SCHIBSTED

WORKSHOP ANDROID III

# WE NEED A BETTER SUBTITLE YET

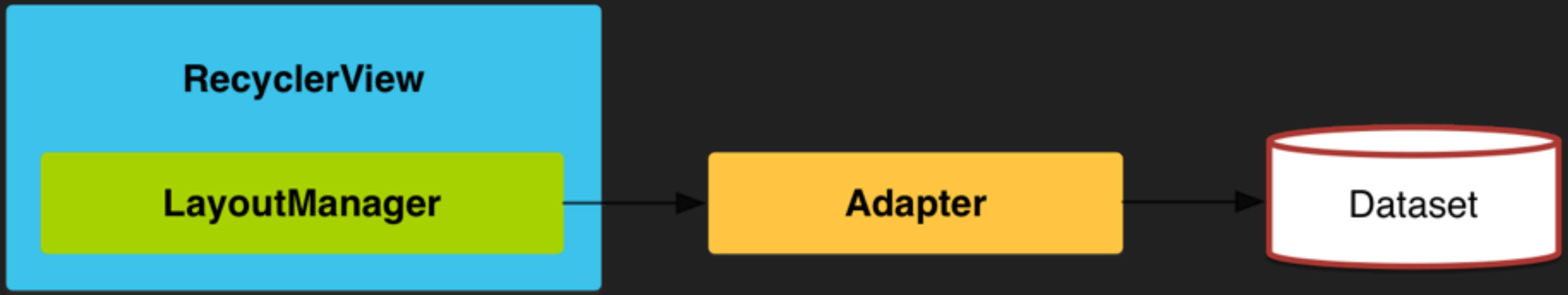
I'M WITH
ADMIRAL
STUPID

## PREVIOUSLY, ON WORKSHOP 2

- Permissions
- Activity Lifecycle
- Logcat
- Intents

## RECYCLERVIEW

## RECYCLERVIEW



## BUILD.GRADE (APP)

```
dependencies {
    ...
    compile 'com.android.support:recyclerview-v7:23.1.1'
}
```

## STEP 1: CREATE OBJECT MODEL

```
public class Offer {

    private String title;

    private String companyName;

    private String city;

….

}
```

## STEP 2: ADD RECYCLER VIEW IN LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <android.support.v7.widget.RecyclerView
       android:id="@+id/rv_main"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content">
    </android.support.v7.widget.RecyclerView>
</LinearLayout>
```

## STEP 3: ADD CUSTOM ROW LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/tv_offer_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

## STEP 4: CREATE RECYCLERVIEW.ADAPTER

- The adapter's role is to convert an object at a position into a list row item to be inserted.
- The adapter requires the existence of a "ViewHolder" object which describes and provides access to all the views within each item row.

## STEP 4: CREATE RECYCLERVIEW.ADAPTER

```
public class OfferAdapter extends RecyclerView.Adapter<OfferAdapter.ViewHolder> {
    public static class ViewHolder extends RecyclerView.ViewHolder {
        TextView offerTitleTV;
        TextView companyNameTV;
        TextView cityTV;
        public ViewHolder(View itemView) {
            super(itemView);
            offerTitleTV = ((TextView) itemView.findViewById(R.id.tv_offer_title));
            companyNameTV = ((TextView) itemView.findViewById(R.id.tv_company_name));
            cityTV = ((TextView) itemView.findViewById(R.id.tv_city));
        }
    }
}
```

## STEP 4: CREATE RECYCLERVIEW.ADAPTER

We need to begin filling in our adapter

```
….

private List<Offer> offers;

    public OfferAdapter(List<Offer> offers) {

        this.offers = offers;

    }

….
```

## STEP 4: CREATE RECYCLERVIEW.ADAPTER

Every adapter has three primary methods:

- onCreateViewHolder: to inflate the item layout and create the holder
- onBindViewHolder: to set the view attributes based on the data
- getItemCount: to determine the number of items

We need to implement all three to finish the adapter

## STEP 4: CREATE RECYCLERVIEW.ADAPTER

- onCreateViewHolder: to inflate the item layout and create the holder

```
public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        Context context = parent.getContext();
        LayoutInflater inflater = LayoutInflater.from(context);
        View offerView = inflater.inflate(R.layout.item_offer, parent, false);
        ViewHolder viewHolder = new ViewHolder(offerView);
        return viewHolder;
}
```

## STEP 4: CREATE RECYCLERVIEW.ADAPTER

- onBindViewHolder: to set the view attributes based on the data

```
public void onBindViewHolder(ViewHolder holder, int position) {
    Offer offer = offers.get(position);
    holder.titleTV.setText(offer.getOfferTitle());
    holder.companyNameTV.setText(offer.getCompanyName());
    holder.cityTV.setText(offer.getCity());
}
```

## STEP 5: BINDING THE ADAPTER TO THE RECYCLER VIEW

```
RecyclerView rv = (RecyclerView)
findViewById(R.id.rv_main);

OfferAdapter adapter = new OfferAdapter(offers);

rv.setAdapter(adapter);

rv.setLayoutManager(new LinearLayoutManager(this));
```

## NOTIFYING THE ADAPTER

There are many method available to use when notifying the adapter of different changes:

- notifyItemChanged(int pos): Notify that item at position has changed.
- notifyItemInserted(int pos): Notify that item reflected at position has been newly inserted.
- notifyItemRemoved(int pos): Notify that items previously located at position has been removed from the data set.
- notifyDataSetChanged(): Notify that the dataset has changed. Use only as last resort.

## ANIMATIONS

```
dependencies {
    compile 'jp.wasabeef:recyclerview-animators:2.2.0'
}

RecyclerView.ItemAnimator itemAnimator = new
ScaleInBottomAnimator();

rv.setItemAnimator(itemAnimator);
```

# THAT'S ALL, FOLKS!

## REFERENCE

- https://developer.android.com/reference/android/support/v7/widget/RecyclerView.html
- http://www.grokkingandroid.com/first-glance-androids-recyclerview/
- http://www.grokkingandroid.com/statelistdrawables-for-recyclerview-selection/
- http://www.bignerdranch.com/blog/recyclerview-part-1-fundamentals-for-listview-experts/
- https://developer.android.com/training/material/lists-cards.html
- http://antonioleiva.com/recyclerview/
- https://code.tutsplus.com/tutorials/getting-started-with-recyclerview-and-cardview-on-android--cms-23465
- https://code.tutsplus.com/tutorials/introduction-to-the-new-lollipop-activity-transitions--cms-23711

## SPECIAL THANKS TO

- Infojobs' Transformers Team
- Schibsted Spain
- Mom and dad