

You're Going To Build An Android App

at

CodeMash!



Obligatory Introduction



@daveshah



@willkesling



@mattfousek

We are NOT EXPERTS

(I declare) expertise in no single language or methodology and (am) immediately suspicious of anyone who declares such expertise.



@docondev

@daveshah

@mattfousek

What to Expect

Section 1 - Android, Android Studio, and the Play Store

Section 2 - Activities, Fragments, Lifecycles, and more

Section 3 - Data, Permissions, Tasks

Lunch

Section 4 - Adapters, Custom Views, and More

Section 5 - Persistence, Filtering, and more on Views

Section 6 - Intents, both Explicit and Implicit

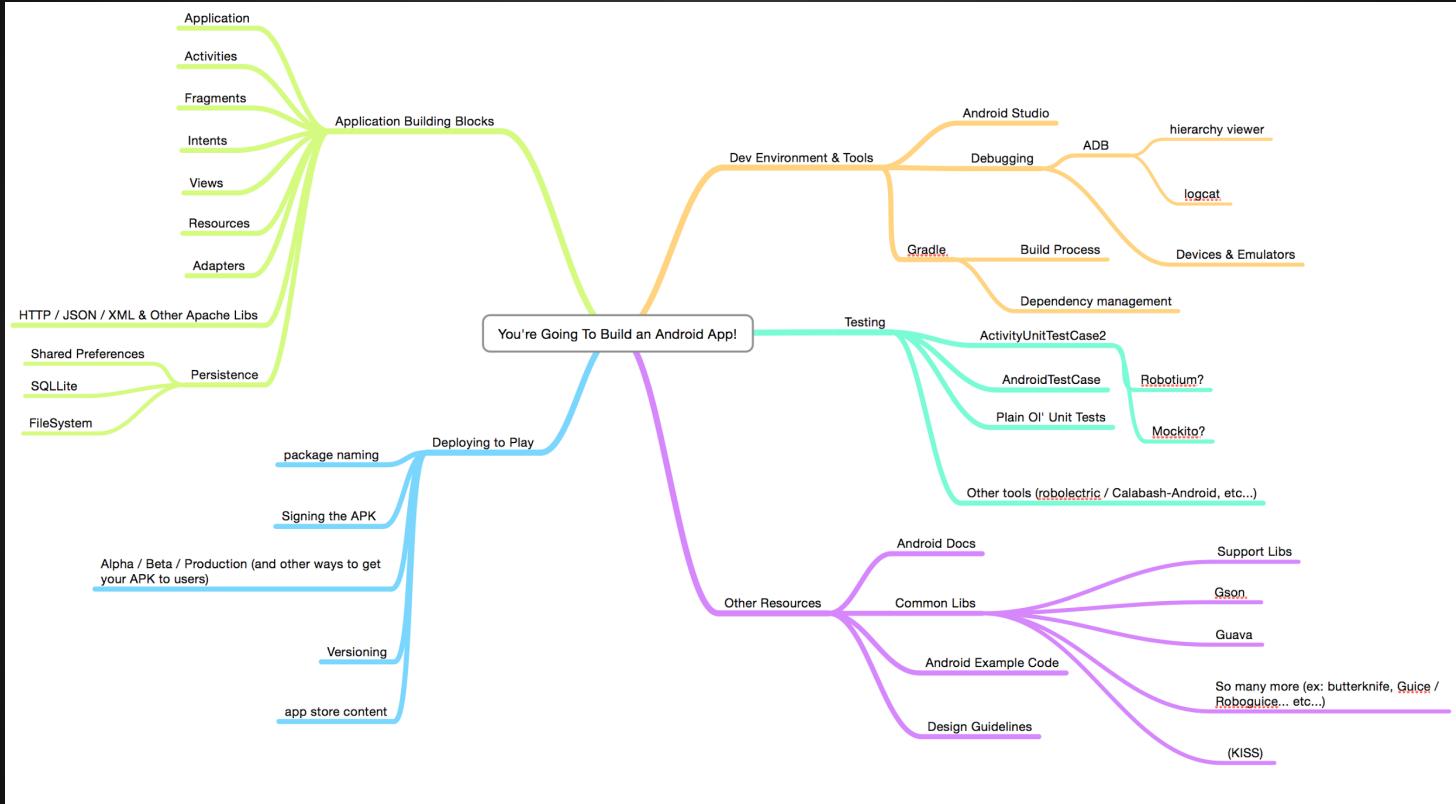
Section 7 - Be Clean, Resources, and a few Other Things

Section 8 - Challenges

Section 1

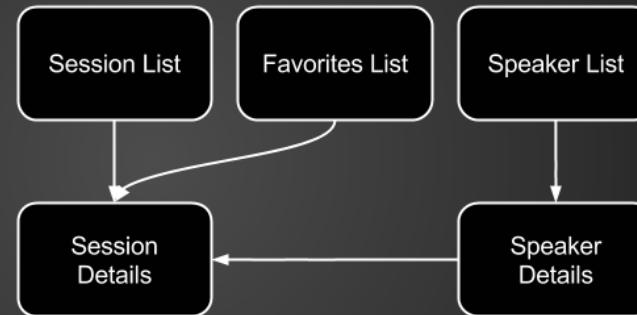
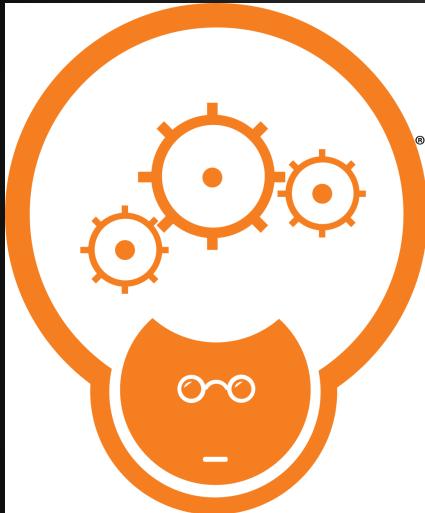
What are you doing, Dave?

What you're in for

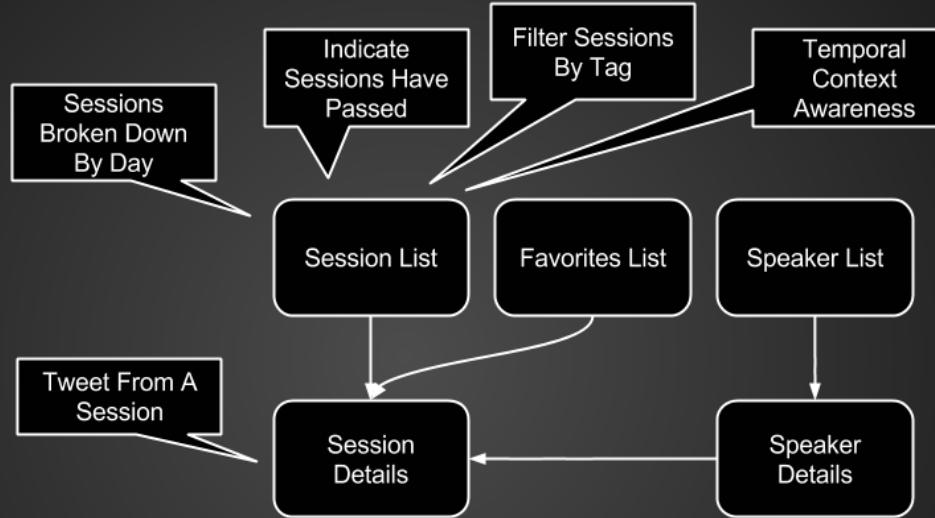
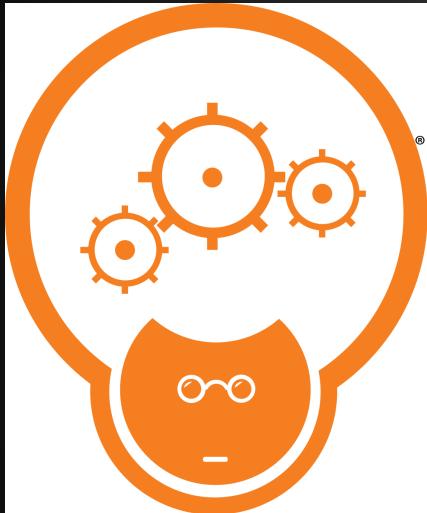


What we wish we had time to cover
(and we forgot a few too)

Where we're heading



Where we're heading





Sacrifices were made.

What you should have already

- Java JDK 7
- Android Studio
- Android SDK + extras
- An Android Emulator or Android Device
- (and it will help to have Git)

**Let's get right to it and crack open
Android Studio**



Quick Start

-  Start a new Android Studio project
-  Open an existing Android Studio project
-  Import an Android code sample
-  VCS Check out project from Version Control
-  Import Non-Android Studio project
-  Configure
-  Docs and How-Tos



New Project

Android Studio

Select the form factors your app will run on

Different platforms require separate SDKs

Phone and Tablet

Minimum SDK

API 15: Android 4.0.3 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available. By targeting API 15 and later, your app will run on approximately **87.9%** of the devices that are active on the Google Play Store. [Help me choose.](#)

TV

Minimum SDK

API 21: Android 5.0 (Lollipop)

Wear

Minimum SDK

API 21: Android 5.0 (Lollipop)

Glass (Not Installed)

Minimum SDK

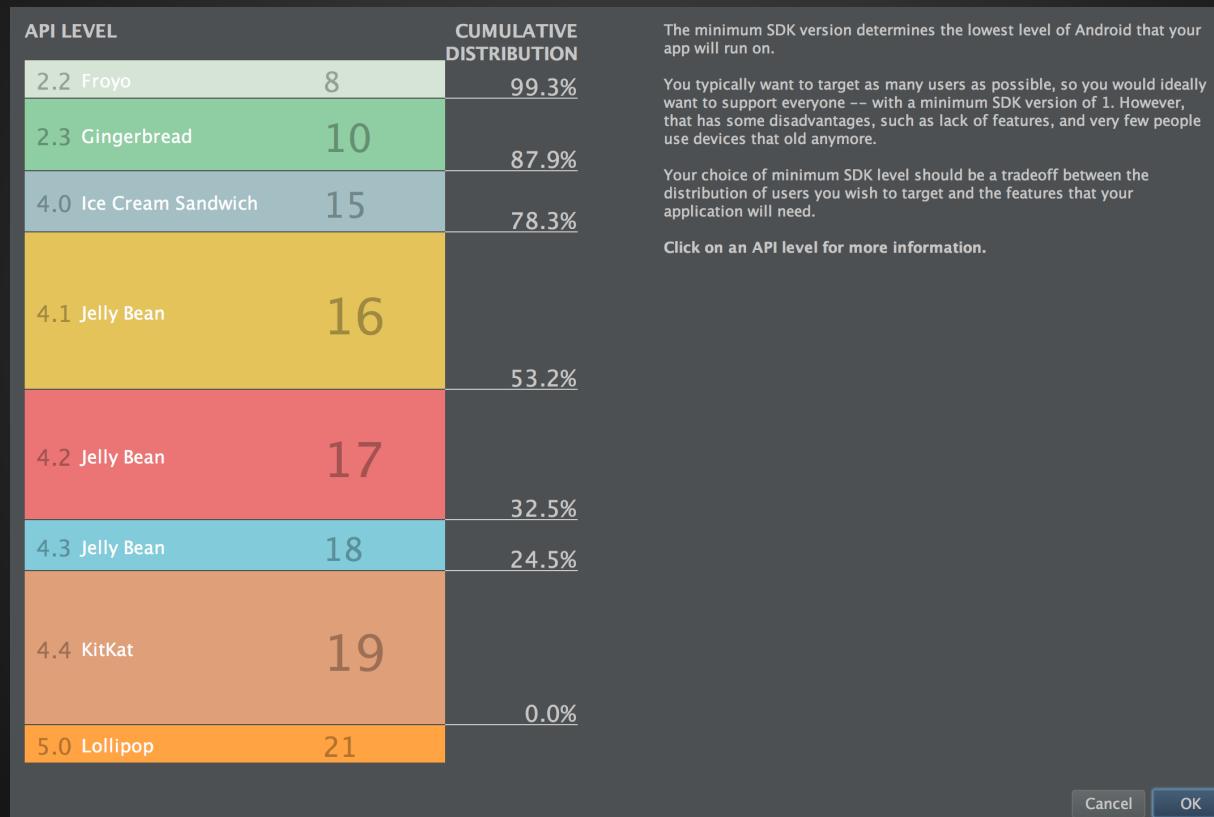
Cancel

Previous

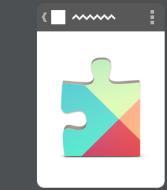
Next

Finish

A quick note about fragmentation...



Add an activity to Mobile



Google Play Services Activity



Login Activity



Master/Detail Flow



Navigation Drawer Activity



Settings Activity



Tabbed Activity

[Cancel](#) [Previous](#) [Next](#) [Finish](#)

What's an Activity?... (we'll get to that soon!)



Choose options for your new file



Creates a new blank activity, with an action bar and navigational elements such as tabs or horizontal swipe.

Activity Name:

Layout Name:

Fragment Layout Name:

Title:

Menu Resource Name:

Navigation Style:

Tabbed Activity

Additional features to include, such as a fragment, swipe views, or a navigation drawer

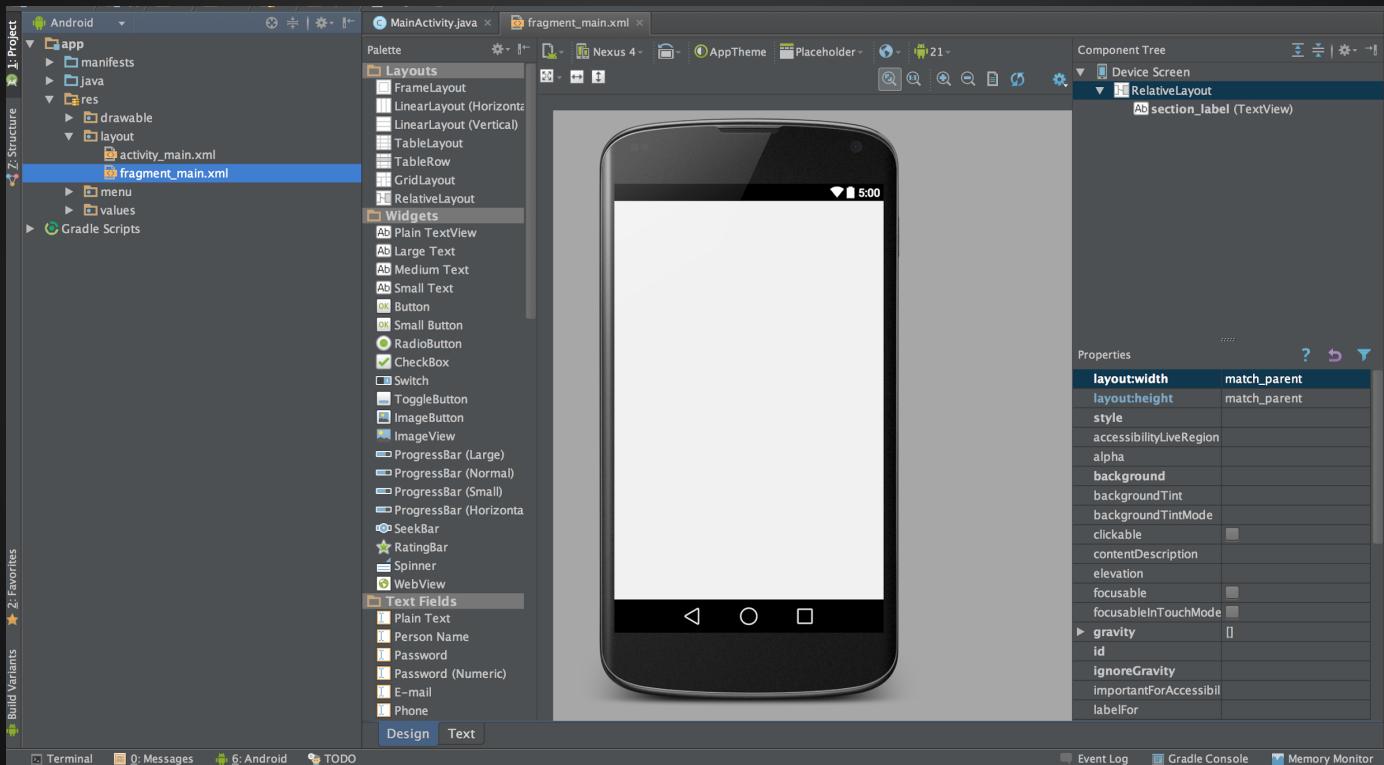
Cancel

Previous

Next

Finish

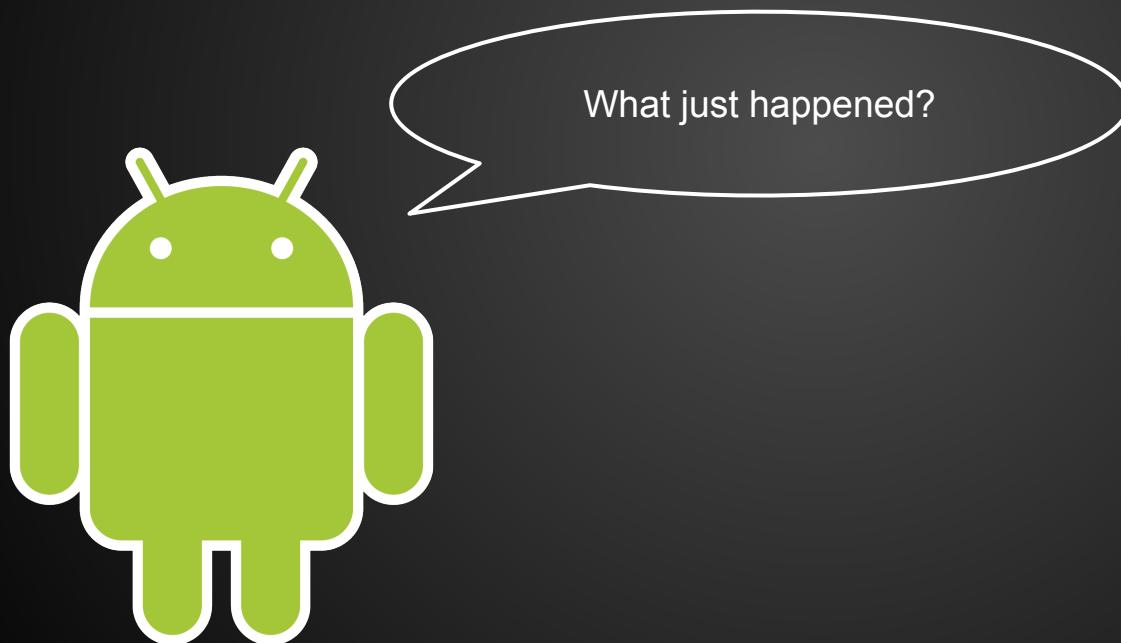
And if all went well...



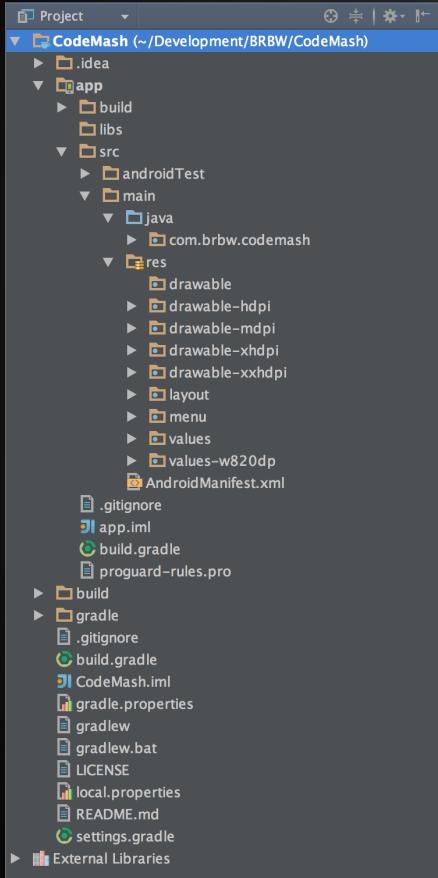
Bring Forth the Android Emulators or Devices!



So hopefully you got up and running...

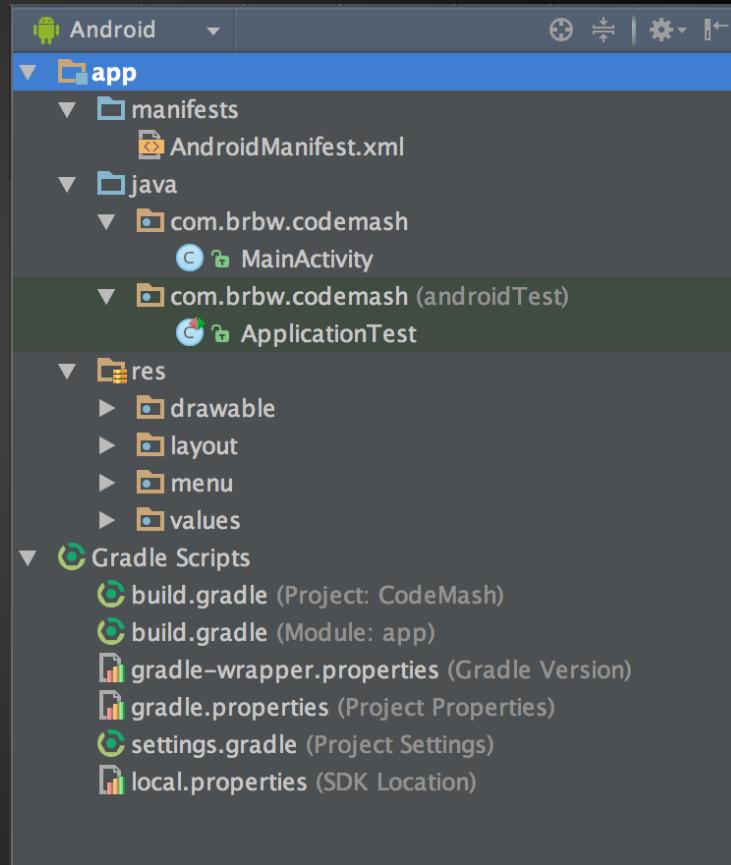


What Android Gave you out of the box



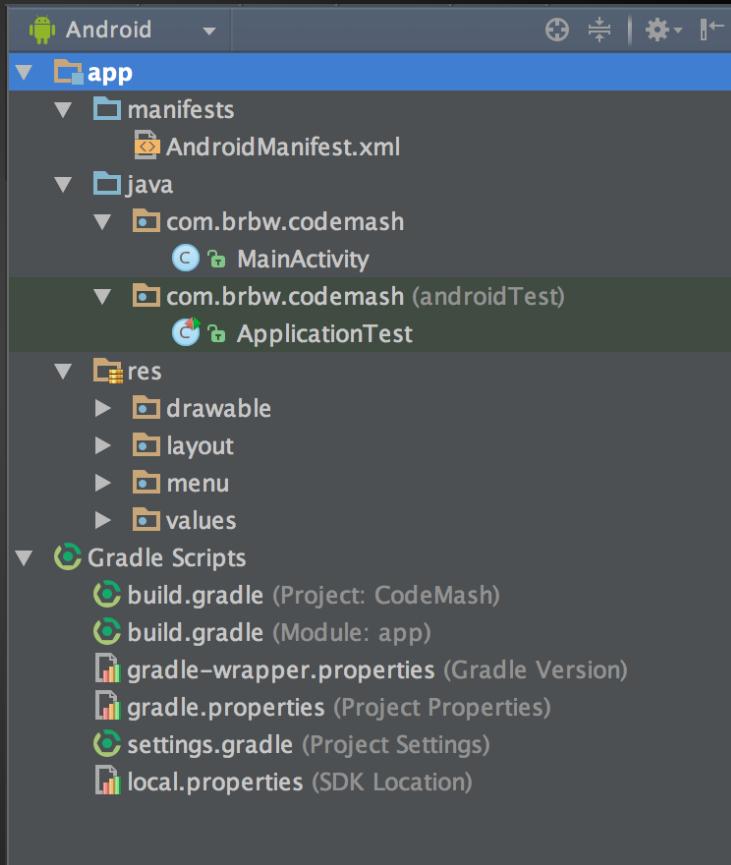
The whole enchilada

The stuff you'll care most about

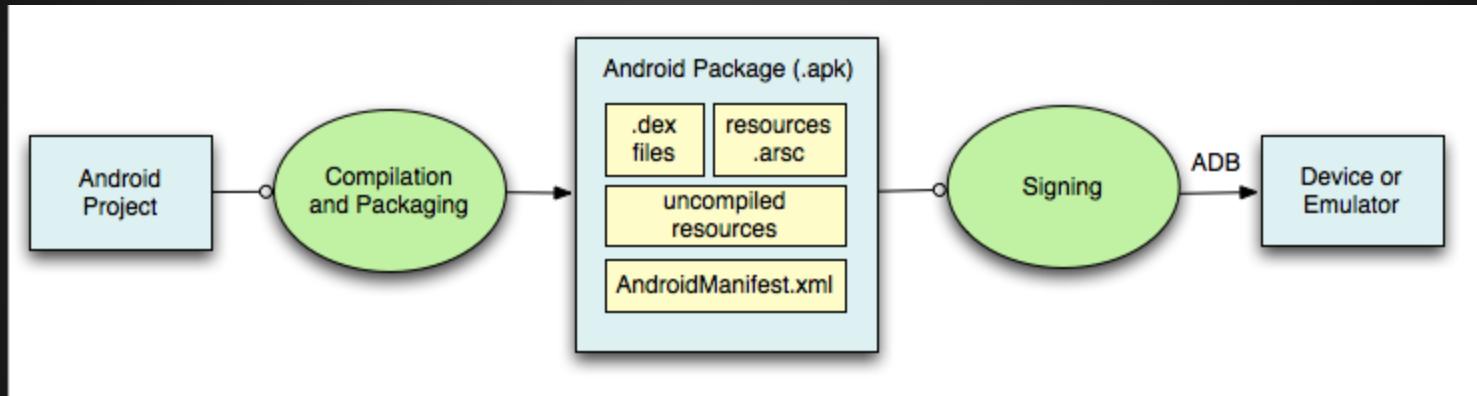


What Android Gave you out of the box

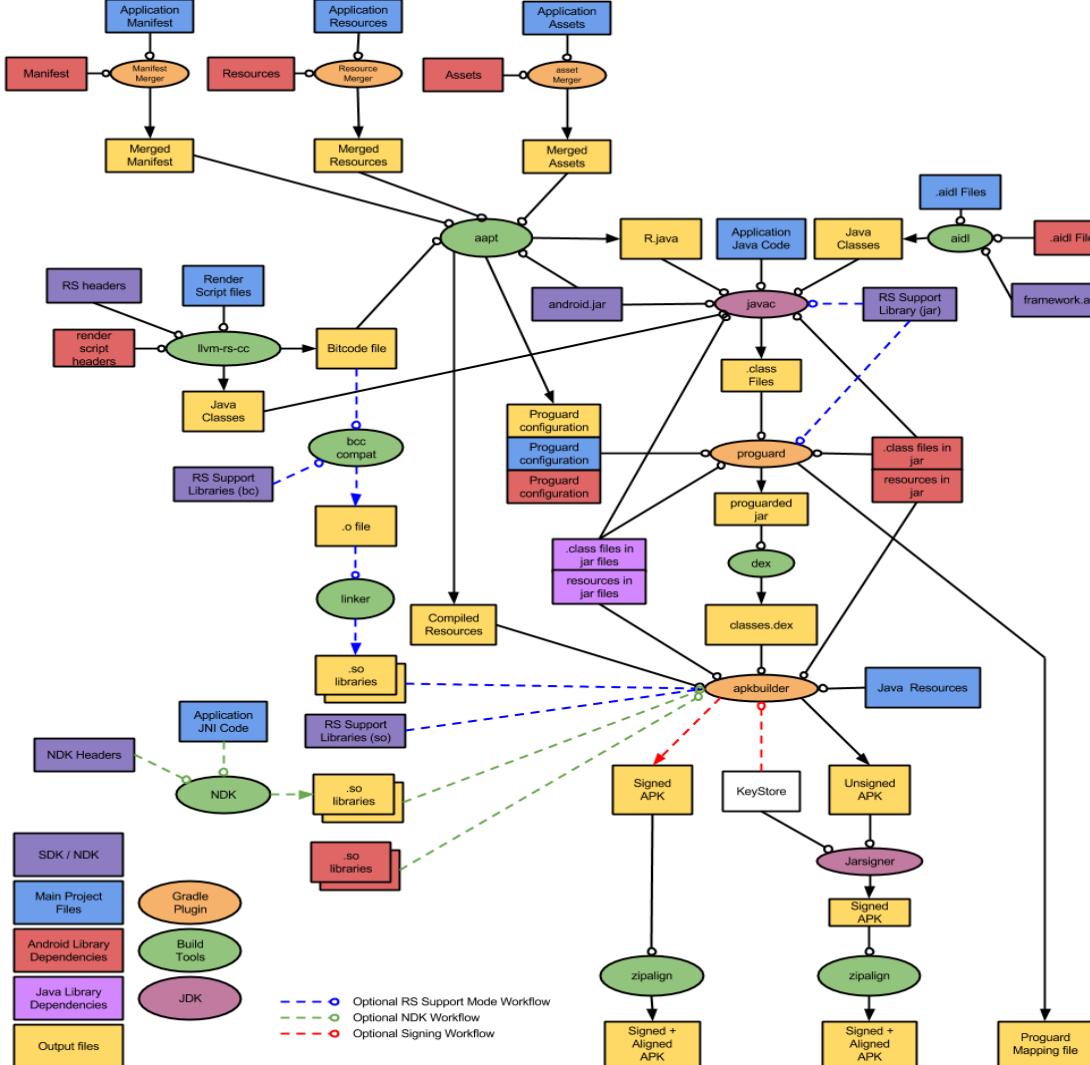
- app/src/main/AndroidManifest.xml
- app/src/main/java/*
- app/src/androidTest/java/*
- app/src/main/res/*
- app/build.gradle



Building and Deploying (simplified)



source: <http://developer.android.com/tools/building/index.html>



Building and Deploying (not simplified)

source: <http://tools.android.com/tech-docs/new-build-system/build-workflow>

So what's next?...

To the Play Store!

To the Play Store!

If you haven't signed up already and want to get your app to the public, lets do that now...

<https://play.google.com/apps/publish/>

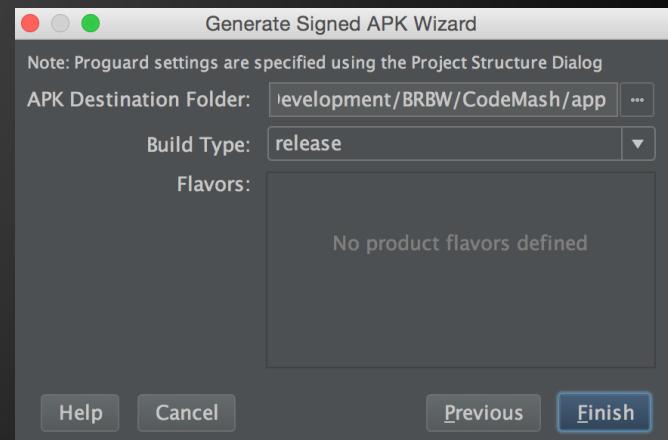
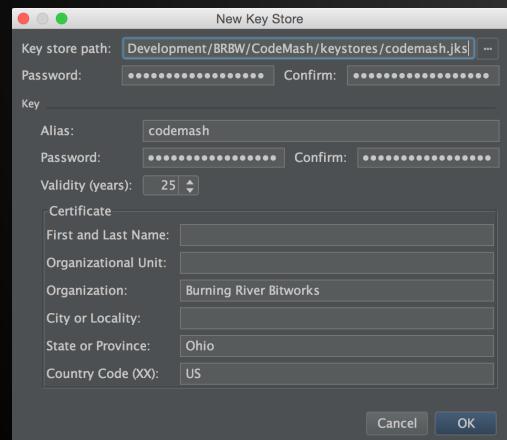
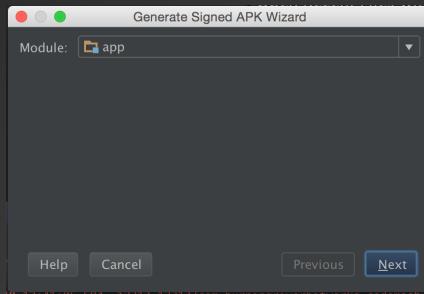
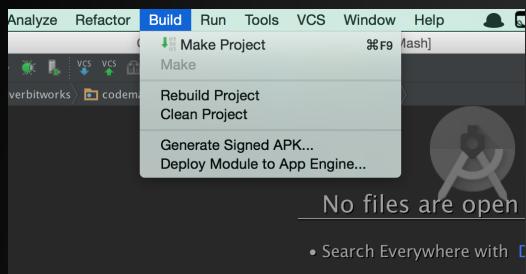
Our production checklist:

- A signed APK
- Store Listing
 - Title
 - Short & Full Descriptions
 - Screenshots *we can take those in a bit
 - High-res icon (512x512 32 bit png)
 - Feature Graphic (1024x500 jpeg or png)
 - and a few other items...

Let's deploy what we have and discuss options for the others...

- A signed APK
- Store Listing
 - Title
 - Short & Full Descriptions
 - Screenshots *we can take those in a bit
 - High-res icon (512x512 32 bit png)
 - Feature Graphic (1024x500 jpeg or png)
 - and a few other items...

Signing your app



Once you publish, don't lose your password!

(you can change it if you know it, but if you don't users will have to re-install)





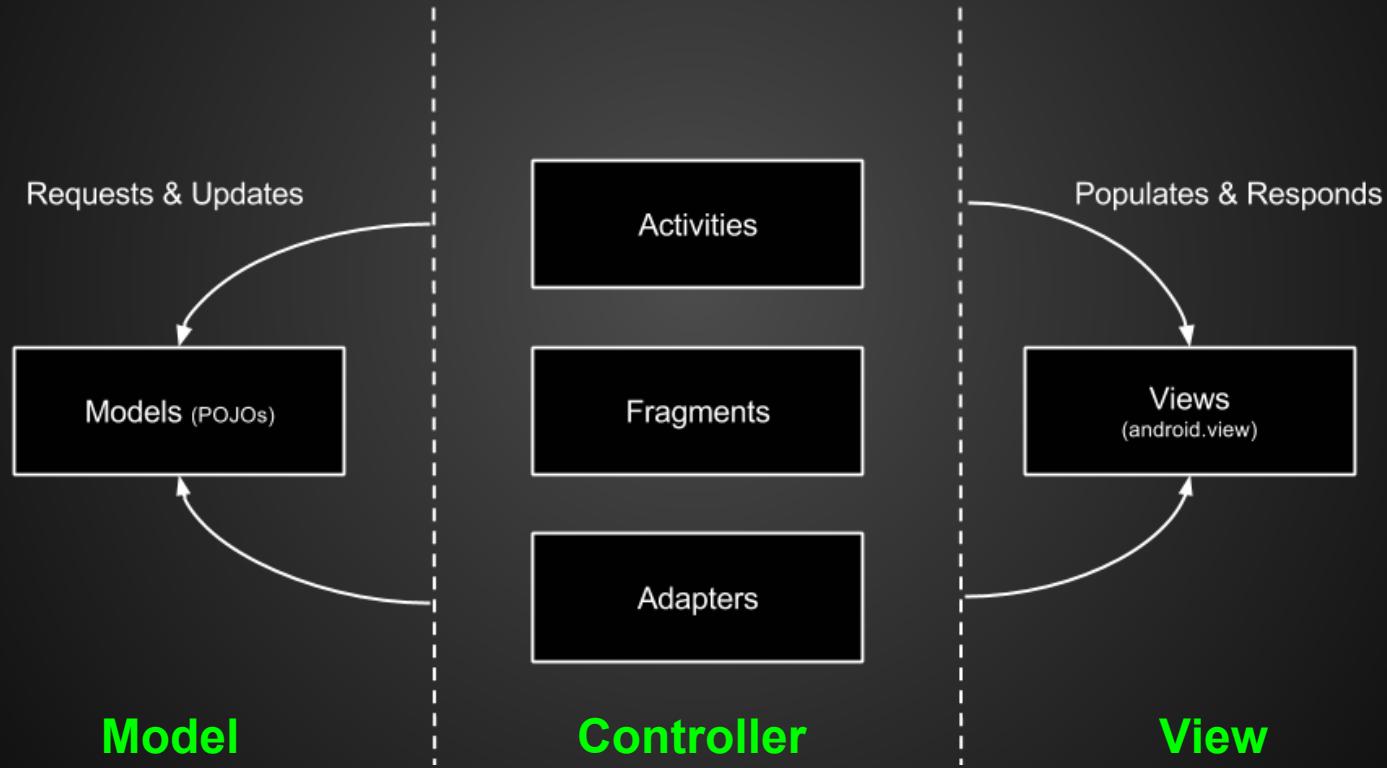
Enough of the
Yaks
On with the Code!
(AND SECTION 2)

Section 2

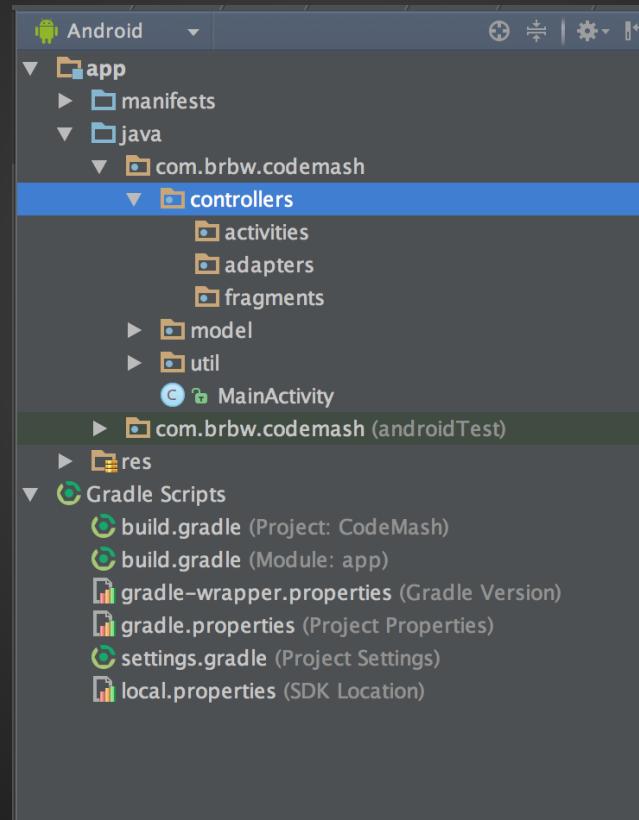
Android Sledge-O-Matic

Raise your hand if you “know MVC”

Android “MVC”

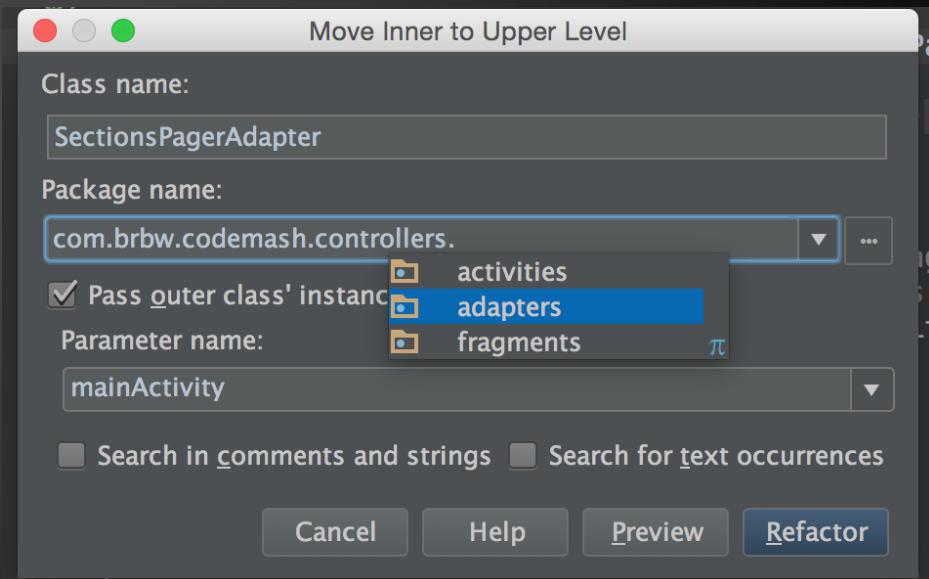


Let's start with a little refactor...



Move the adapter & fragment out

```
/**  
 * A {@link FragmentPagerAdapter} that returns a fragment corresponding to  
 * one of the sections/tabs/pages.  
 */  
public class SectionsPagerAdapter extends FragmentPagerAdapter {  
    public Section getItem(int position) { super(fm); }  
  
    @Override  
    public Fragment getItem(int position) {  
        // get the page title from the section metadata  
        Locale l = Locale.getDefault();  
        switch (position) {  
            case 0: return PlaceholderFragment.newInstance(l.getDisplayNames());  
            case 1: return PlaceholderFragment.newInstance(l.getDisplayNames());  
            case 2: return PlaceholderFragment.newInstance(l.getDisplayNames());  
        }  
    }  
  
    @Override  
    public CharSequence getPageTitle(int position) {  
        Locale l = Locale.getDefault();  
        switch (position) {  
            case 0: return "Section One";  
            case 1: return "Section Two";  
            case 2: return "Section Three";  
        }  
    }  
}
```



Now let's take a few minutes to discuss each and their role here

- MainActivity
- SectionsPagerAdapter
- PlaceHolderFragment

ActionBar

CodeMash

⋮

Menu

SECTION 1

SECTION 2

SECTION 3

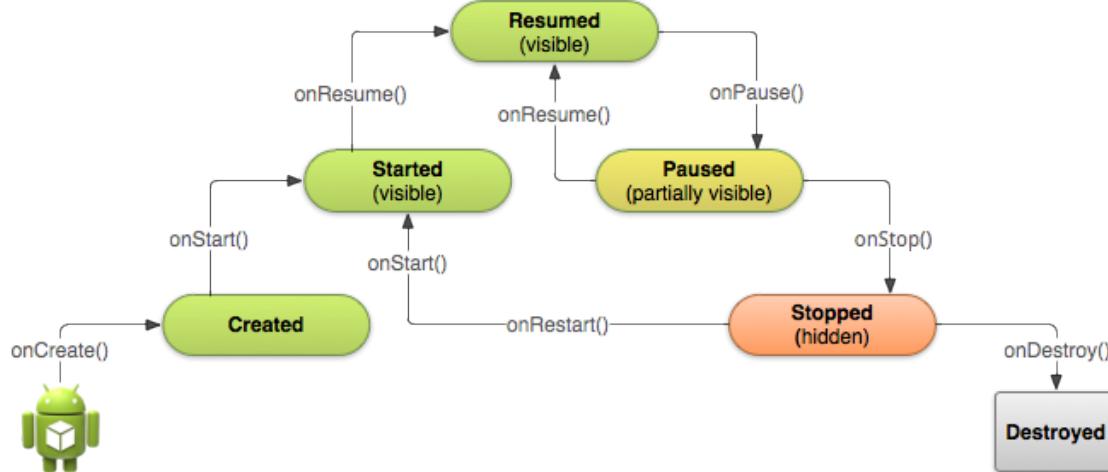
ViewPager

Fragment(s)

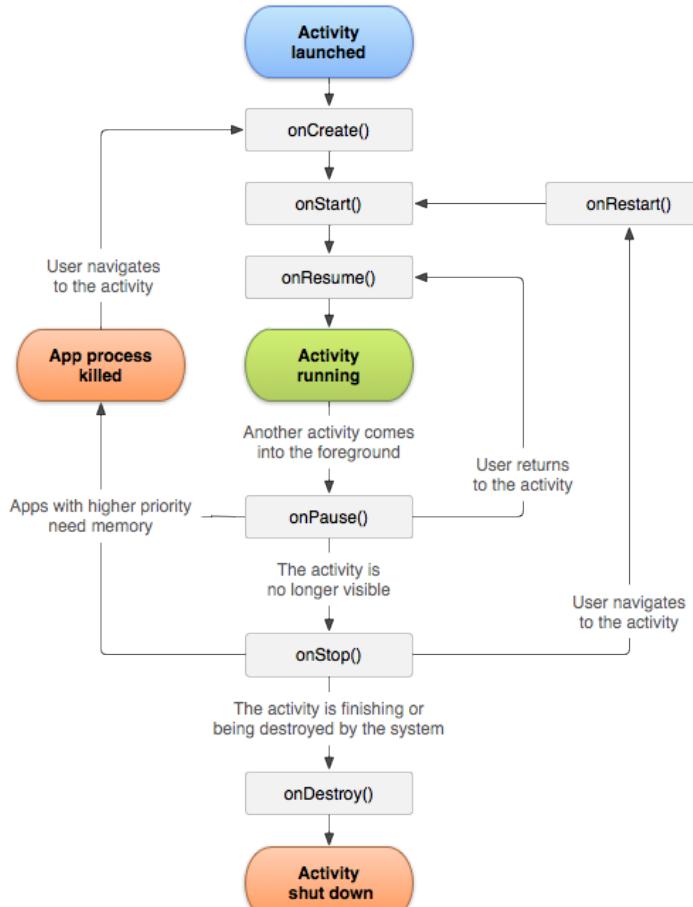
Activity

FragmentPagerAdapter

Why's everything in onCreate() ?

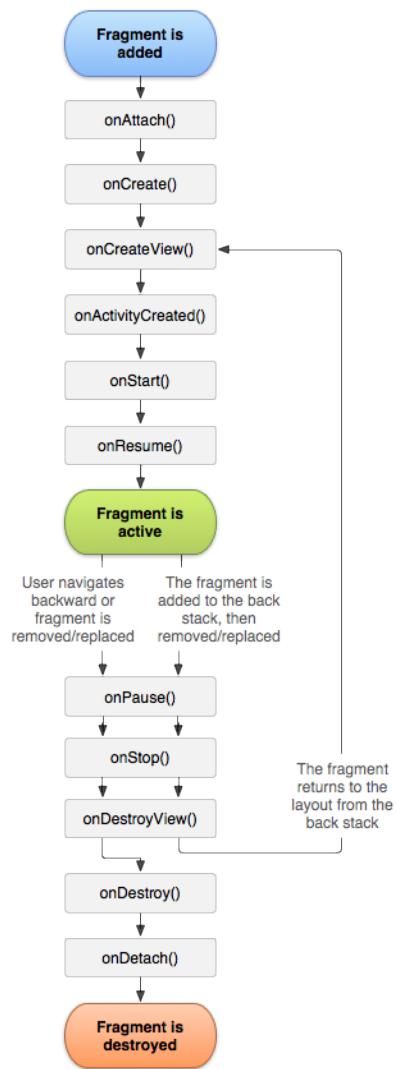


Source: <http://developer.android.com/training/basics/activity-lifecycle/startling.html>



Source: <http://developer.android.com/guide/components/activities.html>

Fragments Have something similar



Source: <http://developer.android.com/guide/components/fragments.html>

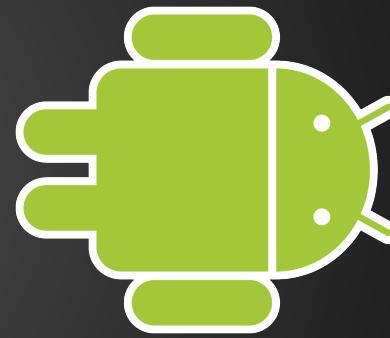
Just to see this in action...

```
public class MainActivity extends ActionBarActivity implements ActionBar.TabListener {  
  
    private static final String LOG_TAG = "MainActivity";  
  
    private SectionsPagerAdapter mSectionsPagerAdapter;  
    private ViewPager mViewPager;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Log.d(LOG_TAG, "onCreate was called");
```

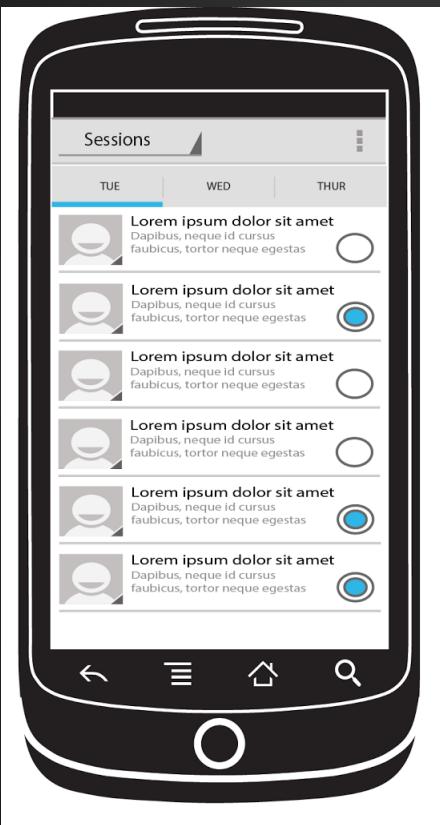
Just to see this in action...

```
PlaceholderFragment.java x
20         Bundle args = new Bundle();
21         args.putInt(ARG_SECTION_NUMBER, sectionNumber);
22         fragment.setArguments(args);
23         return fragment;
24     }
25
26     @Override
27     public void onCreate(Bundle savedInstanceState) {
28         super.onCreate(savedInstanceState);
29         Log.d(LOG_TAG, "onCreate was called");
30     }
31
32     @Override
33     public View onCreateView(LayoutInflater inflater, ViewGroup container,
34                             Bundle savedInstanceState) {
35         View rootView = inflater.inflate(R.layout.fragment_main, container, false);
36         Log.d(LOG_TAG, "onCreateView was called");
37         return rootView;
38     }
39
40     @Override
41     public void onActivityCreated(@Nullable Bundle savedInstanceState) {
42         super.onActivityCreated(savedInstanceState);
43         Log.d(LOG_TAG, "onActivityCreated was called");
44     }
45
46     @Override
47     public void onStart() {
48         super.onStart();
49         Log.d(LOG_TAG, "onStart was called");
50     }
51
52     @Override
53     public void onResume() {
54         super.onResume();
55         Log.d(LOG_TAG, "onResume was called");
56     }
57 }
```

Now Rotate!



On to the days!



USB Twitter GitHub Email C. 4G 82% 1:21 PM

CodeMash

⋮

SECTION 1

SECTION 2

SECTION 3

Modifying the SectionsPagerAdapter

```
public class SectionsPagerAdapter extends FragmentPagerAdapter {  
  
    private MainActivity mainActivity;  
  
    public SectionsPagerAdapter(MainActivity mainActivity, FragmentManager fm) {  
        super(fm);  
        this.mainActivity = mainActivity;  
    }  
  
    public class SectionsPagerAdapter extends FragmentPagerAdapter {  
  
        private Context context;  
  
        public SectionsPagerAdapter(Context context, FragmentManager fm) {  
            super(fm);  
            this.context = context;  
        }  
  
        @Override  
        public Fragment getItem(int position) { return PlaceholderFragment.newInstance(position + 1); }  
  
        @Override  
        public int getCount() { return 3; }  
  
        @Override  
        public CharSequence getPageTitle(int position) {  
            Locale l = Locale.getDefault();  
            switch (position) {  
                case 0:  
                    return context.getString(R.string.title_section1).toUpperCase(l);  
                case 1:  
                    return context.getString(R.string.title_section2).toUpperCase(l);  
                case 2:  
                    return context.getString(R.string.title_section3).toUpperCase(l);  
            }  
            return null;  
        }  
    }  
}
```

What's a Context?

public abstract class

Context

extends [Object](#)

Summary: [Constants](#) | [Ctors](#) | [Methods](#) | [Inherited Methods](#) | [Expand All]

Added in API level 1

[java.lang.Object](#)

↳ android.content.Context

► Known Direct Subclasses

[ContextWrapper](#), [MockContext](#)

► Known Indirect Subclasses

[AbstractInputMethodService](#), [AccessibilityService](#), [AccountAuthenticatorActivity](#), [ActionBarActivity](#), [Activity](#), [ActivityGroup](#), [AliasActivity](#), [Application](#), [BackupAgent](#),
[BackupAgentHelper](#), and 31 others.



That's a lot of context!

Class Overview

Interface to global information about an application environment. This is an abstract class whose implementation is provided by the Android system. It allows access to application-specific resources and classes, as well as up-calls for application-level operations such as launching activities, broadcasting and receiving intents, etc.

source: <http://developer.android.com/reference/android/content/Context.html>

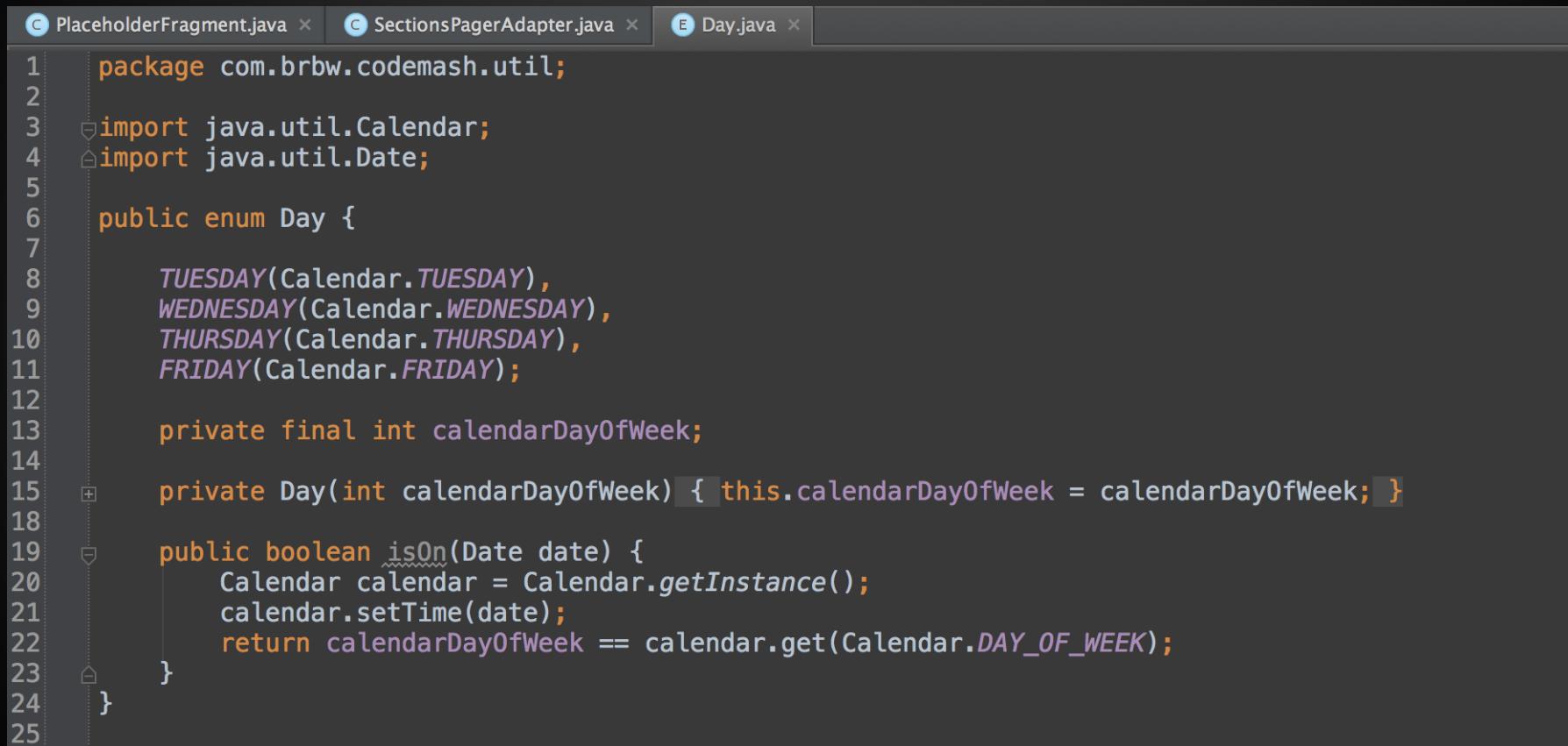
Back to Modifying the SectionsPagerAdapter...

```
public class SectionsPagerAdapter extends FragmentPagerAdapter {  
  
    private MainActivity mainActivity;  
  
    public SectionsPagerAdapter(MainActivity mainActivity, FragmentManager fm) {  
        super(fm);  
        this.mainActivity = mainActivity;  
    }  
  
    public class SectionsPagerAdapter extends FragmentPagerAdapter {  
  
        private Context context;  
  
        public SectionsPagerAdapter(Context context, FragmentManager fm) {  
            super(fm);  
            this.context = context;  
        }  
  
        @Override  
        public Fragment getItem(int position) { return PlaceholderFragment.newInstance(position + 1); }  
  
        @Override  
        public int getCount() { return 3; }  
  
        @Override  
        public CharSequence getPageTitle(int position) {  
            Locale l = Locale.getDefault();  
            switch (position) {  
                case 0:  
                    return context.getString(R.string.title_section1).toUpperCase(l);  
                case 1:  
                    return context.getString(R.string.title_section2).toUpperCase(l);  
                case 2:  
                    return context.getString(R.string.title_section3).toUpperCase(l);  
            }  
            return null;  
        }  
    }  
}
```

Modifying the SectionsPagerAdapter

```
public class SectionsPagerAdapter extends FragmentPagerAdapter {  
  
    private static final int TOTAL_NUMBER_OF_CONFERENCE_DAYS = 4;  
    private Context context;  
  
    public SectionsPagerAdapter(Context context, FragmentManager fm) {  
        super(fm);  
        this.context = context;  
    }  
  
    @Override  
    public Fragment getItem(int position) { return PlaceholderFragment.newInstance(position + 1); }  
  
    @Override  
    public int getCount() {  
        return TOTAL_NUMBER_OF_CONFERENCE_DAYS;  
    }  
  
    @Override  
    public CharSequence getPageTitle(int position) {  
        Locale l = Locale.getDefault();  
        switch (position) {  
            case 0:  
                return context.getString(R.string.tuesday).toUpperCase(l);  
            case 1:  
                return context.getString(R.string.wednesday).toUpperCase(l);  
            case 2:  
                return context.getString(R.string.thursday).toUpperCase(l);  
            case 3:  
                return context.getString(R.string.friday).toUpperCase(l);  
        }  
        return "";  
    }  
}
```

Now let's have our fragments tell us what day they're on...



The screenshot shows a code editor with three tabs at the top: PlaceholderFragment.java, SectionsPagerAdapter.java, and Day.java. The Day.java tab is active, displaying the following Java code:

```
1 package com.brbw.codemash.util;
2
3 import java.util.Calendar;
4 import java.util.Date;
5
6 public enum Day {
7
8     TUESDAY(Calendar.TUESDAY),
9     WEDNESDAY(Calendar.WEDNESDAY),
10    THURSDAY(Calendar.THURSDAY),
11    FRIDAY(Calendar.FRIDAY);
12
13    private final int calendarDayOfWeek;
14
15    private Day(int calendarDayOfWeek) { this.calendarDayOfWeek = calendarDayOfWeek; }
16
17    public boolean isOn(Date date) {
18        Calendar calendar = Calendar.getInstance();
19        calendar.setTime(date);
20        return calendarDayOfWeek == calendar.get(Calendar.DAY_OF_WEEK);
21    }
22}
23
24}
25}
```

The code defines an enum named Day with four values: TUESDAY, WEDNESDAY, THURSDAY, and FRIDAY, each corresponding to a specific day of the week from the Calendar class. It includes a private constructor that initializes a field calendarDayOfWeek with the value from the constructor of the Calendar class. The class also contains a public method isOn that takes a Date object and returns true if the day of the week matches the enum value. The code editor shows syntax highlighting and some code completion or inspection annotations.

And let's have our fragments tell us what day they're on...

```
public class PlaceholderFragment extends Fragment {

    private static final String LOG_TAG = "PlaceholderFragment";
    private static final String ARG_DAY = "day";

    private Day day;

    public static Fragment newInstance(Day day) {
        PlaceholderFragment fragment = new PlaceholderFragment();
        Bundle args = new Bundle();
        args.putSerializable(ARG_DAY, day);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (this.getArguments() != null) {
            day = (Day) getArguments().getSerializable(ARG_DAY);
        }
    }

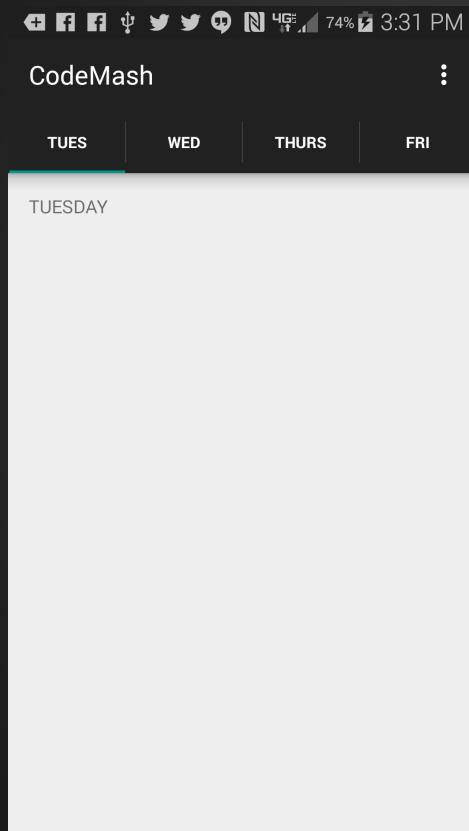
    private boolean hasArguments() {
        return getArguments() != null && getArguments().containsKey(ARG_DAY);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_main, container, false);
        if (day != null) {
            TextView textView = (TextView) rootView.findViewById(R.id.section_label);
            textView.setText(day.name());
        }
        return rootView;
    }
}
```

And let's have our fragments tell us what day they're on...

```
public class SectionsPagerAdapter extends FragmentPagerAdapter {  
  
    private static final int TOTAL_NUMBER_OF_CONFERENCE_DAYS = 4;  
    private Context context;  
  
    public SectionsPagerAdapter(Context context, FragmentManager fm) {  
        super(fm);  
        this.context = context;  
    }  
  
    @Override  
    public Fragment getItem(int position) {  
  
        switch (position) {  
            case 0:  
                return PlaceholderFragment.newInstance(Day.TUESDAY);  
            case 1:  
                return PlaceholderFragment.newInstance(Day.WEDNESDAY);  
            case 2:  
                return PlaceholderFragment.newInstance(Day.THURSDAY);  
            case 3:  
                return PlaceholderFragment.newInstance(Day.FRIDAY);  
        }  
        return null;  
    }  
}
```

And if all went well you should see...

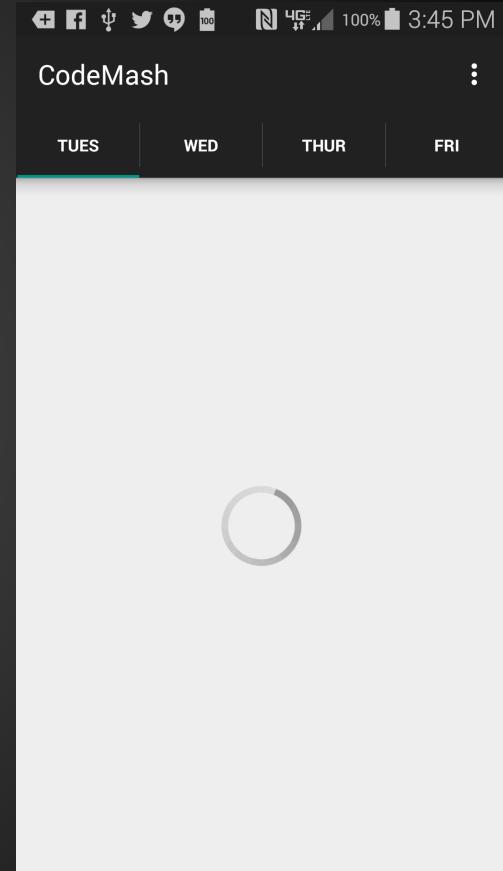


Now let's get a list displayed on these fragments...

Introducing ListFragment!

Now let's get a list displayed on these fragments...

```
SessionListFragment.java x
1 package com.brbw.codemash.controllers.fragments;
2
3 import android.os.Bundle;
4 import android.support.v4.app.Fragment;
5 import android.support.v4.app.ListFragment;
6
7 import com.brbw.codemash.models.Day;
8
9 public class SessionListFragment extends ListFragment {
10
11     private static final String LOG_TAG = "PlaceHolderFragment";
12     private static final String ARG_DAY = "day";
13
14     private Day day;
15
16     public static Fragment newInstance(Day day) {
17         SessionListFragment fragment = new SessionListFragment();
18         Bundle args = new Bundle();
19         args.putSerializable(ARG_DAY, day);
20         fragment.setArguments(args);
21         return fragment;
22     }
23
24     @Override
25     public void onCreate(Bundle savedInstanceState) {
26         super.onCreate(savedInstanceState);
27         if (this.getArguments() != null) {
28             day = (Day) getArguments().getSerializable(ARG_DAY);
29         }
30     }
31
32     private boolean hasArguments() {
33         return getArguments() != null && getArguments().containsKey(ARG_DAY);
34     }
35
36 }
37
```



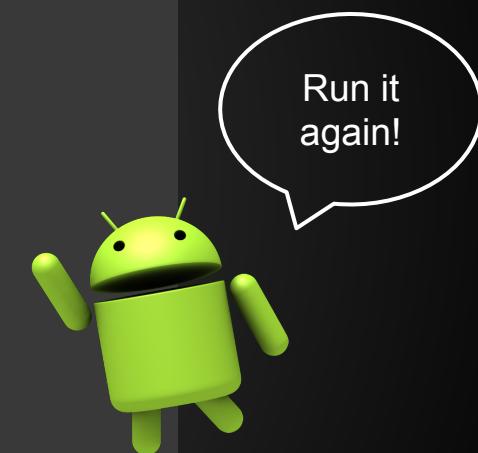
Now let's get a list displayed on these fragments...

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    if (this.getArguments() != null)  
        day = (Day) getArguments().getSerializable(ARG_DAY);  
  
    List<String> emptyList = new ArrayList<>();  
  
    ArrayAdapter<String> arrayAdapter = new ArrayAdapter<>(getActivity(),  
        android.R.layout.simple_list_item_1, emptyList);  
  
    setListAdapter(arrayAdapter);  
}
```

```
@Override  
public void onViewCreated(View view, Bundle savedInstanceState) {  
    super.onViewCreated(view, savedInstanceState);  
    CharSequence message = getText(R.string.empty_session_list_message);  
    setEmptyText(message);  
}
```

Now let's get a list displayed on these fragments...

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    ArrayAdapter<String> arrayAdapter = new ArrayAdapter<>(getActivity()  
        , android.R.layout.simple_list_item_1);  
  
    if (this.getArguments().containsKey(ARG_DAY)) {  
        Day day = (Day) getArguments().getSerializable(ARG_DAY);  
  
        List<String> singleItemList = Arrays.asList(day.name());  
        arrayAdapter.addAll(singleItemList);  
    }  
  
    setListAdapter(arrayAdapter);  
}
```



Let's take a look @ this ArrayAdapter thing : [http://developer.android.](http://developer.android.com/reference/android/widget/ArrayAdapter.html)

[com/reference/android/widget/ArrayAdapter.html](http://developer.android.com/reference/android/widget/ArrayAdapter.html)

Let's recap before moving forward

- Activities
- Fragments
- Adapters
- Contexts
- Resources



Section 3

Lies, Damn Lies, and Data

Your first Android Tests

- Making a simple HTTP request with SimpleJsonRequestor
 - HttpURLConnection & URL from java.net
 - Strings from the InputStream
- Models & JSON.org



checkout: <http://developer.android.com/reference/android/test/package-summary.html>

and:
<https://developer.android.com/tools/testing/index.html>

Your first Android Tests



There's going to be a lot of live-coding
here.
~fingers crossed~

But Android's
don't have
fingers

Your first Android Tests

```
public class CodeMashServiceIntegrationTest extends AndroidTestCase {  
  
    public void testThatWeCanConnectToTheService() throws Exception {  
  
        URL url = new URL("https://cmprod-speakers.azurewebsites.net/api/sessionsdata");  
        URLConnection urlConnection = url.openConnection();  
        InputStream inputStream = urlConnection.getInputStream();  
  
        assertNotNull(inputStream);  
    }  
}
```

Your first Android Tests

```
Running tests
Test running started
java.lang.SecurityException: Permission denied (missing INTERNET permission?)
at java.net.InetAddress.lookupHostName(InetAddress.java:418)
at java.net.InetAddress.getAllByNameImpl(InetAddress.java:236)
at java.net.InetAddress.getAllByName(InetAddress.java:214)
at com.android.okhttp.internal.Dns$1.getAllByName(Dns.java:28)
at com.android.okhttp.internal.http.RouteSelector.resetNextInetSocketAddress(RouteSelector.java:216)
at com.android.okhttp.internal.http.RouteSelector.next(RouteSelector.java:122)
```



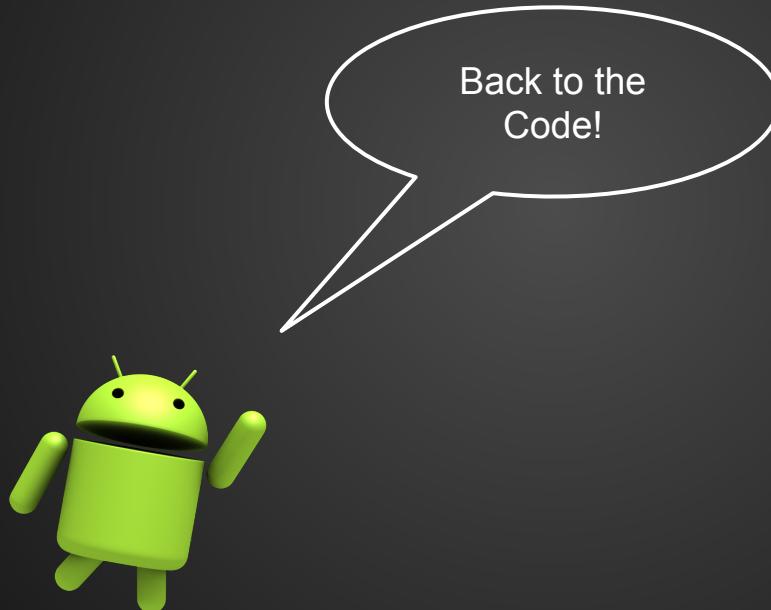
Time to modify our Manifest!

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.brbw.codemash" >

    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
```

Your first Android Tests



Back to the
Code!

**If you're a bit behind, feel free to
add what we've written. It'll save
some time.**

Modify it to your hearts content!

That was a lot of boilerplate! Are there any alternatives?

YES!

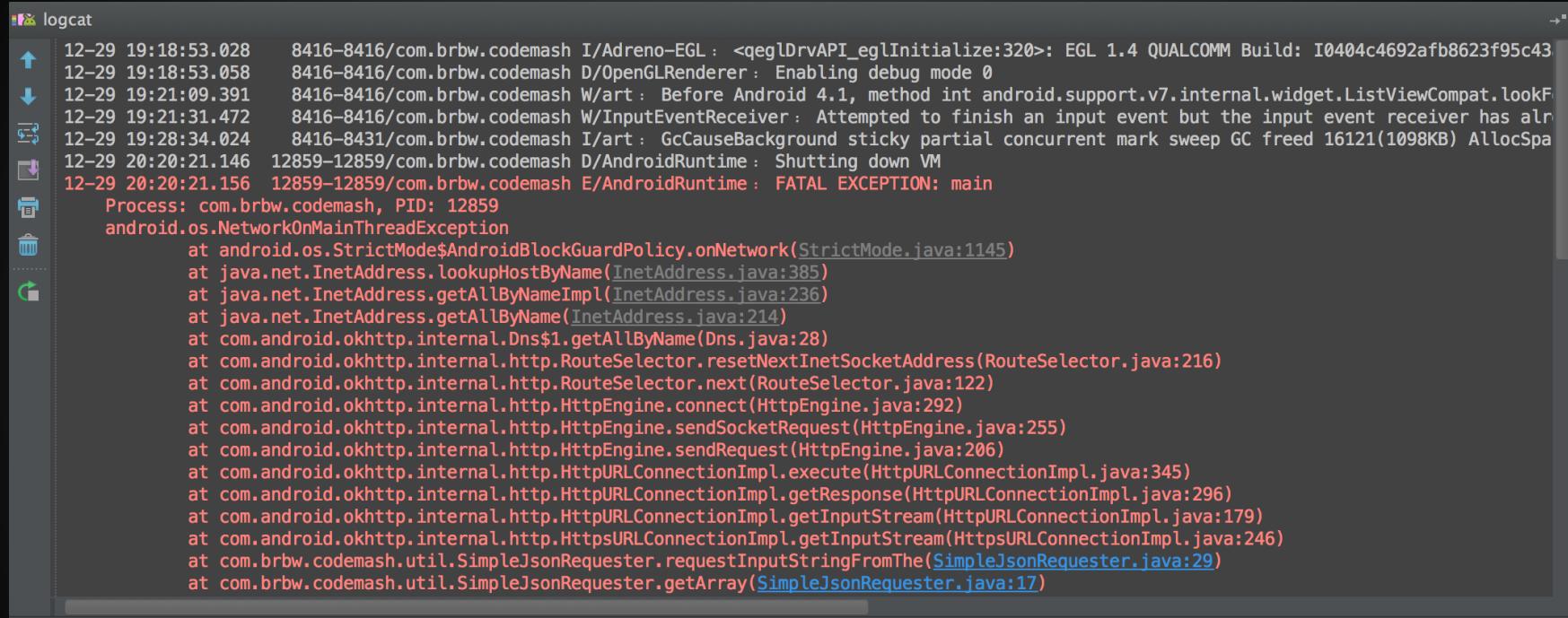
- Retrofit (By Square) <http://square.github.io/retrofit/>
- GSON <https://code.google.com/p/google-gson/>
- Guava <https://code.google.com/p/guava-libraries/>
- In some cases, volley: <https://developer.android.com/training/volley/simple.html>
- And then some...

For now, we're going to learn Android the hard way ;)

So... can we use that service we wrote?

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    ArrayAdapter<Session> arrayAdapter = new ArrayAdapter<>(getActivity()  
        , android.R.layout.simple_list_item_1);  
  
    if (this.hasArgumentsFor(ARG_DAY)) {  
        day = (Day) getArguments().getSerializable(ARG_DAY);  
  
        CodeMashService service = new CodeMashService(new SimpleJsonRequester());  
        List<Session> sessions = service.getSessions();  
  
        arrayAdapter.addAll(sessions);  
    }  
  
    setListAdapter(arrayAdapter);  
}
```

So... can we use that service we wrote?



The screenshot shows the Android Logcat window with the following log entries:

```
12-29 19:18:53.028 8416-8416/com.brbw.codemash I/Adreno-EGL : <qeglDrvAPI_eglInitialize:320>: EGL 1.4 QUALCOMM Build: I0404c4692afb8623f95c43
12-29 19:18:53.058 8416-8416/com.brbw.codemash D/OpenGLESRenderer: Enabling debug mode 0
12-29 19:21:09.391 8416-8416/com.brbw.codemash W/art: Before Android 4.1, method int android.support.v7.internal.widget.ListViewCompat.lookForSelectablePosition() would have leaked java.util.List local reference
12-29 19:21:31.472 8416-8416/com.brbw.codemash W/InputEventReceiver: Attempted to finish an input event but the input event receiver has already been disposed
12-29 19:28:34.024 8416-8431/com.brbw.codemash I/art: GcCauseBackground sticky partial concurrent mark sweep GC freed 16121(1098KB) AllocSpace
12-29 20:20:21.146 12859-12859/com.brbw.codemash D/AndroidRuntime: Shutting down VM
12-29 20:20:21.156 12859-12859/com.brbw.codemash E/AndroidRuntime: FATAL EXCEPTION: main
    Process: com.brbw.codemash, PID: 12859
    android.os.NetworkOnMainThreadException
        at android.os.StrictMode$AndroidBlockGuardPolicy.onNetwork(StrictMode.java:1145)
        at java.net.InetAddress.lookupHostName(InetAddress.java:385)
        at java.net.InetAddress.getAllByNameImpl(InetAddress.java:236)
        at java.net.InetAddress.getAllByName(InetAddress.java:214)
        at com.android.okhttp.internal.Dns$1.getAllByName(Dns.java:28)
        at com.android.okhttp.internal.http.RouteSelector.resetNextInetSocketAddress(RouteSelector.java:216)
        at com.android.okhttp.internal.http.RouteSelector.next(RouteSelector.java:122)
        at com.android.okhttp.internal.http.HttpEngine.connect(HttpEngine.java:292)
        at com.android.okhttp.internal.http.HttpEngine.sendSocketRequest(HttpEngine.java:255)
        at com.android.okhttp.internal.http.HttpEngine.sendRequest(HttpEngine.java:206)
        at com.android.okhttp.internal.http.HttpURLConnectionImpl.execute(HttpURLConnectionImpl.java:345)
        at com.android.okhttp.internal.http.HttpURLConnectionImpl.getResponse(HttpURLConnectionImpl.java:296)
        at com.android.okhttp.internal.http.HttpURLConnectionImpl.getInputStream(HttpURLConnectionImpl.java:179)
        at com.android.okhttp.internal.http.HttpsURLConnectionImpl.getInputStream(HttpsURLConnectionImpl.java:246)
        at com.brbw.codemash.util.SimpleJsonRequester.requestInputStringFromThe(SimpleJsonRequester.java:29)
        at com.brbw.codemash.util.SimpleJsonRequester.getArray(SimpleJsonRequester.java:17)
```

(NO)

So let's talk about AsyncTasks

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (this.hasArgumentsFor(ARG_DAY)) {
            day = (Day) getArguments().getSerializable(ARG_DAY);

            AsyncTask<Void,Void,List<Session>> task = new AsyncTask<Void, Void, List<Session>>() {
                @Override
                protected List<Session> doInBackground(Void... params) {
                    CodeMashService service = new CodeMashService(new SimpleJsonRequester());
                    return service.getSessions();
                }

                @Override
                protected void onPostExecute(List<Session> sessions) {
                    super.onPostExecute(sessions);
                    ArrayAdapter<Session> arrayAdapter = new ArrayAdapter<>(getActivity()
                        , android.R.layout.simple_list_item_1,
                        sessions);

                    setListAdapter(arrayAdapter);
                }
            };
            task.execute();
        }
    }
```

The 4 steps

When an asynchronous task is executed, the task goes through 4 steps:

1. `onPreExecute()`, invoked on the UI thread before the task is executed. This step is normally used to setup the task, for instance by showing a progress bar in the user interface.
2. `doInBackground(Params...)`, invoked on the background thread immediately after `onPreExecute()` finishes executing. This step is used to perform background computation that can take a long time. The parameters of the asynchronous task are passed to this step. The result of the computation must be returned by this step and will be passed back to the last step. This step can also use `publishProgress(Progress...)` to publish one or more units of progress. These values are published on the UI thread, in the `onProgressUpdate(Progress...)` step.
3. `onProgressUpdate(Progress...)`, invoked on the UI thread after a call to `publishProgress(Progress...)`. The timing of the execution is undefined. This method is used to display any form of progress in the user interface while the background computation is still executing. For instance, it can be used to animate a progress bar or show logs in a text field.
4. `onPostExecute(Result)`, invoked on the UI thread after the background computation finishes. The result of the background computation is passed to this step as a parameter.

Cancelling a task

A task can be cancelled at any time by invoking `cancel(boolean)`. Invoking this method will cause subsequent calls to `isCancelled()` to return true. After invoking this method, `onCancelled(Object)`, instead of `onPostExecute(Object)` will be invoked after `doInBackground(Object[])` returns. To ensure that a task is cancelled as quickly as possible, you should always check the return value of `isCancelled()` periodically from `doInBackground(Object[])`, if possible (inside a loop for instance.)

Threading rules

There are a few threading rules that must be followed for this class to work properly:

- The `AsyncTask` class must be loaded on the UI thread. This is done automatically as of `JELLY_BEAN`.
- The task instance must be created on the UI thread.
- `execute(Params...)` must be invoked on the UI thread.
- Do not call `onPreExecute()`, `onPostExecute(Result)`, `doInBackground(Params...)`, `onProgressUpdate(Progress...)` manually.
- The task can be executed only once (an exception will be thrown if a second execution is attempted.)

```
1 package com.brbw.codemash.tasks;
2
3 import android.os.AsyncTask;
4
5 import com.brbw.codemash.model.CodeMashService;
6 import com.brbw.codemash.model.Session;
7 import com.brbw.codemash.util.SimpleJsonRequester;
8
9 import java.util.List;
10
11 public class SessionRetrievalTask extends AsyncTask<Void,Void,List<Session>>{
12
13     @Override
14     protected List<Session> doInBackground(Void... params) {
15         CodeMashService service = new CodeMashService(new SimpleJsonRequester());
16         return service.getSessions();
17     }
18 }
19
```

And just a quick
refactor...



```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    if (this.getArguments().containsKey(ARG_DAY)) {
        day = (Day) getArguments().getSerializable(ARG_DAY);

        SessionRetrievalTask task = new SessionRetrievalTask() {
            @Override
            protected void onPostExecute(List<Session> sessions) {
                super.onPostExecute(sessions);
                fillListWith(sessions);
            }
        };
        task.execute();
    }

    private void fillListWith(List<Session> sessions) {
        ArrayAdapter<Session> arrayAdapter = new ArrayAdapter<>(getActivity(),
            android.R.layout.simple_list_item_1,
            sessions);
        setListAdapter(arrayAdapter);
    }
}
```



CodeMash

TUES WED THURS FRI

- Registration Booth - Tuesday
- Artemis - Tuesday
- Game Room - Tuesday
- Get Shipping with Docker
- Get your game on with Unity and Oculus Rift! (Part 1)
- Holiday Light Fight DIY (Instance 1)
- Humanitarian Toolbox Codeathon (Part 1)
- Intro to Node.JS
- Java in the (Amazon) Cloud Workshop
- JavaScript Everywhere - Learn to Build a Cross-platform JavaScript Application that Works EVERYWHERE (Part 1)

Let's fix that...
We need a
smarter
Session List.
More Tests!



```
import static org.mockito.BDDMockito.given;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

public class SessionListTest extends CodeMashTestCase {

    private SessionList sessionList;
    private List<Session> stubSessions;

    @Override
    protected void setUp() throws Exception {
        super.setUp();
        stubSessions = getStubSessions();
        CodeMashService stubService = mock(CodeMashService.class);
        given(stubService.getSessions()).willReturn(stubSessions);
        sessionList = new SessionList(stubService);
    }

    public void testThatItGivesMeAllSessions() {
        List<Session> sessions = sessionList.getAllSessions();

        assertEquals(stubSessions.size(), sessions.size());
    }

    public void testThatItFiltersSessionsByDay() {
        List<Session> tuesdaysSessions = sessionList.getSessionsFor(Day.TUESDAY);
        List<Session> wednesdaySessions = sessionList.getSessionsFor(Day.WEDNESDAY);
        List<Session> thursdaySessions = sessionList.getSessionsFor(Day.THURSDAY);
        List<Session> fridaySessions = sessionList.getSessionsFor(Day.FRIDAY);

        assertEquals(2, tuesdaysSessions.size());
        assertEquals(1, wednesdaySessions.size());
        assertEquals(1, thursdaySessions.size());
        assertEquals(1, fridaySessions.size());
    }
}
```

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:21.0.3'
    androidTestCompile 'org.mockito:mockito-core:1.9.5'
    androidTestCompile "com.google.dexmaker:dexmaker:1.0"
    androidTestCompile "com.google.dexmaker:dexmaker-mockito:1.0"
}
```

```
public class SessionList {  
    private final Object lock = new Object();  
    private final CodeMashService service;  
    private List<Session> sessions;  
  
    public SessionList(CodeMashService service) {  
        this.service = service;  
    }  
  
    public List<Session> getAllSessions() {  
  
        if (sessions == null) {  
            synchronized (lock) {  
                sessions = service.getSessions();  
            }  
        }  
        return sessions;  
    }  
  
    public List<Session> getSessionsFor(Day dayOfTheWeek) {  
        List<Session> sessionsForDay = new ArrayList<>();  
        for (Session session : getAllSessions()) {  
            if (dayOfTheWeek.isOn(session.getSessionStartTime())) {  
                sessionsForDay.add(session);  
            }  
        }  
        return sessionsForDay;  
    }  
}
```

We really only
need 1 of these in
our app...



```
CodeMashApplication.java x
1 package com.brbw.codemash;
2
3 import android.app.Application;
4
5 import com.brbw.codemash.models.SessionList;
6 import com.brbw.codemash.util.network.SimpleJsonRequester;
7
8 public class CodeMashApplication extends Application {
9
10    private static SessionList sessionList;
11
12    public synchronized static SessionList sessionListInstance() {
13        if (sessionList == null) {
14            sessionList = new SessionList(new CodeMashService(new SimpleJsonRequester()));
15        }
16        return sessionList;
17    }
18}
19
```

We really only
need 1 of
these in our
app...



```
CodeMashApplication.java x AndroidManifest.xml x
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.brbw.codemash" >
4
5     <uses-permission android:name="android.permission.INTERNET"/>
6
7     <application
8         android:name=".CodeMashApplication"
9         android:allowBackup="true"
10        android:icon="@drawable/ic_launcher"
11        android:label="@string/app_name"
12        android:theme="@style/AppTheme" >
13            <activity
14                android:name=".controllers.activities.MainActivity"
15                android:label="CodeMash" >
```

Now let's
modify our
AsyncTask and
use it!



```
public class SessionRetrievalTask extends AsyncTask<Day, Void, List<Session>> {  
  
    @Override  
    protected List<Session> doInBackground(Day... day) {  
        if (this.has(day)) {  
            SessionList sessionList = CodeMashApplication.sessionListInstance();  
            return sessionList.getSessionsFor(day[0]);  
        } else {  
            return Arrays.asList();  
        }  
    }  
  
    private boolean has(Day[] day) {  
        return day != null && day.length == 1;  
    }  
}
```

```
SessionListFragment.java x SessionRetrievalTask.java x  
30  
31  
32 @Override  
33 public void onCreate(Bundle savedInstanceState) {  
34     super.onCreate(savedInstanceState);  
35  
36     if (this.hasArgumentsFor(ARG_DAY)) {  
37         day = (Day) getArguments().getSerializable(ARG_DAY);  
38     }  
39  
40     SessionRetrievalTask task = new SessionRetrievalTask() {  
41         @Override  
42         protected void onPostExecute(List<Session> sessions) {  
43             super.onPostExecute(sessions);  
44             fillListWith(sessions);  
45         }  
46     };  
47     task.execute(day);  
48 }  
49 }
```

Let's recap before moving forward

- android.test.*
- Android Permissions
- HTTP Requests (and a few good links)
- AsyncTasks
- Extending Application (and simple singletons)
- And then some...



Section 4

Adaptation

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:orientation="vertical">
6
7     <android.support.v7.widget.CardView
8         android:id="@+id/card_view"
9         android:layout_width="match_parent"
10        android:layout_height="match_parent">
11
12         </android.support.v7.widget.CardView>
13
14     </LinearLayout>
```

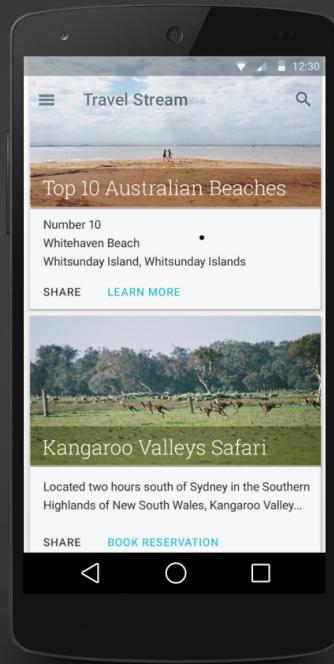
The following classes could not be found:

- android.support.
[\(Fix Build Path,](#)

💡 Tip: Try to [build](#) the project.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.v7.widget.CardView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:id="@+id/card_view"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8
9 </android.support.v7.widget.CardView>
10
11
```

What's a Cardview?



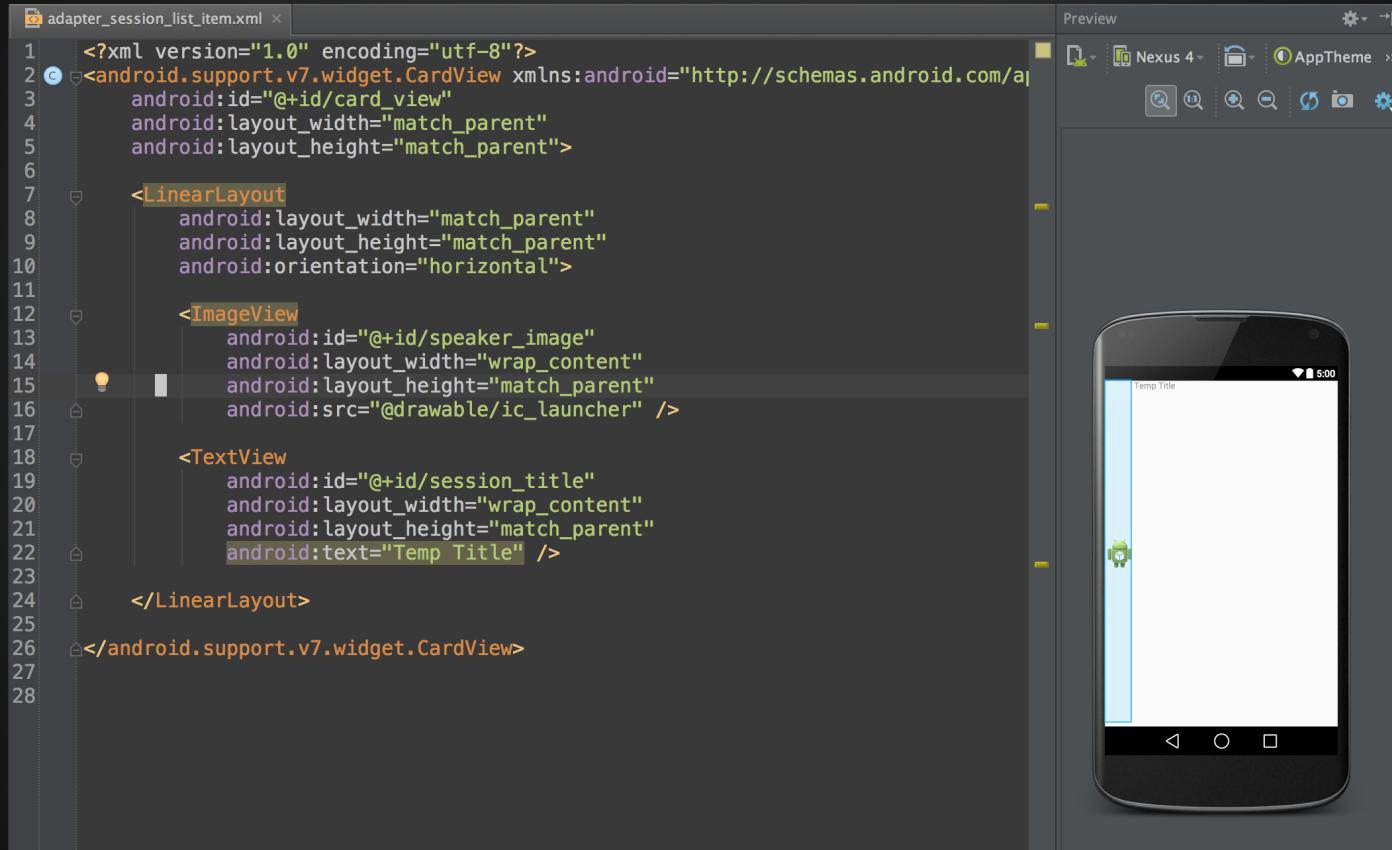
source: <https://developer.android.com/training/material/lists-cards.html>

adapter_session_list_item.xml x app x

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly. [Sync Now](#)

```
1 apply plugin: 'com.android.application'
2
3 android {
4     compileSdkVersion 21
5     buildToolsVersion "21.1.2"
6
7     defaultConfig {
8         applicationId "com.brbw.codemash"
9         minSdkVersion 15
10        targetSdkVersion 21
11        versionCode 1
12        versionName "1.0"
13    }
14    buildTypes {
15        release {
16            minifyEnabled false
17            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
18        }
19    }
20}
21
22 dependencies {
23     compile fileTree(dir: 'libs', include: ['*.jar'])
24     compile 'com.android.support:appcompat-v7:21.0.3'
25     compile 'com.android.support:cardview-v7:21.0.+'
26
27}
```

Back to our layout!



The screenshot shows the Android Studio interface with the XML layout editor on the left and the preview window on the right.

XML Layout Editor:

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

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">

        <ImageView
            android:id="@+id/speaker_image"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:src="@drawable/ic_launcher" />

        <TextView
            android:id="@+id/session_title"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:text="Temp Title" />
    </LinearLayout>
</android.support.v7.widget.CardView>
```

Preview Window:

The preview window shows a Nexus 4 device with a black background. A light blue vertical bar is visible on the left side. The main content area displays the text "Temp Title".

Now let's populate it!

```
adapter_session_list_item.xml SessionListAdapter.java
1 package com.brbw.codemash.controllers.adapters;
2
3 import ...
4
5 public class SessionListAdapter extends ArrayAdapter<Session> {
6
7     private static final int LAYOUT_ID = R.layout.adapter_session_list_item;
8
9     public SessionListAdapter(Context context, List<Session> sessions) {
10         super(context, LAYOUT_ID, sessions);
11     }
12
13     @Override
14     public View getView(int position, View convertView, ViewGroup parent) {
15
16         if(convertView == null) {
17             LayoutInflater inflater = (LayoutInflater) getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
18             convertView = inflater.inflate(LAYOUT_ID, parent, false);
19         }
20
21         Session session = getItem(position);
22
23         TextView title = (TextView) convertView.findViewById(R.id.session_title);
24         title.setText(session.getTitle());
25
26         return convertView;
27     }
28 }
```



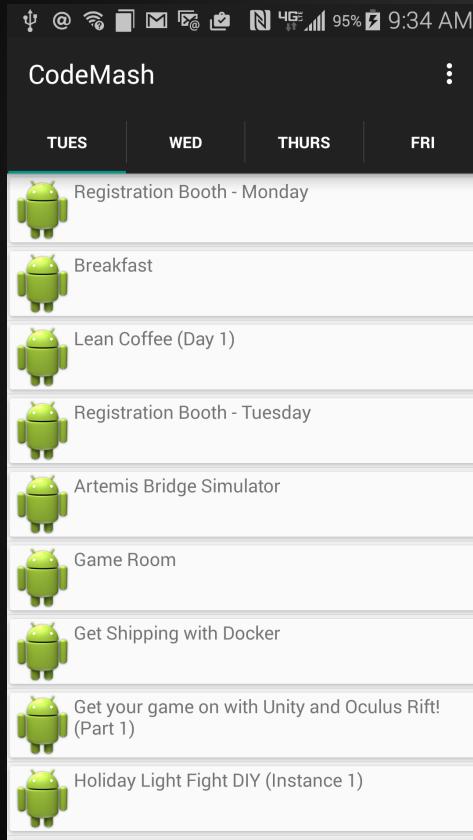
checkout: <http://developer.android.com/reference/android/widget/Adapter.html>

Now let's populate it!

```
adapter_session_list_item.xml SessionListAdapter.java
1 package com.brbw.codemash.controllers.adapters;
2
3 import ...
4
5 public class SessionListAdapter extends ArrayAdapter<Session> {
6
7     private static final int LAYOUT_ID = R.layout.adapter_session_list_item;
8
9     public SessionListAdapter(Context context, List<Session> sessions) {
10         super(context, LAYOUT_ID, sessions);
11     }
12
13     @Override
14     public View getView(int position, View convertView, ViewGroup parent) {
15
16         if(convertView == null) {
17             LayoutInflator inflater = (LayoutInflator) getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
18             convertView = inflater.inflate(LAYOUT_ID, parent, false);
19         }
20
21         Session session = getItem(position);
22
23         TextView title = (TextView) convertView.findViewById(R.id.session_title);
24         title.setText(session.getTitle());
25
26         return convertView;
27     }
28 }
```



But what about those images?...



Picasso

A powerful image downloading and caching library for Android

Introduction

Images add much-needed context and visual flair to Android applications. Picasso allows for hassle-free image loading in your application—often in one line of code!

```
Picasso.with(context).load("http://i.imgur.com/DvpvklR.png").into(imageView);
```

Many common pitfalls of image loading on Android are handled automatically by Picasso:

- Handling `ImageView` recycling and download cancelation in an adapter.
- Complex image transformations with minimal memory use.
- Automatic memory and disk caching.

The image shows a black smartphone with a white home screen. The screen displays a grid of nine Picasso sample images, which are abstract, colorful artworks. The phone's status bar at the top shows signal strength, battery level (95%), and the time (9:34 AM).

source: <http://square.github.io/picasso/>

But what about those images?...

Picasso

A powerful image downloading and caching library for Android

```
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
    compile 'com.android.support:appcompat-v7:21.0.3'  
    compile 'com.android.support:cardview-v7:21.0.3'  
    compile 'com.squareup.picasso:picasso:2.4.0'  
}
```

A black smartphone is shown from a slightly elevated angle, displaying a grid of nine abstract artworks in a Picasso style. The images are arranged in a 3x3 grid on the screen. The phone is positioned in front of a dark background that features a red header bar with the word "Picasso" and a white text area below it.

source: <http://square.github.io/picasso/>

But what about those images?...

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

    if (convertView == null) {
        LayoutInflator inflater = (LayoutInflator) getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        convertView = inflater.inflate(LAYOUT_ID, parent, false);
    }

    Session session = getItem(position);

    TextView title = (TextView) convertView.findViewById(R.id.session_title);
    title.setText(session.getTitle());

    ImageView imageView = (ImageView) convertView.findViewById(R.id.speaker_image);
    String firstSpeakerUrl = session.getSpeakers().get(0).getGravatarUrl();

    String imageUrl;
    if (firstSpeakerUrl != null && !firstSpeakerUrl.startsWith("http")) {
        imageUrl = String.format("http:%s?s=100", firstSpeakerUrl);
    } else {
        imageUrl = String.format("%s?s=100", firstSpeakerUrl);
    }

    Picasso.with(getContext())
        .load(imageUrl)
        .into(imageView);

    return convertView;
}
```

<ImageView

```
    android:id="@+id/speaker_image"
    android:layout_width="70dp"
    android:layout_height="70dp"
    android:src="@drawable/ic_launcher" />
```

Let's start with something ugly and clean it up a bit....

A tad bit better... for now :)

```
public class SessionListAdapter extends ArrayAdapter<Session> {

    private static final int LAYOUT_ID = R.layout.adapter_session_list_item;

    public SessionListAdapter(Context context, List<Session> sessions) {
        super(context, LAYOUT_ID, sessions);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        if (convertView == null) {
            convertView = ViewHelper.inflateForAdapter(getContext(), parent, LAYOUT_ID);
        }

        Session session = getItem(position);

        ViewHelper viewHelper = new ViewHelper(convertView);
        viewHelper.setText(R.id.session_title, session.getTitle());
        viewHelper.loadImageFromUrlIntoImageView(R.id.speaker_image,
            String.format("%s?s=100", session.getSessionImageUrl()));

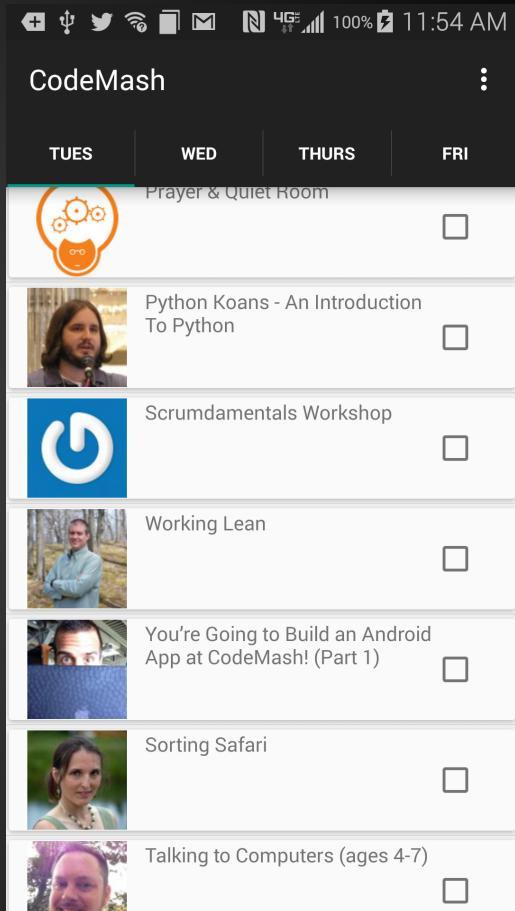
        return convertView;
    }
}
```

Section 5

You Are My Favorite!

Let's start by modifying the layout...

```
adapter_session_list_item.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.v7.widget.CardView
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:id="@+id/card_view"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:descendantFocusability="blocksDescendants">
8
9      <LinearLayout
10         android:layout_width="match_parent"
11         android:layout_height="match_parent"
12         android:orientation="horizontal"
13         android:weightSum="1">
14
15         <ImageView
16             android:id="@+id/speaker_image"
17             android:layout_width="70dp"
18             android:layout_height="70dp"
19             android:layout_weight=".10"
20             android:src="@drawable/ic_launcher" />
21
22         <TextView
23             android:id="@+id/session_title"
24             android:layout_width="0dp"
25             android:layout_height="match_parent"
26             android:layout_weight=".85"
27             android:text="Temp Title" />
28
29         <CheckBox
30             android:id="@+id/session_favorite"
31             android:layout_width="wrap_content"
32             android:layout_height="match_parent"
33             android:layout_weight=".05" />
34
35     </LinearLayout>
36
37 </android.support.v7.widget.CardView>
```



And let's add these lines to our adapter...

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView = ViewHelper.inflateForAdapter(getContext(), parent, LAYOUT_ID);
    }

    Session session = getItem(position);

    ViewHelper viewHelper = new ViewHelper(convertView);
    viewHelper.setText(R.id.session_title, session.getTitle());
    viewHelper.loadImageFromUrlIntoImageView(R.id.speaker_image,
        String.format("%s?s=100", session.getSessionImageUrl()));

    CheckBox favorites = ViewHelper.findView(convertView, R.id.session_favorite);
    favorites.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            Toast toast = Toast.makeText(getContext(), String.format("Favorited = %b", isChecked), Toast.LENGTH_LONG);
            toast.show();
        }
    });

    return convertView;
}
```



We should probably talk a bit about persistence!

Storage Options

Android provides several options for you to save persistent application data. The solution you choose depends on your specific needs, such as whether the data should be private to your application or accessible to other applications (and the user) and how much space your data requires.

Your data storage options are the following:

Shared Preferences

Store private primitive data in key-value pairs.

Internal Storage

Store private data on the device memory.

External Storage

Store public data on the shared external storage.

SQLite Databases

Store structured data in a private database.

Network Connection

Store data on the web with your own network server.

Android provides a way for you to expose even your private data to other applications – with a [content provider](#). A content provider is an optional component that exposes read/write access to your application data, subject to whatever restrictions you want to impose. For more information about using content providers, see the [Content Providers](#) documentation.

STORAGE QUICKVIEW

- Use Shared Preferences for primitive data
- Use internal device storage for private data
- Use external storage for large data sets that are not private
- Use SQLite databases for structured storage

IN THIS DOCUMENT

[Using Shared Preferences](#)

[Using the Internal Storage](#)

[Using the External Storage](#)

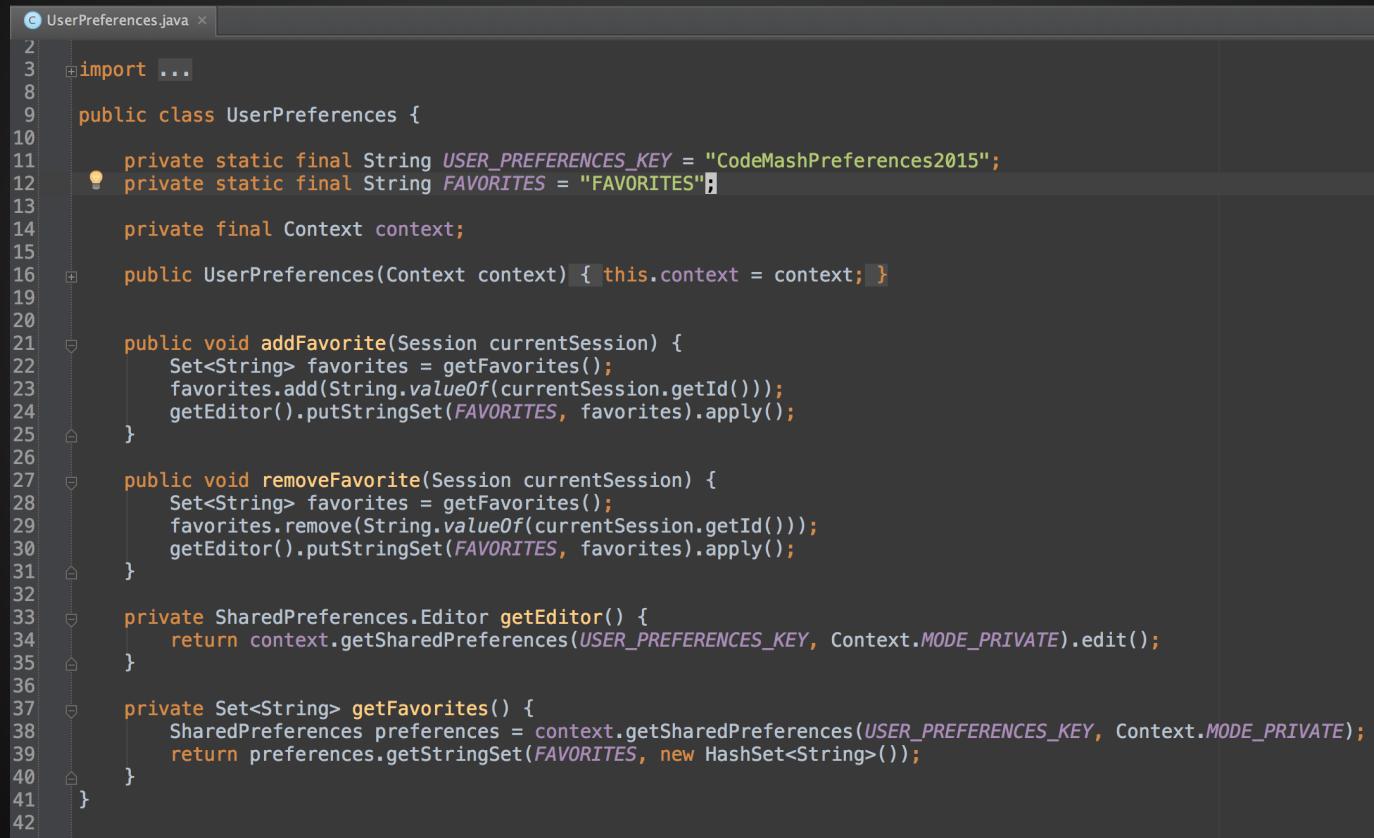
[Using Databases](#)

[Using a Network Connection](#)

SEE ALSO

[Content Providers and Content Resolvers](#)

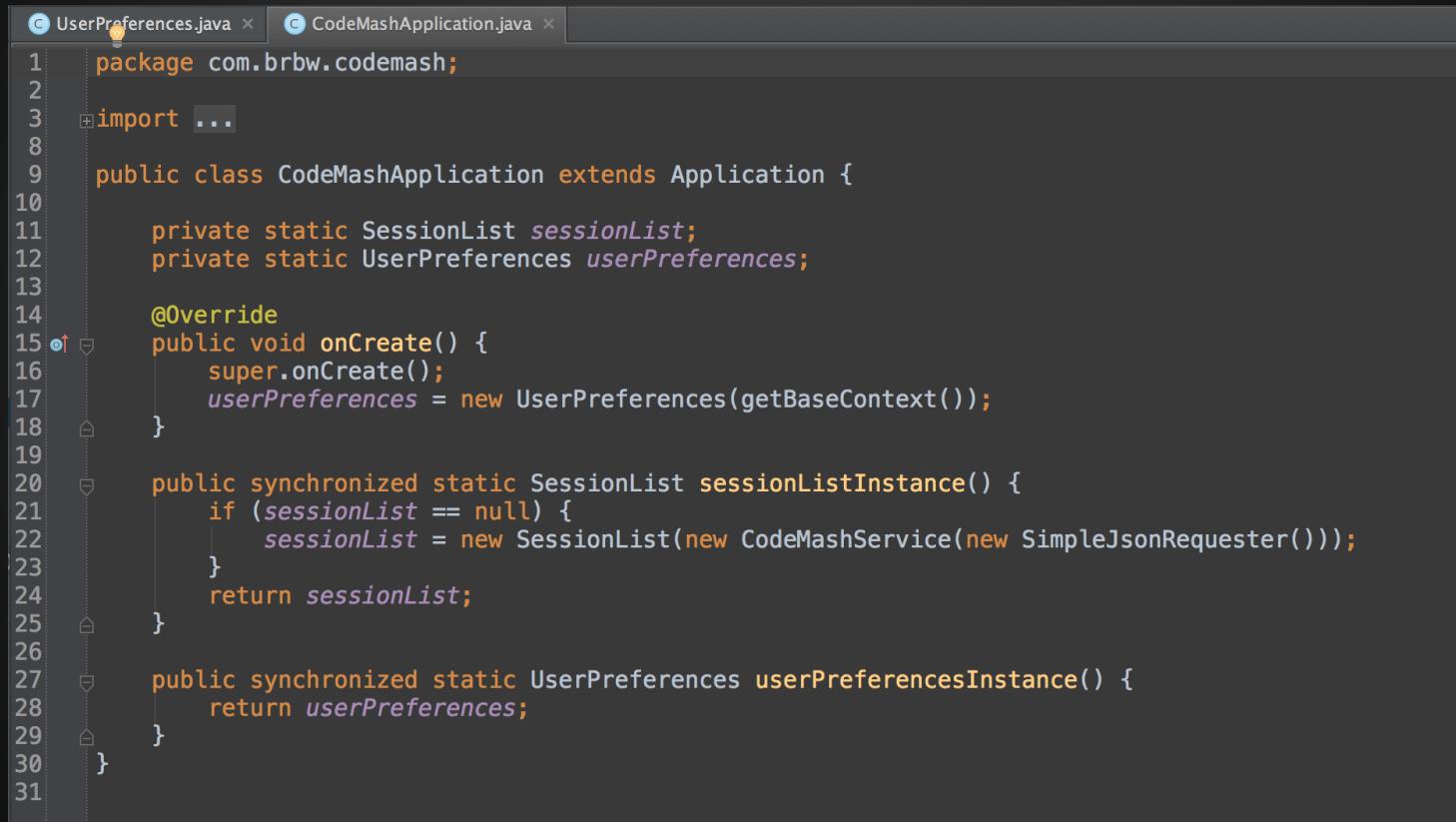
Introducing UserPreferences (feel free to name it something else. Names are hard!)



The screenshot shows a Java code editor window with the title "UserPreferences.java". The code implements a class named UserPreferences that manages user preferences, specifically favorite sessions. It uses a static final String for the preference key ("USER_PREFERENCES_KEY") and a private static final String for the favorites key ("FAVORITES"). The class has a private final Context variable and a constructor that initializes it. It contains two main methods: addFavorite and removeFavorite, which both update a Set<String> of favorites and apply the changes using a SharedPreferences.Editor object. The getFavorites method retrieves the set from SharedPreferences. The code is annotated with line numbers from 2 to 42.

```
2 import ...
3
4 public class UserPreferences {
5
6     private static final String USER_PREFERENCES_KEY = "CodeMashPreferences2015";
7     private static final String FAVORITES = "FAVORITES";
8
9     private final Context context;
10
11    public UserPreferences(Context context) { this.context = context; }
12
13
14    public void addFavorite(Session currentSession) {
15        Set<String> favorites = getFavorites();
16        favorites.add(String.valueOf(currentSession.getId()));
17        getEditor().putStringSet(FAVORITES, favorites).apply();
18    }
19
20    public void removeFavorite(Session currentSession) {
21        Set<String> favorites = getFavorites();
22        favorites.remove(String.valueOf(currentSession.getId()));
23        getEditor().putStringSet(FAVORITES, favorites).apply();
24    }
25
26    private SharedPreferences.Editor getEditor() {
27        return context.getSharedPreferences(USER_PREFERENCES_KEY, Context.MODE_PRIVATE).edit();
28    }
29
30    private Set<String> getFavorites() {
31        SharedPreferences preferences = context.getSharedPreferences(USER_PREFERENCES_KEY, Context.MODE_PRIVATE);
32        return preferences.getStringSet(FAVORITES, new HashSet<String>());
33    }
34
35    private void saveFavorites() {
36        Set<String> favorites = getFavorites();
37        getEditor().putStringSet(FAVORITES, favorites).apply();
38    }
39
40    private void loadFavorites() {
41        Set<String> favorites = getFavorites();
42    }
43}
```

Let's access this through the application as well



```
1 package com.brbw.codemash;
2
3 import ...
4
5 public class CodeMashApplication extends Application {
6
7     private static SessionList sessionList;
8     private static UserPreferences userPreferences;
9
10    @Override
11    public void onCreate() {
12        super.onCreate();
13        userPreferences = new UserPreferences(getApplicationContext());
14    }
15
16    public synchronized static SessionList sessionListInstance() {
17        if (sessionList == null) {
18            sessionList = new SessionList(new CodeMashService(new SimpleJsonRequester()));
19        }
20        return sessionList;
21    }
22
23    public synchronized static UserPreferences userPreferencesInstance() {
24        return userPreferences;
25    }
26
27}
```

Now lets use it in our adapter

```
@Override  
public View getView(int position, View convertView, ViewGroup parent) {  
  
    if (convertView == null) {  
        convertView = ViewHelper.inflateForAdapter(getContext(), parent, LAYOUT_ID);  
    }  
  
    final Session session = getItem(position);  
  
    ViewHelper viewHelper = new ViewHelper(convertView);  
    viewHelper.setText(R.id.session_title, session.getTitle());  
    viewHelper.loadImageFromUrlIntoImageView(R.id.speaker_image,  
        String.format("%s?size=100", session.getSessionImageUrl()));  
  
    CheckBox favorites = ViewHelper.findView(convertView, R.id.session_favorite);  
    favorites.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  
  
        @Override  
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {  
            UserPreferences userPreferences = CodeMashApplication.userPreferencesInstance();  
  
            if (isChecked != session.isFavorited()) {  
                if (isChecked) {  
                    userPreferences.addFavorite(session);  
                } else {  
                    userPreferences.removeFavorite(session);  
                }  
                session.isFavorited(isChecked);  
            }  
        }  
    });  
  
    favorites.setChecked(session.isFavorited());  
  
    return convertView;  
}
```



Now let's filter by favorites!

```
menu_main.xml:
```

```
1<menu xmlns:android="http://schemas.android.com/apk/res/android"
2    xmlns:app="http://schemas.android.com/apk/res-auto"
3    xmlns:tools="http://schemas.android.com/tools"
4    tools:context=".MainActivity">
5
6    <item
7        android:id="@+id/menu_filter"
8        android:icon="@android:drawable/ic_menu_sort_by_size"
9        android:title="@string/filter"
10       app:showAsAction="always" />
11
12</menu>
```

```
MainActivity.java:
```

```
62
63
64
65
66
67    @Override
68    public boolean onCreateOptionsMenu(Menu menu) {
69        getMenuInflater().inflate(R.menu.menu_main, menu);
70        return true;
71    }
72
73    @Override
74    public boolean onOptionsItemSelected(MenuItem item) {
75        int id = item.getItemId();
76        if (id == R.id.menu_filter) {
77            Toast.makeText(this,"That tickled",Toast.LENGTH_LONG).show();
78        }
79        return super.onOptionsItemSelected(item);
80
81    }
```

Let's start by modifying our menu

Let's bring up a dialog when the filter menu item's clicked

```
public class FilterDialogFragment extends DialogFragment {  
  
    public static final String TAG = "FILTER_DIALOG";  
  
    public static FilterDialogFragment newInstance() { return new FilterDialogFragment(); }  
  
    @NotNull  
    @Override  
    public Dialog onCreateDialog(Bundle savedInstanceState) {  
        final UserPreferences userPreferences = CodeMashApplication.userPreferencesInstance();  
        int favoritesOptionSelected = userPreferences.isFavoritesOnly() ? 1 : 0;  
  
        return new AlertDialog.Builder(getActivity())  
            .setTitle(R.string.filter_title)  
            .setSingleChoiceItems(R.array.favorites, favoritesOptionSelected, null)  
            .setPositiveButton(android.R.string.ok, null)  
            .create();  
    }  
}
```

Introducing DialogFragments!

Now lets modify our response in our MainActivity

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    int id = item.getItemId();  
    if (id == R.id.menu_filter) {  
        FilterDialogFragment filterDialogFragment = FilterDialogFragment.newInstance();  
        filterDialogFragment.show(getSupportFragmentManager(),FilterDialogFragment.TAG);  
        return true;  
    }  
    return super.onOptionsItemSelected(item);  
}
```

Now let's make this a bit sticky

```
public class FilterDialogFragment extends DialogFragment {

    public static final String TAG = "FILTER_DIALOG";

    public static FilterDialogFragment newInstance() { return new FilterDialogFragment(); }

    @NotNull
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        final UserPreferences userPreferences = CodeMashApplication.userPreferencesInstance();
        int favoritesOptionSelected = userPreferences.isFavoritesOnly() ? 1 : 0;

        return new AlertDialog.Builder(getActivity())
            .setTitle(R.string.filter_title)
            .setSingleChoiceItems(R.array.favorites, favoritesOptionSelected, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    if (which == 1) {
                        userPreferences.isFavoritesOnly(true);
                    } else {
                        userPreferences.isFavoritesOnly(false);
                    }
                }
            })
            .setPositiveButton(android.R.string.ok, null)
            .create();
    }
}
```

Now let's make this a bit sticky

```
public class UserPreferences {

    private static final String USER_PREFERENCES_KEY = "CodeMashPreferences2015";
    private static final String FAVORITES = "FAVORITES";

    private final Context context;
    private static boolean favoritesOnly;

    public UserPreferences(Context context) { this.context = context; }

    public void addFavorite(Session currentSession) {
        Set<String> favorites = getFavorites();
        favorites.add(String.valueOf(currentSession.getId()));
        getEditor().putStringSet(FAVORITES, favorites).apply();
    }

    public void removeFavorite(Session currentSession) {
        Set<String> favorites = getFavorites();
        favorites.remove(String.valueOf(currentSession.getId()));
        getEditor().putStringSet(FAVORITES, favorites).apply();
    }

    private SharedPreferences.Editor getEditor() {
        return context.getSharedPreferences(USER_PREFERENCES_KEY, Context.MODE_PRIVATE).edit();
    }

    private Set<String> getFavorites() {
        SharedPreferences preferences = context.getSharedPreferences(USER_PREFERENCES_KEY, Context.MODE_PRIVATE);
        return preferences.getStringSet(FAVORITES, new HashSet<String>());
    }

    public boolean isFavoritesOnly() { return favoritesOnly; }

    public void isFavoritesOnly(boolean favoritesOnly) {
        UserPreferences.favoritesOnly = favoritesOnly;
    }
}
```

So, how do we update our views?



There are a number of ways... we chose a simple one (that worked here). We'll get to that in a minute!

Let's prep for that by updating our SessionList ...

So, how do we update our views?

```
public List<Session> getAllSessions() {  
  
    if (sessions == null) {  
        synchronized (lock) {  
            sessions = service.getSessions();  
        }  
    }  
    addFavoritedStateToSessions(sessions);  
    return sessions;  
}  
  
private void addFavoritedStateToSessions(List<Session> sessions) {  
    List<Integer> favoriteSessionIds = userPreferences.getFavoriteSessionIds();  
    for (Session session : sessions) {  
        if (favoriteSessionIds.contains(session.getId())) {  
            session.isFavorited(true);  
        }  
    }  
}
```



Let's prep for
that by
updating our
SessionList

```
public List<Session> getFilteredSessionsFor(Day dayOfTheWeek) {  
    List<Session> sessionsForDay = new ArrayList<>();  
    for (Session session : getAllSessions()) {  
        if (dayOfTheWeek.isOn(session.getSessionStartTime())) {  
            sessionsForDay.add(session);  
        }  
    }  
    return filterByFavoritedPreference(sessionsForDay);  
}  
  
private List<Session> filterByFavoritedPreference(List<Session> sessionsForDay) {  
    boolean favoritesEnabled = userPreferences.isFavoritesOnly();  
    List<Session> allSessionsAfterFilters = new ArrayList<>();  
    for (Session session : sessionsForDay) {  
        if (favoritesEnabled) {  
            if (session.isFavorited()) {  
                allSessionsAfterFilters.add(session);  
            }  
        } else {  
            allSessionsAfterFilters.add(session);  
        }  
    }  
    return allSessionsAfterFilters;  
}
```

Let's recap before moving forward

Persistence in Android
More on Android Views



Section 6

Paved with Good Intent

So, back to updating those views (and then some!...)

```
public class FilterDialogFragment extends DialogFragment {  
    public static final String TAG = "FILTER_DIALOG";  
    private DialogInterface.OnClickListener listener;  
  
    public static FilterDialogFragment newInstance() {  
        return new FilterDialogFragment();  
    }  
  
    public void setOnClickListener(DialogInterface.OnClickListener listener) {  
        this.listener = listener;  
    }  
  
    @NotNull  
    @Override  
    public Dialog onCreateDialog(Bundle savedInstanceState) {  
        final UserPreferences userPreferences = CodeMashApplication.userPreferencesInstance();  
        int favoritesOptionSelected = userPreferences.isFavoritesOnly() ? 1 : 0;  
  
        return new AlertDialog.Builder(getActivity())  
            .setTitle("Filter Sessions")  
            .setSingleChoiceItems(R.array.favorites, favoritesOptionSelected, (dialog, which) -> {  
                if (which == 1) {  
                    userPreferences.isFavoritesOnly(true);  
                } else {  
                    userPreferences.isFavoritesOnly(false);  
                }  
            })  
            .setPositiveButton(android.R.string.ok, listener)  
            .create();  
    }  
}
```



Let's modify our
Fragment and
Activity

So, back to updating those views (and then some!...)

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.menu_filter) {
        FilterDialogFragment filterDialogFragment = FilterDialogFragment.newInstance();
        filterDialogFragment.setOnClickListener(new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Intent intent = new Intent(MainActivity.this,MainActivity.class);
                startActivity(intent);
            }
        });
        filterDialogFragment.show(getSupportFragmentManager(), FilterDialogFragment.TAG);
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```



Let's modify our
Fragment and
Activity

So, back to updating those views (and then some!...)



Let's Run It!

But what about
the...

Shhhh! Let em'
learn the hard way!

So, back to updating those views (and then some!...)

Much better

```
@Override  
public void onClick(DialogInterface dialog, int which) {  
    Intent intent = new Intent(MainActivity.this,MainActivity.class);  
    intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);  
    startActivity(intent);  
}
```



Intents

Let's talk about intents a bit more...

To the docs!

<http://developer.android.com/reference/android/content/Intent.html>



Intents



I bet we can do some nifty things with these, like show details about our sessions!

```
@Override  
public void onListItemClick(ListView l, View v, int position, long id) {  
    Intent intent = new Intent(getActivity(), SessionDetailsActivity.class);  
    startActivity(intent);  
}
```

I love when stuff
blows up!



```
public class SessionDetailsActivity extends ActionBarActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
    }  
}
```

USB 100% 4G 11:32 AM



Twitter



LinkedIn



Trello



Google



Kids



Chrome



Play Music



Wallet



Unfortunately, CodeMash has stopped.

Un

OK



Email



Gmail



Phone



Camera

Apps

Yes! We took down
the whole
conference!



```
01-04 11:31:49.777 3361-3361/com.brbw.codemash E/AndroidRuntime: FATAL EXCEPTION: main
Process: com.brbw.codemash, PID: 3361
android.content.ActivityNotFoundException: Unable to find explicit activity class {com.brbw.codemash/com.brbw.codemash.controllers.SessionListFragment}. Make sure the name matches the one specified in .setActivityClass()
at android.app.Instrumentation.checkStartActivityResult(Instrumentation.java:1636)
at android.app.Instrumentation.execStartActivity(Instrumentation.java:1430)
at android.app.Activity.startActivityForResult(Activity.java:3629)
at android.app.Activity.startActivityForResult(Activity.java:3590)
at android.support.v4.app.FragmentActivity.startActivityFromFragment(FragmentActivity.java:826)
at android.support.v4.app.Fragment.startActivity(Fragment.java:896)
at com.brbw.codemash.controllers.fragments.SessionListFragment.onListItemClick(SessionListFragment.java:56)
at android.support.v4.app.ListFragment$2.onItemClick(ListFragment.java:58)
at android.widget.AdapterView.performItemClick(AdapterView.java:313)
at android.widget.AbsListView.performItemClick(AbsListView.java:1528)
at android.widget.AbsListView$PerformClick.run(AbsListView.java:3540)
at android.widget.AbsListView$3.run(AbsListView.java:5218)
at android.os.Handler.handleCallback(Handler.java:733)
at android.os.Handler.dispatchMessage(Handler.java:95)
at android.os.Looper.loop(Looper.java:146)
at android.app.ActivityThread.main(ActivityThread.java:5678)
at java.lang.reflect.Method.invokeNative(Native Method) <1 internal calls>
at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:1291)
at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:1107)
at dalvik.system.NativeStart.main(Native Method)
```

Don't forget to modify your manifest!



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.brbw.codemash" >

    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:name=".CodeMashApplication"
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="CodeMash"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".controllers.activities.MainActivity"
            android:label="CodeMash" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".controllers.activities.SessionDetailsActivity"/>
    </application>

</manifest>
```

Let's get some Session Info

```
public class SessionDetailsActivity extends ActionBarActivity {

    private static final String LOG_TAG = "SessionDetailsActivity";
    private static final String SESSION_EXTRA = "SESSION_EXTRA";

    public static Intent newIntentToStart(Context context, Session session) {
        Intent intent = new Intent(context, SessionDetailsActivity.class);
        intent.putExtra(SESSION_EXTRA, session);
        return intent;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Session session = (Session) getIntent().getSerializableExtra(SESSION_EXTRA);
        Log.d(LOG_TAG, session.toString());
    }
}
```

```
@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    Session session = (Session) getListView().getItemAtPosition(position);
    Intent intent = SessionDetailsActivity.newIntentToStart(getActivity(), session);
    startActivity(intent);
}
```

Let's display it!

activity_single_fragment.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:orientation="vertical">
7
8      <FrameLayout
9          android:id="@+id/fragment_container"
10         android:layout_width="match_parent"
11         android:layout_height="match_parent" />
12
13  </LinearLayout>
```

fragment_session_details.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical">
6
7      <TextView
8          android:id="@+id/session_title"
9          android:layout_width="match_parent"
10         android:layout_height="match_parent" />
11
12  </LinearLayout>
```

Let's display it!

```
C SessionDetailsFragment.java x
1  import android.view.View;
2  import android.view.ViewGroup;
3
4  import com.brbw.codemash.R;
5  import com.brbw.codemash.models.Session;
6  import com.brbw.codemash.util.ViewHelper;
7
8  public class SessionDetailsFragment extends Fragment {
9
10     private static final String SESSION_ARGS = "SESSION_ARGS";
11
12     public static SessionDetailsFragment newInstance(Session session) {
13         SessionDetailsFragment fragment = new SessionDetailsFragment();
14         Bundle args = new Bundle();
15         args.putSerializable(SESSION_ARGS, session);
16         fragment.setArguments(args);
17         return fragment;
18     }
19
20     @Override
21     public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
22         View view = inflater.inflate(R.layout.fragment_session_details, container, false);
23
24         if (this.hasArgsFor(SESSION_ARGS)) {
25             Session session = (Session) getArguments().getSerializable(SESSION_ARGS);
26             new ViewHelper(view)
27                 .setText(R.id.session_title, session.getTitle());
28         }
29         return view;
30     }
31
32     private boolean hasArgsFor(String args) {
33         return getArguments() != null && getArguments().containsKey(args);
34     }
35 }
```

Let's display it!

```
public class SessionDetailsActivity extends ActionBarActivity {

    private static final String LOG_TAG = "SessionDetailsActivity";
    private static final String SESSION_EXTRA = "SESSION_EXTRA";

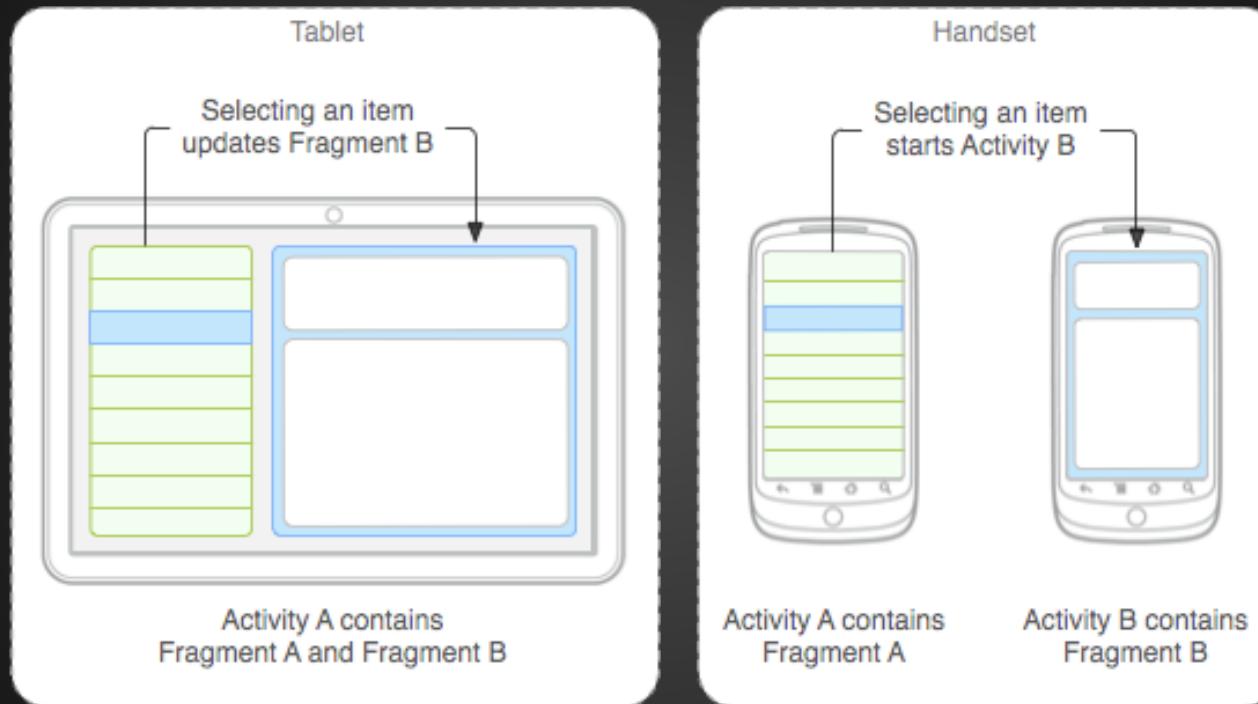
    public static Intent newIntentToStart(Context context, Session session) {
        Intent intent = new Intent(context, SessionDetailsActivity.class);
        intent.putExtra(SESSION_EXTRA, session);
        return intent;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_single_fragment);

        Session session = (Session) getIntent().getSerializableExtra(SESSION_EXTRA);

        getSupportFragmentManager()
            .beginTransaction()
            .replace(R.id.fragment_container, SessionDetailsFragment.newInstance(session))
            .commit();
    }
}
```

Why that level of indirection?



While we're here, lets fire off a quick Implicit Intent!



```
SessionDetailsActivity.java
29
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_single_fragment);
34         getSupportActionBar().setDisplayShowTitleEnabled(true);
35         getSupportActionBar().setDisplayHomeAsUpEnabled(true);
36
37         session = (Session) getIntent().getSerializableExtra(SESSION_EXTRA);
38
39         getSupportFragmentManager()
40             .beginTransaction()
41             .replace(R.id.fragment_container, SessionDetailsFragment.newInstance(session))
42             .commit();
43
44     @Override
45     public boolean onCreateOptionsMenu(Menu menu) {
46         getMenuInflater().inflate(R.menu.session_details, menu);
47         return true;
48     }
49
50     @Override
51     public boolean onOptionsItemSelected(MenuItem item) {
52         if (R.id.session_share == item.getItemId()) {
53             startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(buildTweet())));
54             return true;
55         } else if (android.R.id.home == item.getItemId()) {
56             onBackPressed();
57         }
58         return super.onOptionsItemSelected(item);
59     }
60
61     private String buildTweet() {
62         String message = String.format(getString(R.string.twitter_message), session.getTitle());
63         String twitterUrl = getString(R.string.twitter_url);
64         return String.format(twitterUrl, urlEncode(message));
65     }
66
67 }
```

While we're here, lets fire off a quick Implicit Intent!

```
session_details.xml x
1 <?xml version="1.0" encoding="utf-8"?>
2
3 <menu xmlns:android="http://schemas.android.com/apk/res/android"
4   xmlns:app="http://schemas.android.com/apk/res-auto">
5   <item
6     android:id="@+id/session_share"
7     android:icon="@android:drawable/ic_menu_share"
8     android:title="share"
9     app:showAsAction="always" />
10 </menu>
```



While we're here, lets fire off a quick Implicit Intent!



```
strings.xml x

Edit translations for all locales in the translations editor.

1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3
4      <string name="app_name">CodeMash</string>
5      <string name="action_settings">Settings</string>
6      <string name="tuesday">tues</string>
7      <string name="wednesday">wed</string>
8      <string name="thursday">thurs</string>
9      <string name="friday">fri</string>
10     <string name="empty_session_list_message">No sessions available!</string>
11     <string name="filter">filter</string>
12     <string name="filter_title">Filter Sessions</string>
13
14     <array name="favorites">
15         <item>Show All</item>
16         <item>Show Favorites Only</item>
17     </array>
18
19     <string name="twitter_message">Checking out %s at #CodeMash!</string>
20     <string name="twitter_url">https://twitter.com/intent/tweet?text=%s</string>
21
22 </resources>
23
```

And that's it



Let's recap before moving forward

- Intents! (They're magical)
- Adding Fragments to Activities & even though YAGNI, AUF!
- (and don't actually drop a microphone. They're expensive and a DJ will likely get ticked.)



Section 7

Resources, Clean-up, and PUSH

Resources



So, what other things are in the res directory?

A screenshot of a file explorer window showing the structure of an Android application's resources. The 'res' directory is expanded, revealing subfolders for different screen densities: drawable, drawable-hdpi, drawable-mdpi, drawable-xhdpi, and drawable-xxhdpi. It also contains layout, menu, and values folders. The values folder is further expanded to show colors.xml, dimens.xml, strings.xml, and styles.xml files, along with a values-w820dp folder and the AndroidManifest.xml file.

```
Code in master application
└── res
    ├── drawable
    ├── drawable-hdpi
    ├── drawable-mdpi
    ├── drawable-xhdpi
    ├── drawable-xxhdpi
    ├── layout
    ├── menu
    └── values
        ├── colors.xml
        ├── dimens.xml
        ├── strings.xml
        ├── styles.xml
        └── values-w820dp
    └── AndroidManifest.xml
```

Drawable, Dimens, Shapes, and Colors!



Drawable



So, what can we do with
Drawable?

What the heck is with the mdpi,
hdpi, xhdpi, and xxhdpi?

Drawable



What the heck is
with the mdpi,
hdpi, xhdpi, and
xxhdpi?

- *xlarge* screens are at least 960dp x 720dp
- *large* screens are at least 640dp x 480dp
- *normal* screens are at least 470dp x 320dp
- *small* screens are at least 426dp x 320dp

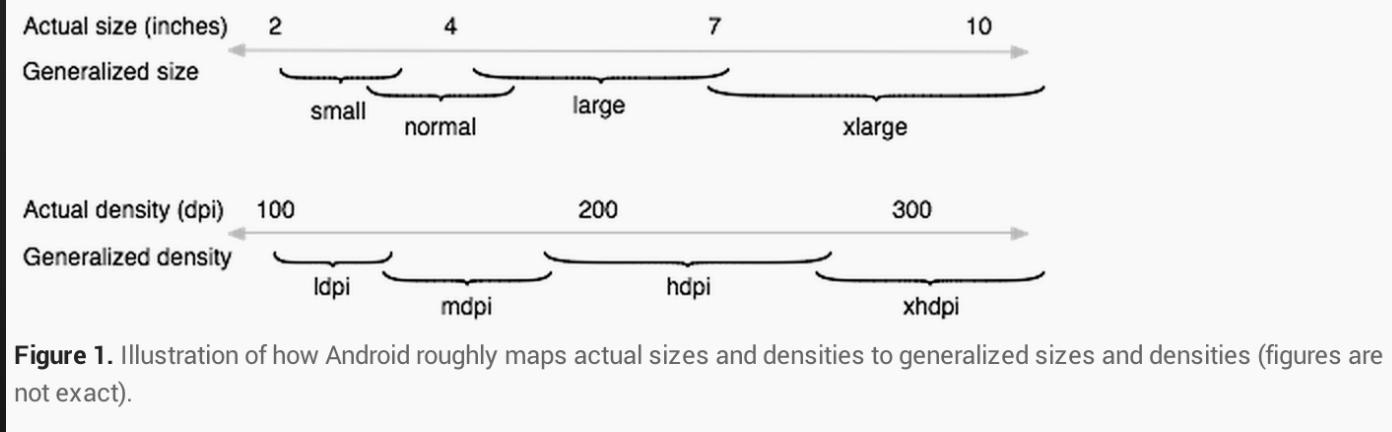


Figure 1. Illustration of how Android roughly maps actual sizes and densities to generalized sizes and densities (figures are not exact).

Drawable

```
    android:layout_width="match_parent"
    android:layout_height="@dimen/detail_view_image_height"
    android:orientation="horizontal"
    android:background="@drawable/session_bg">
```



But, we only have to
tell it once.

Android will pick the
best option for us...

Shapes



So, how do we
make this guy?



Shapes



```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:bottomRightRadius="4dp" android:topRightRadius="4dp"/>
</shape>
```

```
<LinearLayout
    android:id="@+id/adapter_session_border"
    android:layout_width="10dip"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:background="@drawable/border_rounder_corners"
/>
```

Notice, you'll find
your shapes in
@drawable



Colors and Dimens



Why do we have
these?

Abstraction



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="codemash_tuesday">#FFAF00</color>
    <color name="codemash_wednesday">#12ACE0</color>
    <color name="codemash_thursday">#BB71DE</color>
    <color name="codemash_friday">#89BF00</color>
    <color name="codemash_all">#7A7A7A</color>
    <color name="codemash_white">#fffffe</color>
    <color name="codemash_black">#322f19</color>
    <color name="transparent_black">#60000000</color>
</resources>
```

```
<!-- Default screen margins, per the Android Design guidelines. -->
<dimen name="activity_horizontal_margin">16dp</dimen>
<dimen name="activity_vertical_margin">16dp</dimen>

<dimen name="cardview_corner_radius">4dp</dimen>

<dimen name="gridview_item_width">120dp</dimen>
<dimen name="gridview_item_height">@dimen/gridview_item_width</dimen>

<dimen name="detail_view_image_height">200dp</dimen>
</resources>
```

Resources



We can also
reference all these
from the code!

```
switch (calendar.get(Calendar.DAY_OF_WEEK)) {  
    case Calendar.TUESDAY:  
        resId = R.color.codemash_tuesday;  
        break;  
    case Calendar.WEDNESDAY:  
        resId = R.color.codemash_wednesday;  
        break;  
    case Calendar.THURSDAY:  
        resId = R.color.codemash_thursday;  
        break;  
    case Calendar.FRIDAY:  
        resId = R.color.codemash_friday;  
        break;  
    default:  
        resId = R.color.codemash_all;  
}
```

```
GradientDrawable borderBackground = (GradientDrawable)getContext().  
    getResources().getDrawable(R.drawable.border_rounder_corners);
```

A few useful links

<http://romannurik.github.io/AndroidAssetStudio/>

<http://www.norio.be/android-feature-graphic-generator/>

There's also this guy...



@paulfresty

And PUSH!

Section 8

Let's go Nuts

Since We have the Time:

Nothing cements new learning like doing it:

Challenge 1: Create a Speaker List (perhaps add a new tab on the view pager?)

Challenge 2: Create a Speaker Detail from the Speaker List

Challenge 3: Try building some navigation for Speakers

Thanks... We are now very tired, and will be drinking for a while...

However, no matter where you got to, or if you have any questions at all, we'll be around all week. Hit us up on the tweeter if you want... We'd be happy to bust open the lappies and pair any time.