Android
The Rocket App
Getting data from the internet
List Views
The Rocket Database App

• Every rocket launch from 1957 - June 2020

• Goal
  • provide a user friendly portal into querying this database
  • Or at least a subset of the database

• Idea
  • allow users to look at the data starting from different slicing approaches
  • I have only implemented two slices

Tables in rocket

<table>
<thead>
<tr>
<th>launch</th>
</tr>
</thead>
<tbody>
<tr>
<td>site</td>
</tr>
<tr>
<td>vehicle</td>
</tr>
</tbody>
</table>

describe launch;

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>varchar(10)</td>
<td>NO</td>
<td>PRI</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>JD</td>
<td>varchar(12)</td>
<td>NO</td>
<td>PRI</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>date</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Vehicle</td>
<td>varchar(20)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Flight</td>
<td>varchar(20)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Mission</td>
<td>varchar(30)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>LaunchSite</td>
<td>varchar(10)</td>
<td>YES</td>
<td>MUL</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>LaunchPad</td>
<td>varchar(10)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Apogee</td>
<td>mediumint(9)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>varchar(10)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>

describe vehicle;

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>varchar(20)</td>
<td>NO</td>
<td>PRI</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>FamilyName</td>
<td>varchar(20)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>MainStageName</td>
<td>varchar(20)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Manufacturer</td>
<td>varchar(10)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>VehicleVariant</td>
<td>varchar(20)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td>float</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Diameter</td>
<td>float</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>LaunchMass</td>
<td>float</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>LEOPayload</td>
<td>float</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>GTOPayload</td>
<td>float</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>Thrust</td>
<td>float</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>MaxApogee</td>
<td>float</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>
This feels to me like a natural fit for fragment navigation, and for use of the android system back button.
Getting data from internet

Just write some standard Java code to do this
see “webInvokeJSON” in starter

NetworkSecurityConfig: No Network Security Config specified,
using platform default
E/THIS: android.os.NetworkOnMainThreadException
android.os.NetworkOnMainThreadException

Into Android Manifest:

```xml
<uses-permission android:name="android.permission.INTERNET" />
android:usesCleartextTraffic="true"
```

With permissions fixed:

E/THIS: android.os.NetworkOnMainThreadException
android.os.NetworkOnMainThreadException
android.os.NetworkOnMainThreadException
at android.os.StrictMode$AndroidBlockGuardPolicy.onNetwork(StrictMode.java:1450)

“Uses-permission” is fairly standard in Android. Any time you use a subsystem, you have to ask for permission. Allows the user to say no — for instance, “no the app can’t use the camera”.

Allows http connections. Android would far prefer https

DBs can be set up to allow direct connections. The DB on comet does not allow connections except from localhost. So, need to write some PHP. PHP or direct to DB, you still need this
Main Thread??

• ""When an application is launched in Android, it creates the first thread of execution, known as the “main” thread. The main thread is responsible for dispatching events to the appropriate user interface components as well as communicating with components from the Android UI toolkit.””

• So, the main thread is what users interact with and see

• The goal is to always have the main thread be responsive to the user. Even if sometimes that means just telling the user “wait a second”

• Network operations are NOT allowed on the main thread
Running away from the Main Thread

- Again, just plain Java
  - Create a class that implements Runnable
  - Execute that class in a separate thread

```java
private class GetLaunchersRunner implements Runnable {
    @Override
    public void run() {
        final String[][] ee = {
            {"ActionPageSize", "1"},
            {"ActionPageNo", "1"},
        };
        String res = webInvokeJSON("http://comet.cs.brynmawr.edu/~gtowell/380/rocket/sounders.php", ee);
        Log.i("THISS", res);
    }
}
```

Executor exec = new Executor() {
  @Override
  public void execute(Runnable command) {
    new Thread(command).start();
  }
};

try {
  exec.execute(new GetLaunchersRunner());
} catch (RejectedExecutionException | NullPointerException exception) {
  Log.e("THISS", "Thread problem", exception);
}

This WORKS!!!
Wait a second (or 10) or for something to complete

```java
final int delay = 10;
final RelativeLayout rl = findViewById(R.id.mainlayout);
final TextView tv = new TextView(this);
tv.setText(""+delay);
tv.setGravity(Gravity.CENTER);
tv.setBackgroundColor(Color.MAGENTA);
tv.setTextSize(128);
tv.setTypeface(null, Typeface.BOLD);
rl.addView(tv, lpp);
tv.setRotation(0);
tv.animate().rotation(2880).setDuration(delay*1000).setListener(new AnimatorListenerAdapter() {
    @Override
    public void onAnimationEnd(Animator animation) {
        super.onAnimationEnd(animation);
    }
});

// silly simulator for a process that takes 10 seconds
// updating the screen along the way
Executor exec = new Executor() {
    @Override
    public void execute(Runnable command) {
        new Thread(command).start();
    }
};
try {
    exec.execute(new Runnable() {
        @Override
        public void run() {
            for (int i=delay; i>0; i--) {
                try {
                    Thread.sleep(1000);
                } catch (Exception ee) {
                }
                final int j=i;
                new Handler(Looper.getMainLooper()).post(new Runnable() {
                    @Override
                    public void run() {
                        if (j>0)
                            tv.setText(""+j);
                        else
                            rl.removeView(tv);
                    }
                });
            }
        }
    });
} catch (RejectedExecutionException | NullPointerException exception) {
    Log.e("THISS", "Thread problem", exception);
}
```
private class GetLaunchersRunnerV2 implements Runnable {
    @Override
    public void run() {
        final String[][] ee = {
            {"ActionPageNo", "1"},
        };
        String res = webInvokeJSON("http://comet.cs.brynmawr.edu/~gtowell/380/rocket/sounders.php", ee);
        TextView tv = (TextView)MainActivity.this.findViewById(R.id.textview);
        tv.setText(res);
    }
}

E/AndroidRuntime: FATAL EXCEPTION: Thread-2
Process: edu.brynmawr.starter, PID: 31639
android.view.ViewRootImpl$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views.
    at android.view.ViewRootImpl.checkThread(ViewRootImpl.java:7313)
    at android.view.ViewRootImpl.requestLayout(ViewRootImpl.java:1161)

• Only the UI thread can update the UI thread!!
UI updates must be from the UI thread

- So you need to get back there.

```java
private class GetLaunchersRunnerV3 implements Runnable {
    @Override
    public void run() {
        final String[][] ee = {
            {"ActionPageNo", "1"},
        };
        String res = webInvokeJSON("http://comet.cs.brynmawr.edu/~gtowell/380/rocket/sounders.php", ee);
        new Handler(Looper.getMainLooper()).post(new Runnable() {
            @Override
            public void run() {
                TextView tv = (TextView)MainActivity.this.findViewById(R.id.textview);
                tv.setText(res);
            }
        });
    }
}
```

This gets you to the main (UI) thread and executes code there.
The text is too big!!

**Scrolling**

- When everything does not fit, wrap it in a **ScrollView**.
- Usually a good idea to do so because you can never be sure that someone does not have a tiny screen.

```xml
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fadeScrollbars="false"
        tools:layout_editor_absoluteX="200dp"
        tools:layout_editor_absoluteY="299dp">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">
            <TextView
                android:id="@+id/textview"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello World!"/>
        </LinearLayout>
    </ScrollView>
</androidx.constraintlayout.widget.ConstraintLayout>
```
The interface still sucks

ListView

• Could improved the text by inserting \n\nappropriately
  • maybe get rid of a lot of the JSON formatting also
• But it is still just dead text.
• How to make it possible to sect rows and drill down

• Android ListView / RecyclerView
List Views

• When things might take a while, you need to tell users that things are happening
• Simplest ListView are for arrays of Strings
• grey: add the list view to the interface
• red: set up the listview adapter
• green: click on row to drill down in database

```java
private void gotData0(final ArrayList<JSONObject> aList) {
    mainLayout.removeAllViews();
    ArrayList<String> aString = new ArrayList<>();
    for (JSONObject jo : aList)
        aString.add(jo.toString());
    ListView theLv = new ListView(getContext());
    mainLayout.addView(theLv, new
    ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
    ViewGroup.LayoutParams.MATCH_PARENT, new
    ArrayAdapter<String>(getContext(), android.R.layout.simple_list_item_1,
    android.R.id.text1, aString));
    theLv.setAdapter(adapter);
    theLv.setOnItemClickListener(new
    AdapterView.OnItemClickListener() {
        public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {
            JSONObject o = aList.get(position);
            Toast.makeText(getContext(),
            o.toString(), Toast.LENGTH_SHORT).show();
            FragmentManager fragmentManager = ((AppCompatActivity)getContext()).getSupportFragmentManager();
            FragmentTransaction transaction = fragmentManager.beginTransaction();
            transaction.replace(MainActivity.MAIN_ACTIVITY_ID,
            new FragmentLaunches(o), null); transaction.addToBackStack(null);
            transaction.commit();
        }
    });
}
```
More on ListViews

• 2 parts: the view and the adapter
  • The view specifies where it is and what the container looks like
  • The adapter specifies the list items and how they are rendered
  • So the code is:
    • create the ListView
    • create the adapter
    • attach the adapter to the listview

```java
ListView theLv = new ListView(getContext());
mainLayout.addView(theLv, new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.MATCH_PARENT));

ArrayAdapter<String> adapter = new ArrayAdapter<String>(getContext(), android.R.layout.simple_list_item_1, android.R.id.text1, aString);
theLv.setAdapter(adapter);
```

This creates the most basic adapter which only works for array (or arraylist) of strings
Improving on the Basic ListView

• Most improvements come in the form of customizing the adapter
• There are several adapter types; in a decade I have only used ArrayAdapter
• Here, rather than using the basic adapter, I created a custom one so I could use an array of JSONObject

ArrayAdapter<JSONObject> aa = new ArrayAdapter<JSONObject>(getContext(), 0, aList) {
    @NonNull
    @SuppressLint("InflateParams")
    @Override
    public View getView(int position, View convertView, @NonNull ViewGroup parent) {
        View view = convertView;
        if (view == null) {
            view = new LinearLayout(getContext());
        }
        JSONObject msg = this.getItem(position);
        LinearLayout ll = (LinearLayout) view;
        ll.removeAllViews();
        ll.setOrientation(LinearLayout.HORIZONTAL);
        ll.setLayoutParams(new LinearLayout.LayoutParams(0,
            ViewGroup.LayoutParams.WRAP_CONTENT,
            0,
            ViewGroup.LayoutParams.WRAP_CONTENT,
            1));
        return view;
    }
};
Comparing

• Ignoring the background color there is little difference.
• So a bunch of work, not much noise
Going further with Custom

- Note that the textview is put in a linearlayout.
- We can put ANYTHING into that.
- Pictures? (If I had them)
- buttons (or other active elements)
- More, we can render the information we have in a more user friendly way
- Put data in columns with column dividers

```
ArrayAdapter<JSONObject> aa = new ArrayAdapter<JSONObject>(getContext(), 0, aList) {
    @NonNull
    @Override
    public View getCount() {
        // Unchanged code not shown
        TextView tv = newTextView(getContext());
        tv.setPadding((int)(3*density), 0, 0);
        tv.setText(msg.optString("name"));
        tv.setGravity(Gravity.START | Gravity.CENTER_VERTICAL);
        ll.addView(tv, newLinearLayout.LayoutParams(0,
            ViewGroup.LayoutParams.MATCH_PARENT, 3));
    }
    View v = new View(getContext());
    v.setBackgroundColor(ContextCompat.getColor(getContext(),
        R.color.purple_500));
    ll.addView(v, newLinearLayout.LayoutParams((int)(0.5*density),
        ViewGroup.LayoutParams.WRAP_CONTENT, 3));
    TextView m = new TextView(getContext());
    m.setGravity(Gravity.CENTER);
    SpannableString mySpannableString = new SpannableString("vehicle details");
    mySpannableString.setSpan(new UnderlineSpan(), 0, mySpannableString.length(), 0);
    m.setText(mySpannableString);
    m.setTextSize(10);
    m.setTextColor(Color.rgb(0, 122, 255));
    ll.addView(m, newLinearLayout.LayoutParams(0, ViewGroup.LayoutParams.WRAP_CONTENT, 3));
    m.setOnClickListener(new View.OnClickListener{
        @Override
        public void onClick(View view) {
            Log.i("THISS", "launcher details");
            Toast.makeText(getContext(), "Launcher details", Toast.LENGTH_SHORT).show();
            FragmentManager fragmentManager = ((AppCompatActivity)getContext()).getSupportFragmentManager();
            FragmentTransaction transaction = fragmentManager.beginTransaction();
            transaction.replace(MainActivity.MAIN_ACTIVITY_ID, new FragmentLauncherDetails(msg),
                null);
            transaction.addToBackStack(null);
            transaction.commit();
        }
    });
};
```
More Customization

• Columns are cool, but they need column headers
• Column headers are cool, but they need sorting!

```java
private View headerRow(ArrayList<JSONObject> aList, ArrayAdapter<JSONObject> lv) {
    float density = getResources().getDisplayMetrics().density;
    LinearLayout ll = new LinearLayout(getContext());
    ll.setBackgroundColor(ContextCompat.getColor(getContext(), R.color.purple_200));
    ll.setOrientation(LinearLayout.HORIZONTAL);
    ll.setLayoutParams(new AbsListView.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, (int)(40*density)));
    TextView tv = new TextView(getContext());
    tv.setPadding((int)(3*density), 0, 0, 0);
    tv.setText("name");
    tv.setGravity(Gravity.START | Gravity.CENTER_VERTICAL);
    ll.addView(tv, new LinearLayout.LayoutParams(0, ViewGroup.LayoutParams.MATCH_PARENT, 3));
    tv.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            reSort(aList, "name");
            lv.notifyDataSetChanged();
        }
    });
    View v = new View(getContext());
    v.setBackgroundColor(ContextCompat.getColor(getContext(), R.color.purple_500));
    ll.addView(v, new LinearLayout.LayoutParams((int)(0.5*density), ViewGroup.LayoutParams.MATCH_PARENT, 0));
}
```

This goes into the method that creates the listview

Create a new view, actually a LinearLayout, that contains column headers. These need to be laid out exactly as in data.

Click on a column header to sort the java using standard Java, in-place sorting (Not shown). Once the data is sorted, tell the listview that the data has changed.
Before and After

<table>
<thead>
<tr>
<th>name</th>
<th>count</th>
<th>first</th>
<th>recent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ariane 5G</td>
<td>16</td>
<td>1996-06-04</td>
<td>2003-09-27</td>
</tr>
<tr>
<td>Atlas F</td>
<td>69</td>
<td>1961-08-09</td>
<td>1981-06-23</td>
</tr>
<tr>
<td>Atlas II</td>
<td>23</td>
<td>1992-06-10</td>
<td>2002-12-05</td>
</tr>
<tr>
<td>Atlas IIAS</td>
<td>30</td>
<td>1993-12-16</td>
<td>2004-08-31</td>
</tr>
<tr>
<td>Chang Zheng 2F</td>
<td>35</td>
<td>1999-05-10</td>
<td>2019-12-20</td>
</tr>
</tbody>
</table>
ListView and RecyclerView

• Even after customization, a list view still looks like a list.
  • Sometimes this is exactly what you want
    • I would argue that this is correct for the first Rocket screen
• RecyclerView gives you a lot more
  • Horizontal scrolling of items
  • Layout items in grids
  • unequal sized items
  • etc.
• You can make a list view using a recycler, but it is harder.