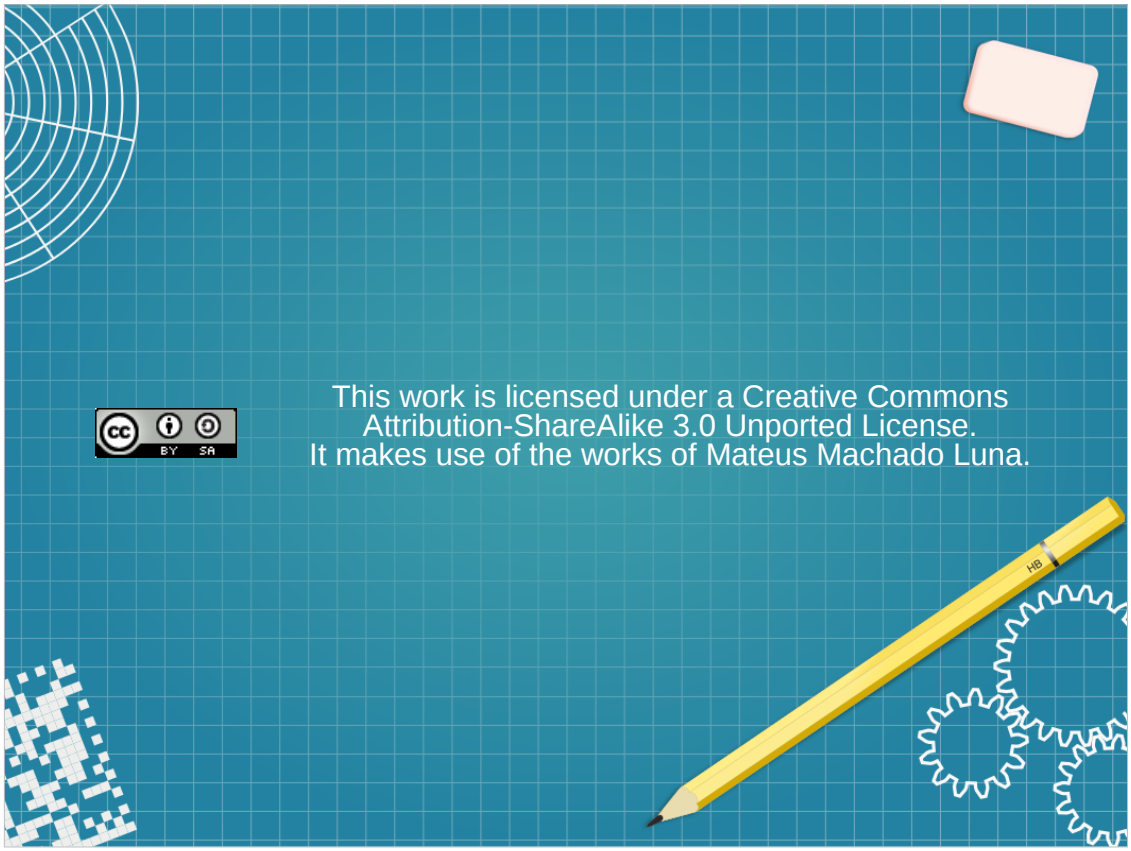
- Once you have added a RecyclerView widget to your layout, obtain a handle to the object, connect it to a layout manager, and attach an adapter for the data to be displayed:

Cont ...

```
@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);
}
```

Add a list adapter

- To feed all your data to the list, you must extend the RecyclerView.Adapter class.
- This object 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.


This work is licensed under a Creative Commons
Attribution-ShareAlike 3.0 Unported License.
It makes use of the works of Mateus Machado Luna.
BY SA
HB