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

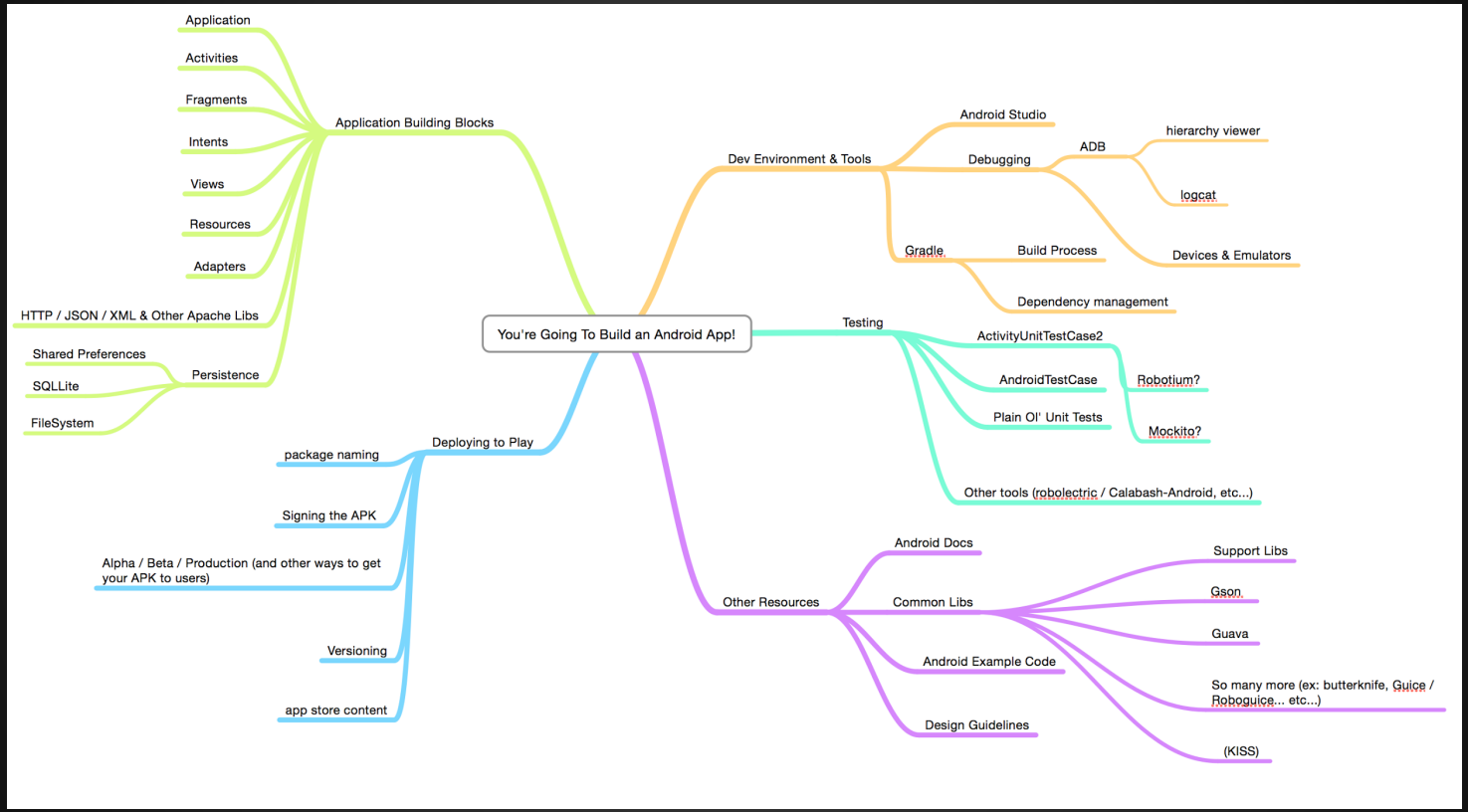
Section 1 - Android, Android Studio, and the Play Store
Section 2 - Activities, Fragments, Lifecycles, and more
Section 3 - Data, Permissions, Tasks

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

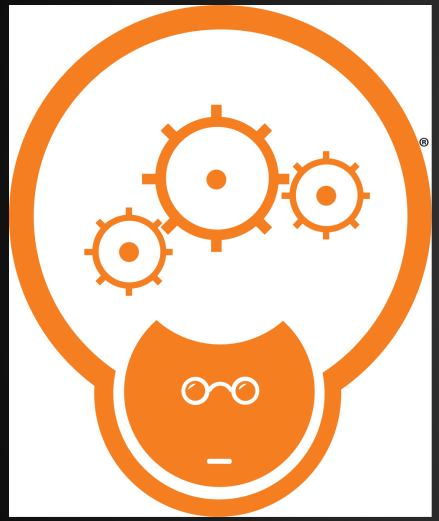
Section 1
What are you doing, Dave?

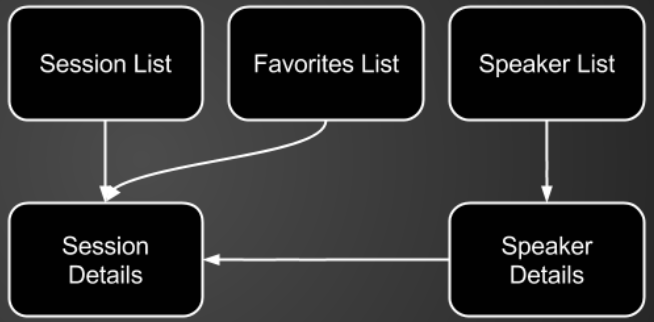
What you're in for



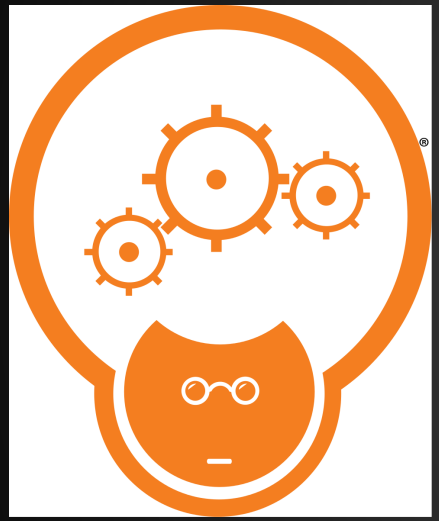
What we wish we had time to cover
(and we forgot a few too)

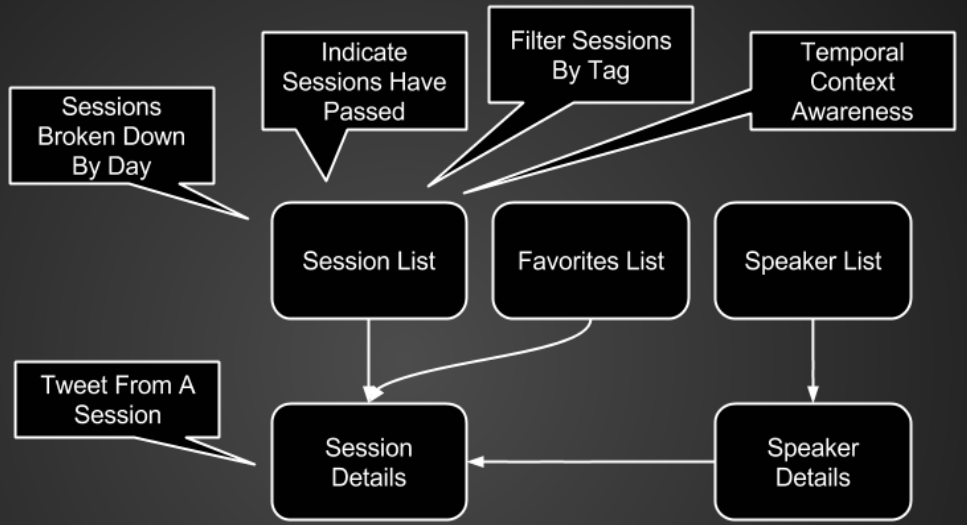
Where we're heading





Where we're heading






Sessions
TUE
WED
THUR
Lorem ipsum dolor sit amet
Dapibus, neque id cursus faubicus, tortor neque egestas
Speakers
Favorites
Filter By Tags
Java
Android
.NET
Ruby on Rails
CANCEL
OK

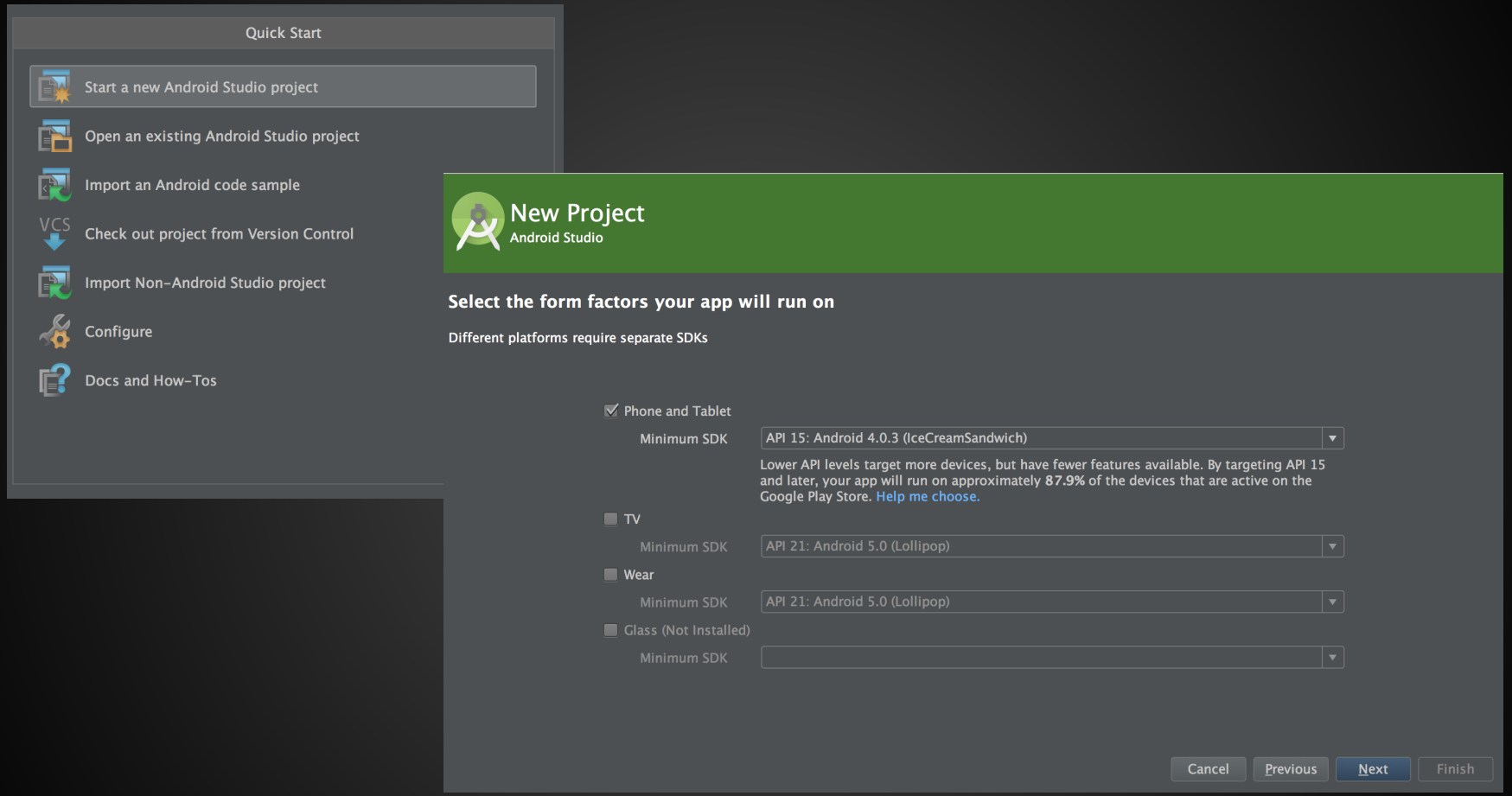

Speaker Name
Session details go here... Suspendisse utna nibh, viverra non, semper suscipit, posuere a, pede.
Speaker Session 1
Speaker Session 2
Session Title
Session details go here... Suspendisse utna nibh, viverra non, semper suscipit, posuere a, pede.

Sacrifices were made.

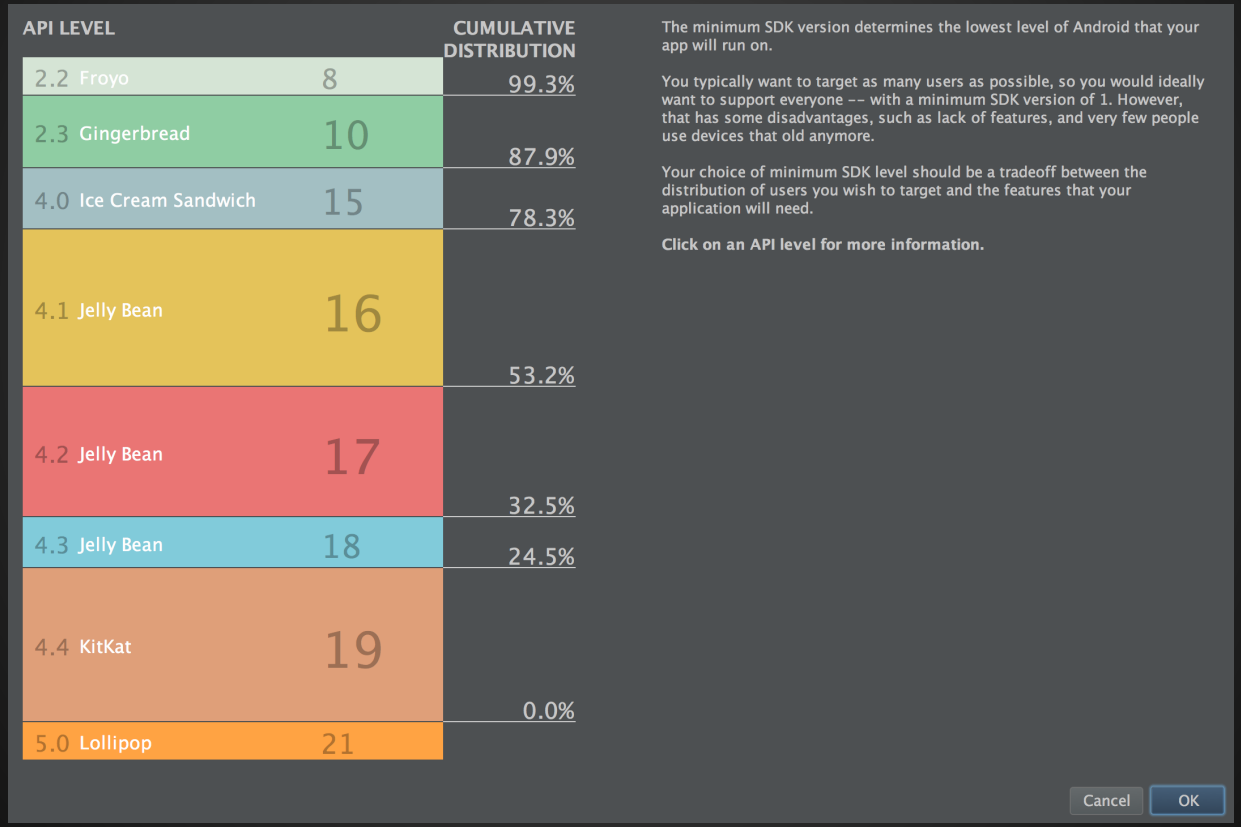
What you should have already

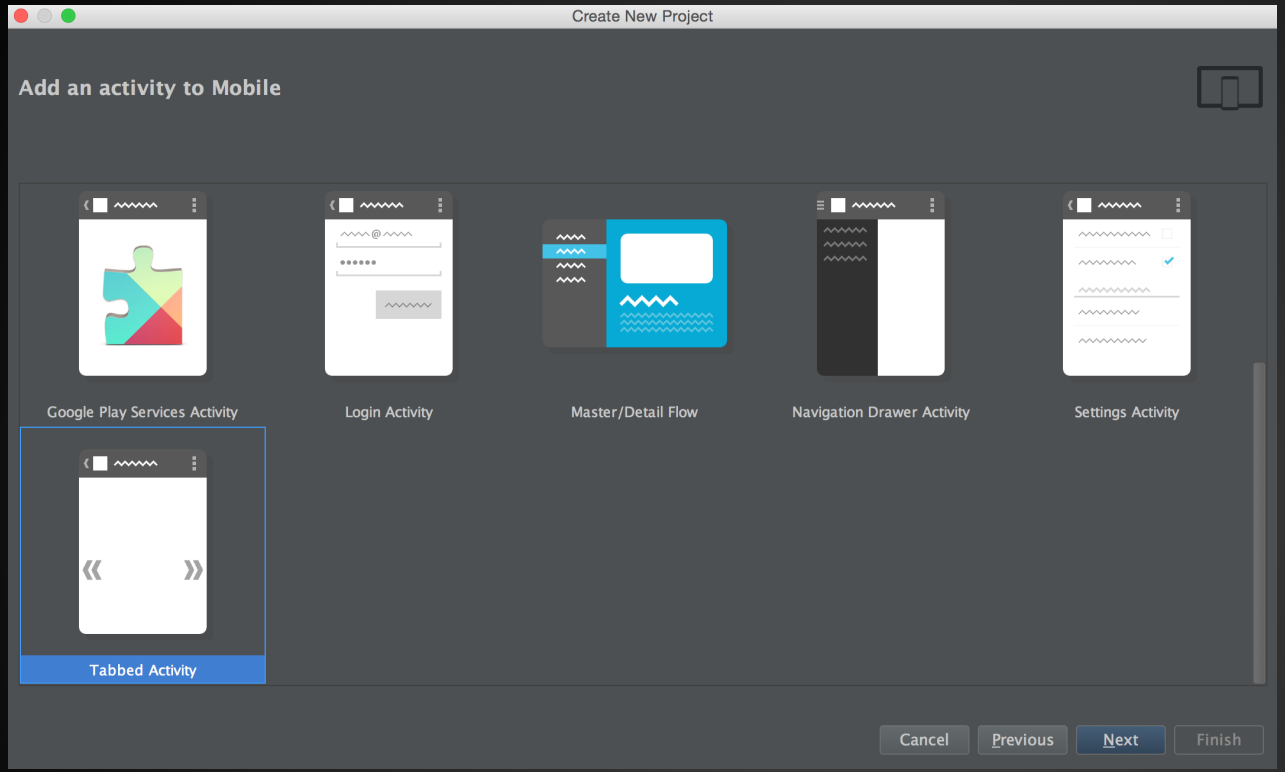
- Java JDK 7
- Android Studio
- Android SDK + extras
- An Android Emulator or Android Device
- (and it will help to have Git)

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
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...





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

Create New Project

Choose options for your new file

Tabbed Activity

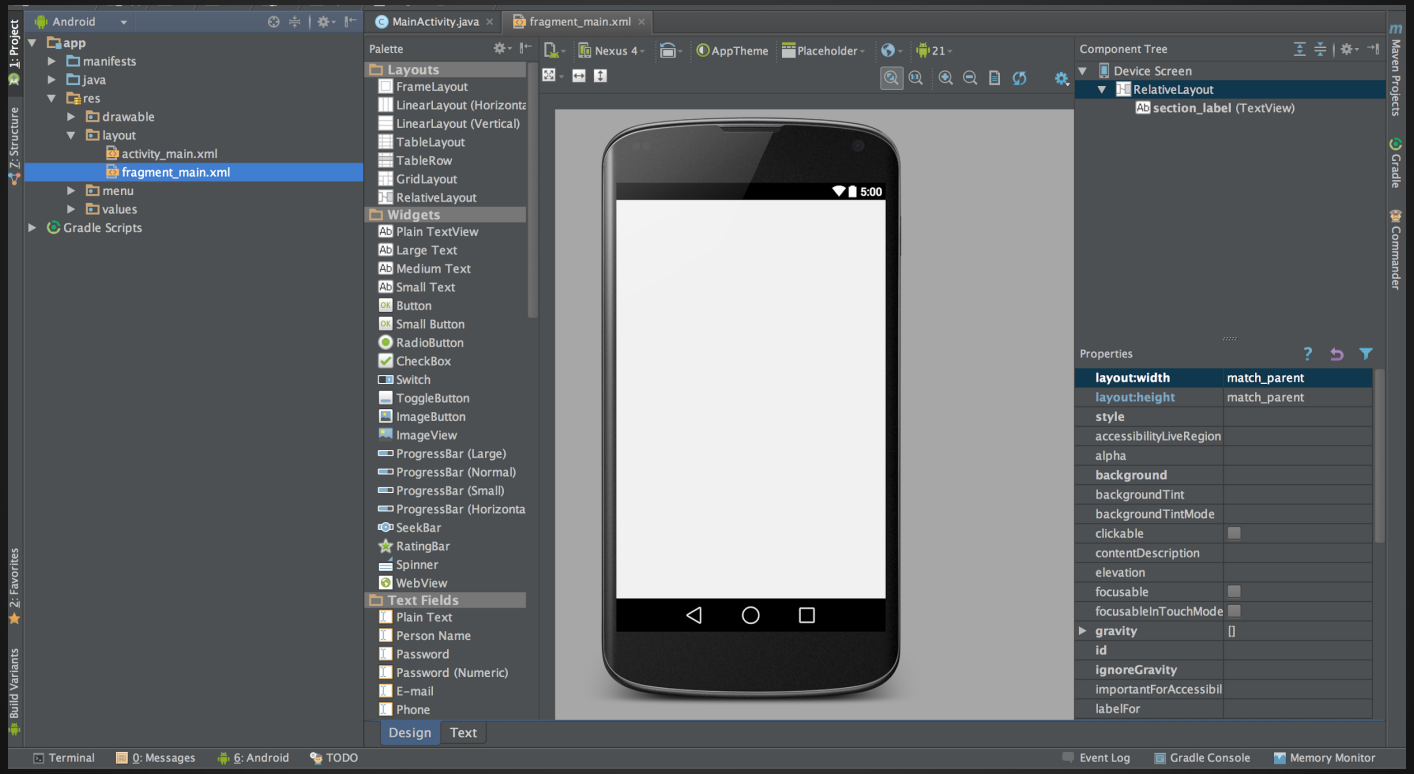
Creates a new blank activity, with an action bar and navigational elements such as tabs or horizontal swipe.

Activity Name:	MainActivity
Layout Name:	activity_main
Fragment Layout Name:	fragment_main
Title:	MainActivity
Menu Resource Name:	menu_main
Navigation Style:	Action Bar Tabs (with ViewPager)

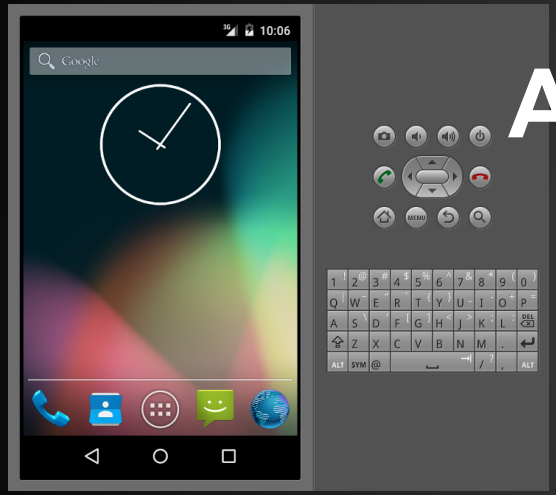
Additional features to include, such as a fragment, swipe views, or a navigation drawer

Cancel | Previous | Next | Finish

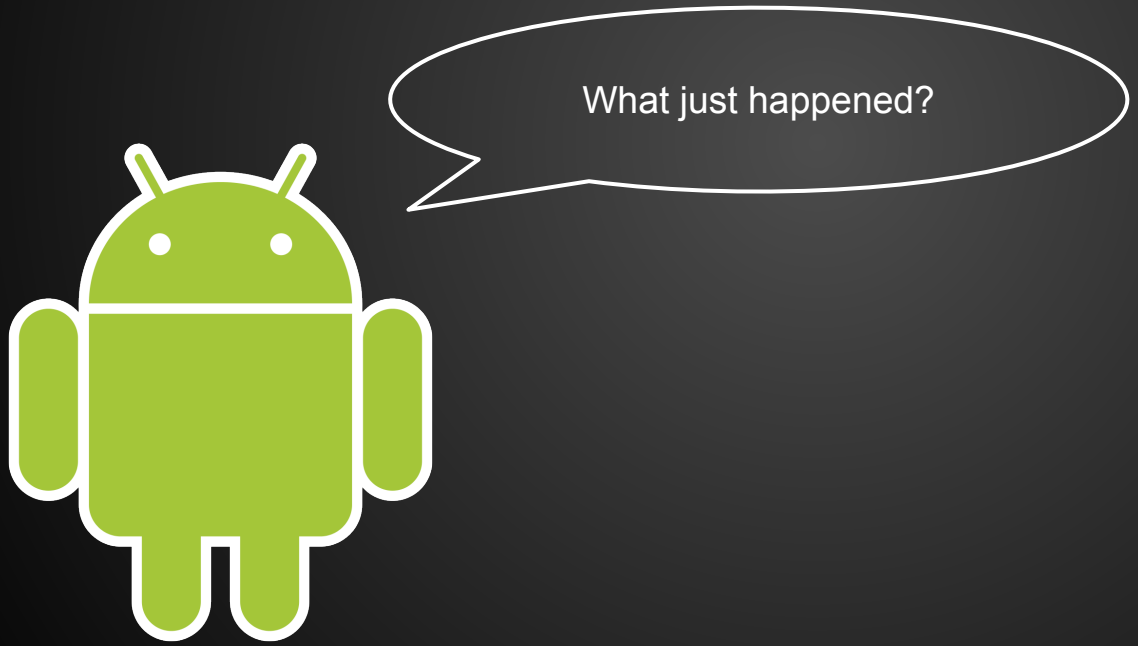
And if all went well...



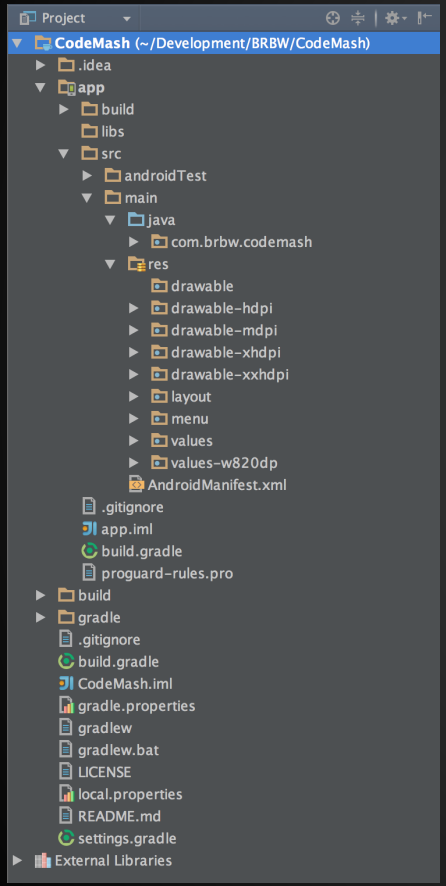
Bring Forth the Android Emulators or Devices!



So hopefully you got up and running...

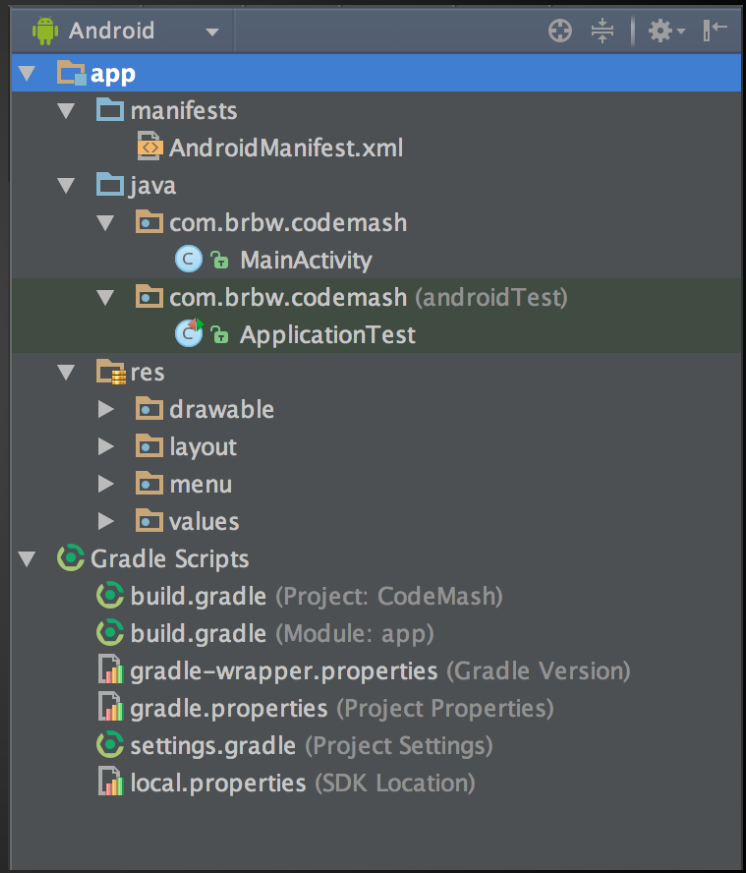


What Android Gave you out of the box



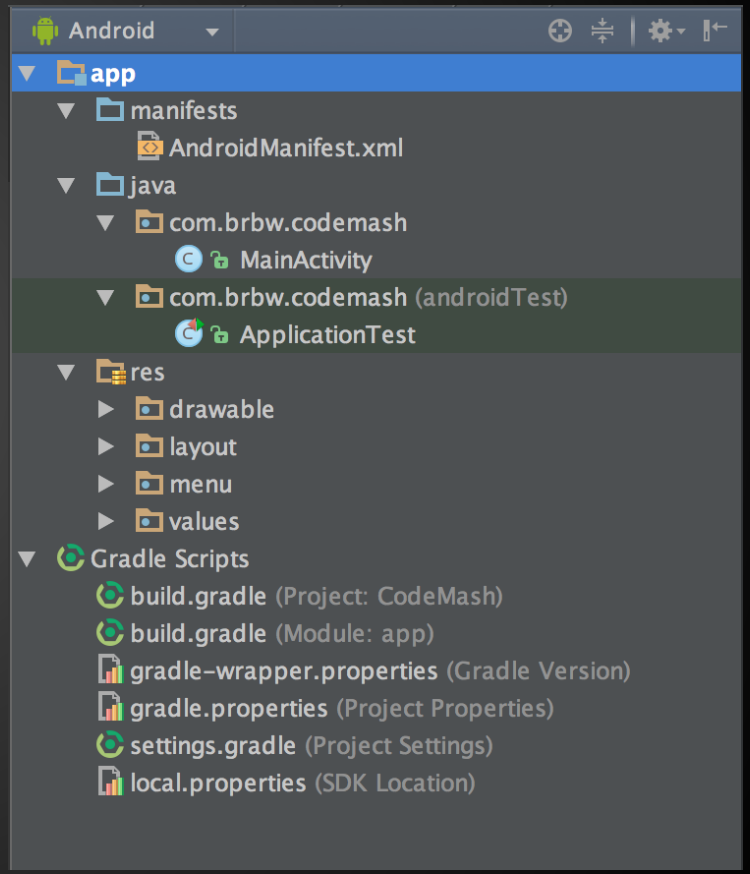
The whole enchilada

The stuff you'll care most about

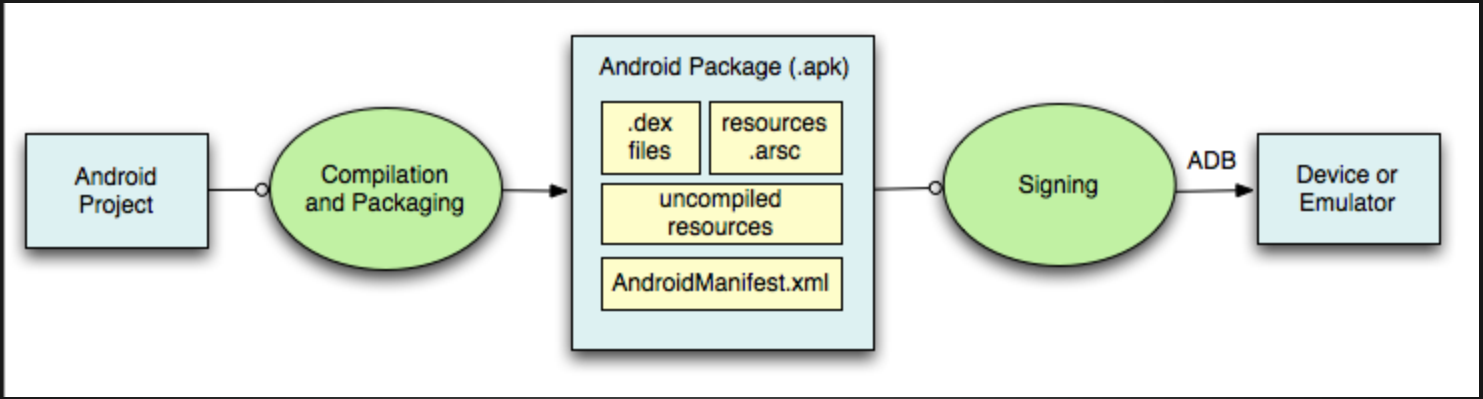


What Android Gave you out of the box

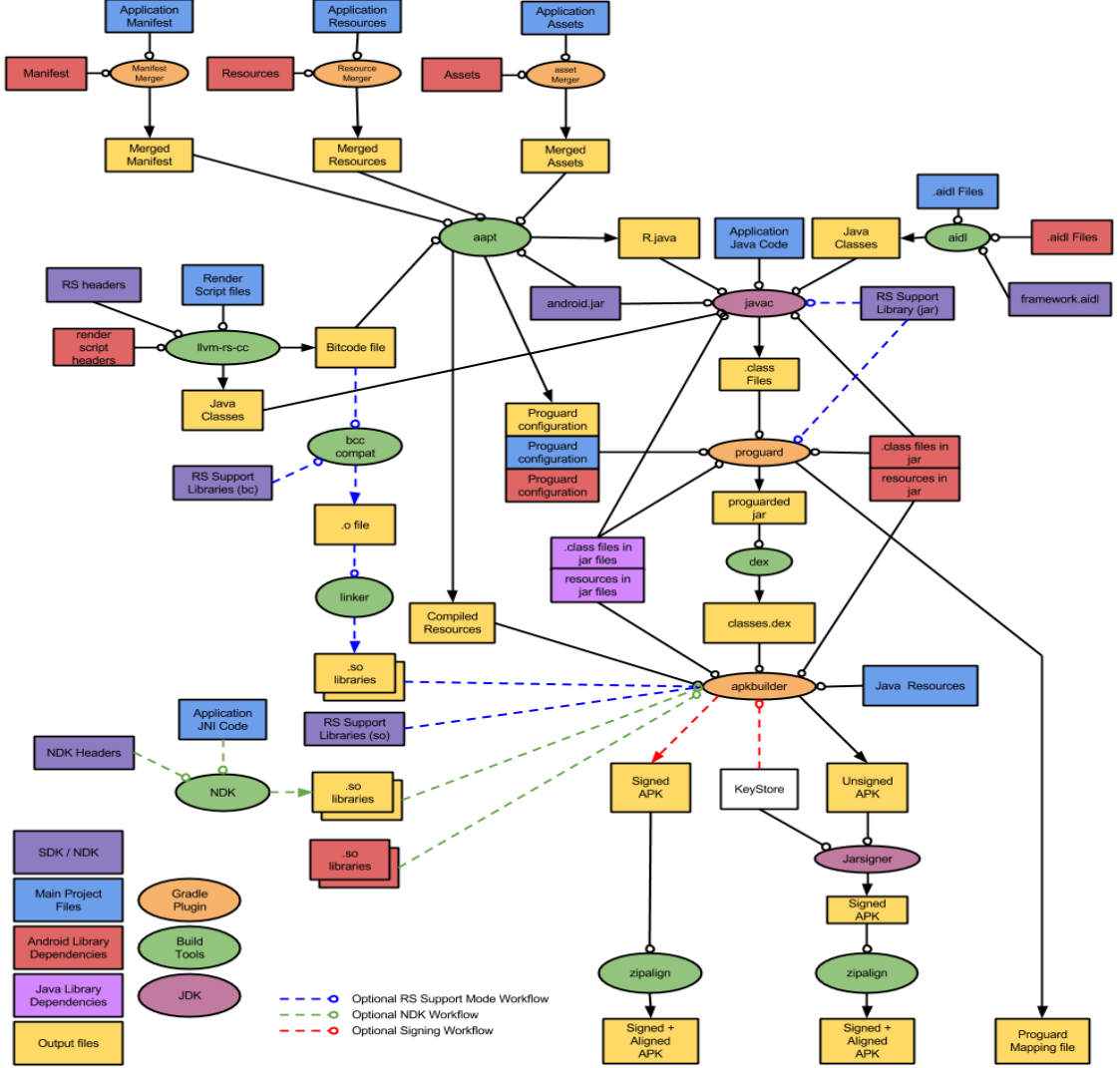
- app/src/main/AndroidManifest.xml
- app/src/main/java/*
- app/src/androidTest/java/*
- app/src/main/res/*
- app/build.gradle



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/

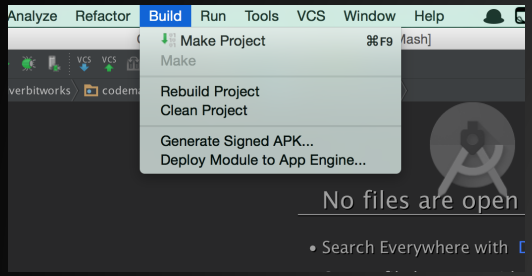
Our production checklist:

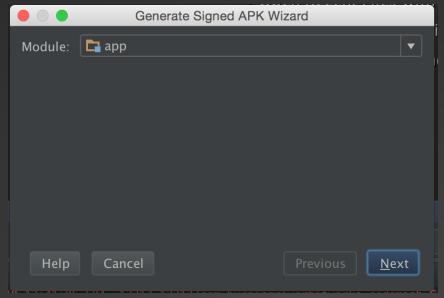
- ❑ A signed APK
- ❑ Store Listing
  - ❑ Title
  - ❑ Short & Full Descriptions
  - ❑ Screenshots *we can take those in a bit
  - ❑ High-res icon (512x512 32 bit png)
  - ❑ Feature Graphic (1024x500 jpeg or png)
  - ❑ and a few other items...

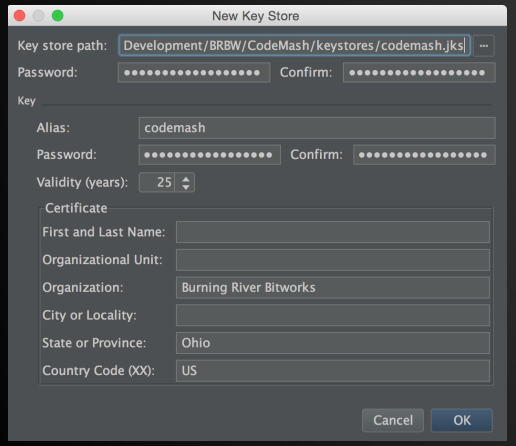
Let’s deploy what we have and discuss options for the others...

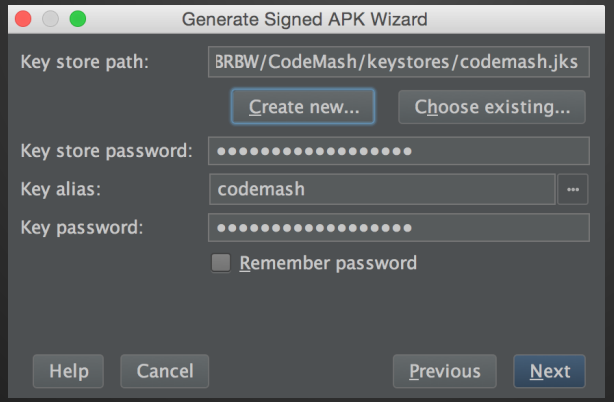
- ❏ A signed APK
- ❏ Store Listing
  - ❏ Title
  - ❏ Short & Full Descriptions
  - ❏ Screenshots *we can take those in a bit
  - ❏ High-res icon (512x512 32 bit png)
  - ❏ Feature Graphic (1024x500 jpeg or png)
  - ❏ and a few other items...

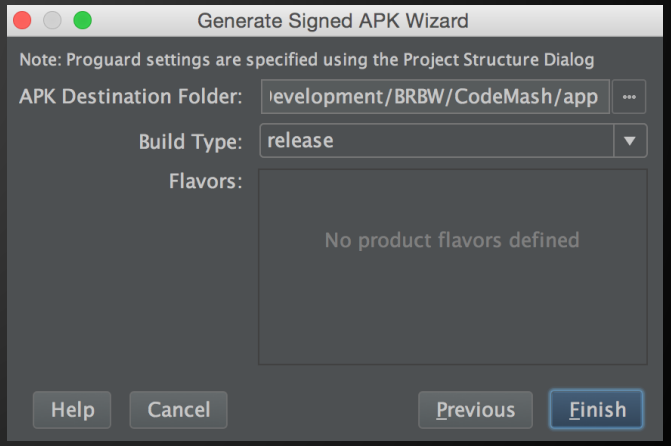
Signing your app











Once you publish, don’t lose your password!

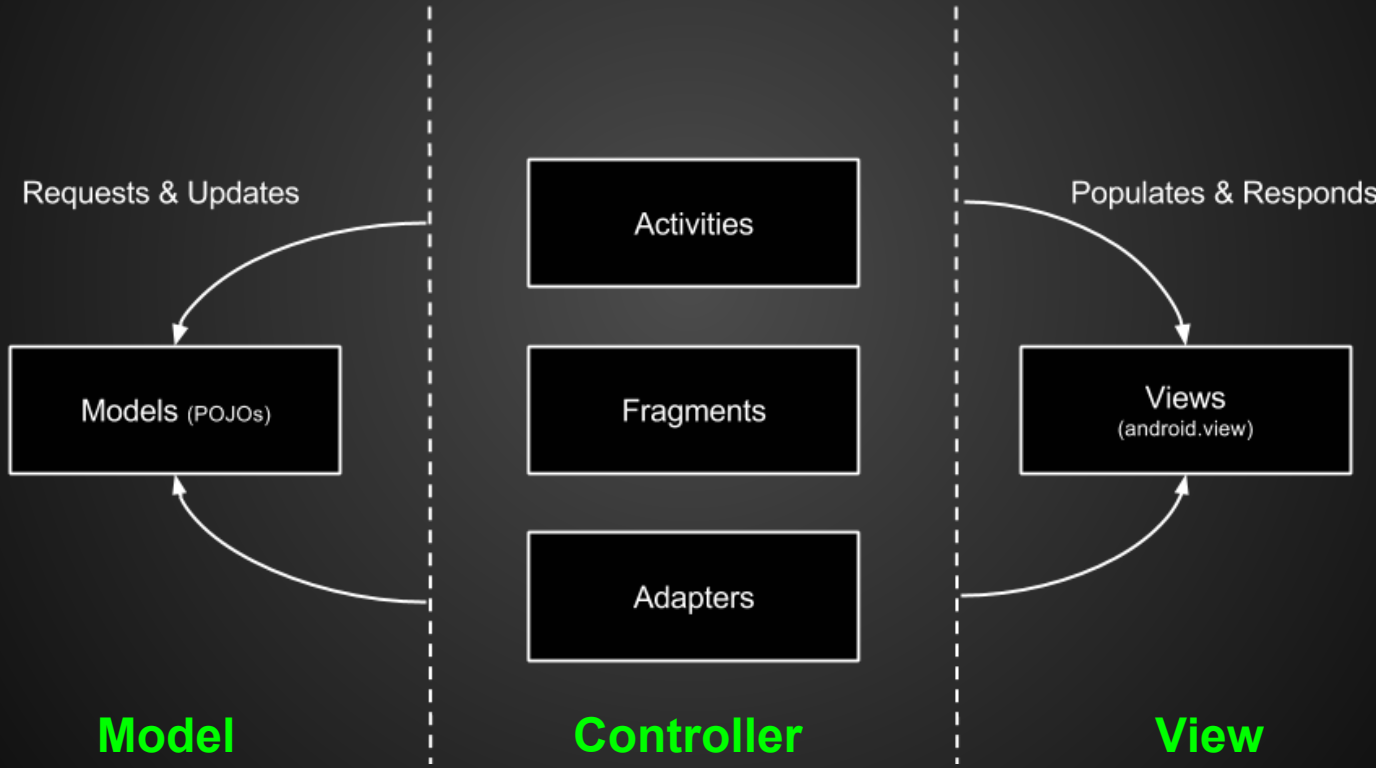
(you can change it if you know it, but if you don’t users will have to re-install)

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

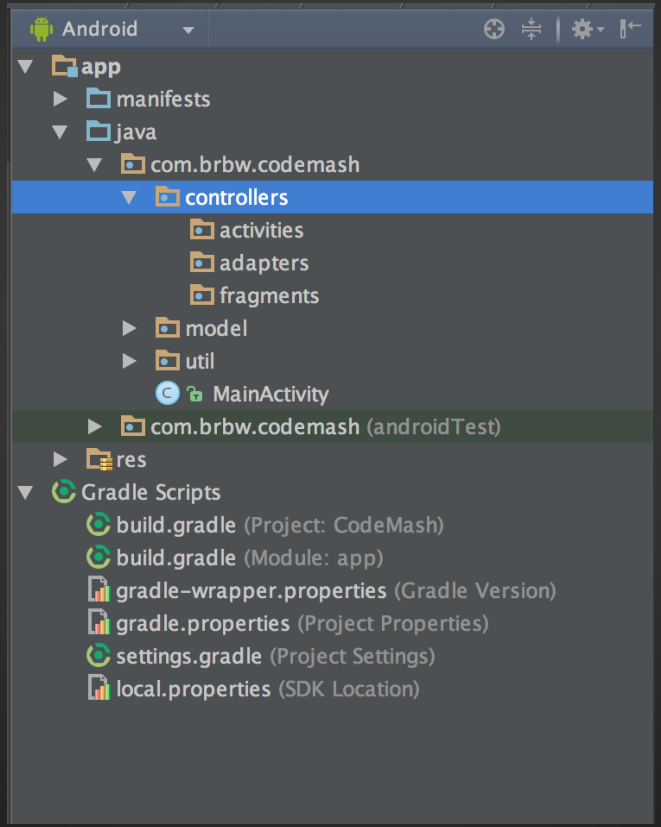
Section 2
Android Sledge-O-Matic

Raise your hand if you “know MVC”

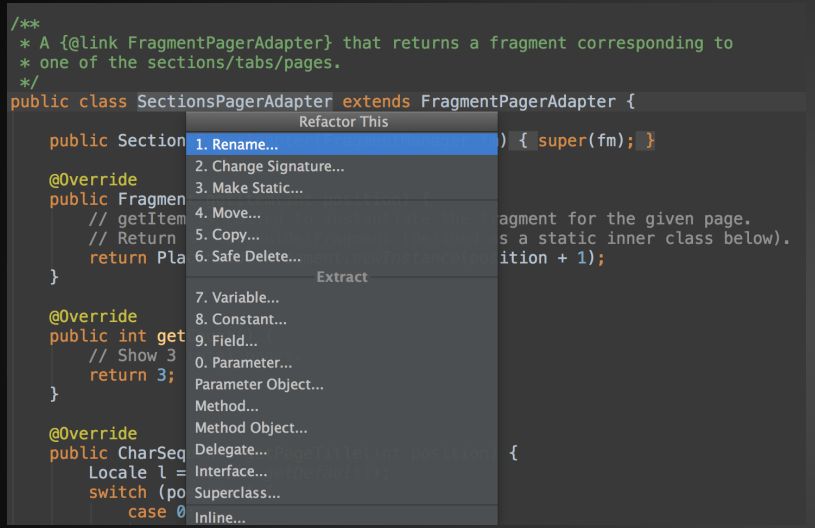
Android “MVC”

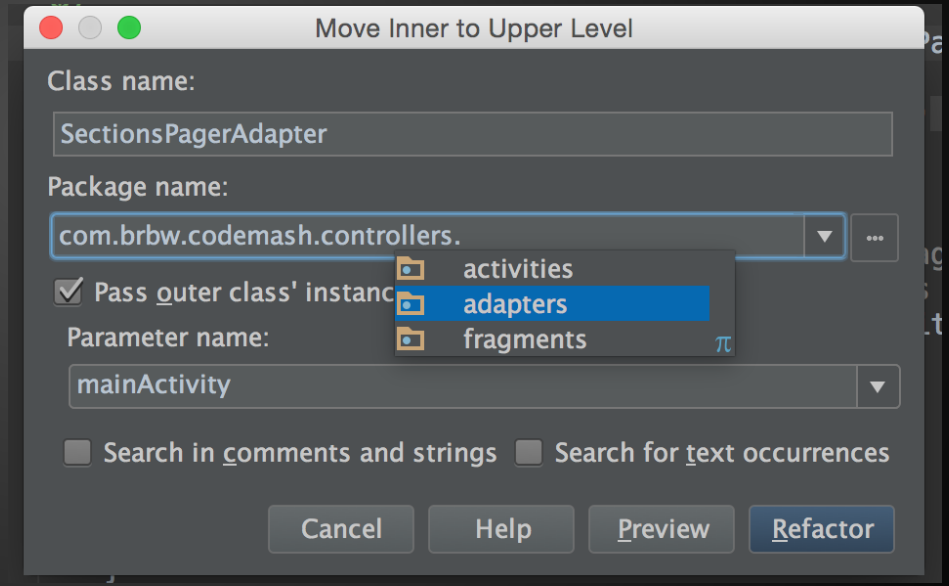

Requests & Updates
Activities
Populates & Responds
Models (POJOs)
Fragments
Views
(android.view)
Adapters
Model
Controller
View

Let’s start with a little refactor...



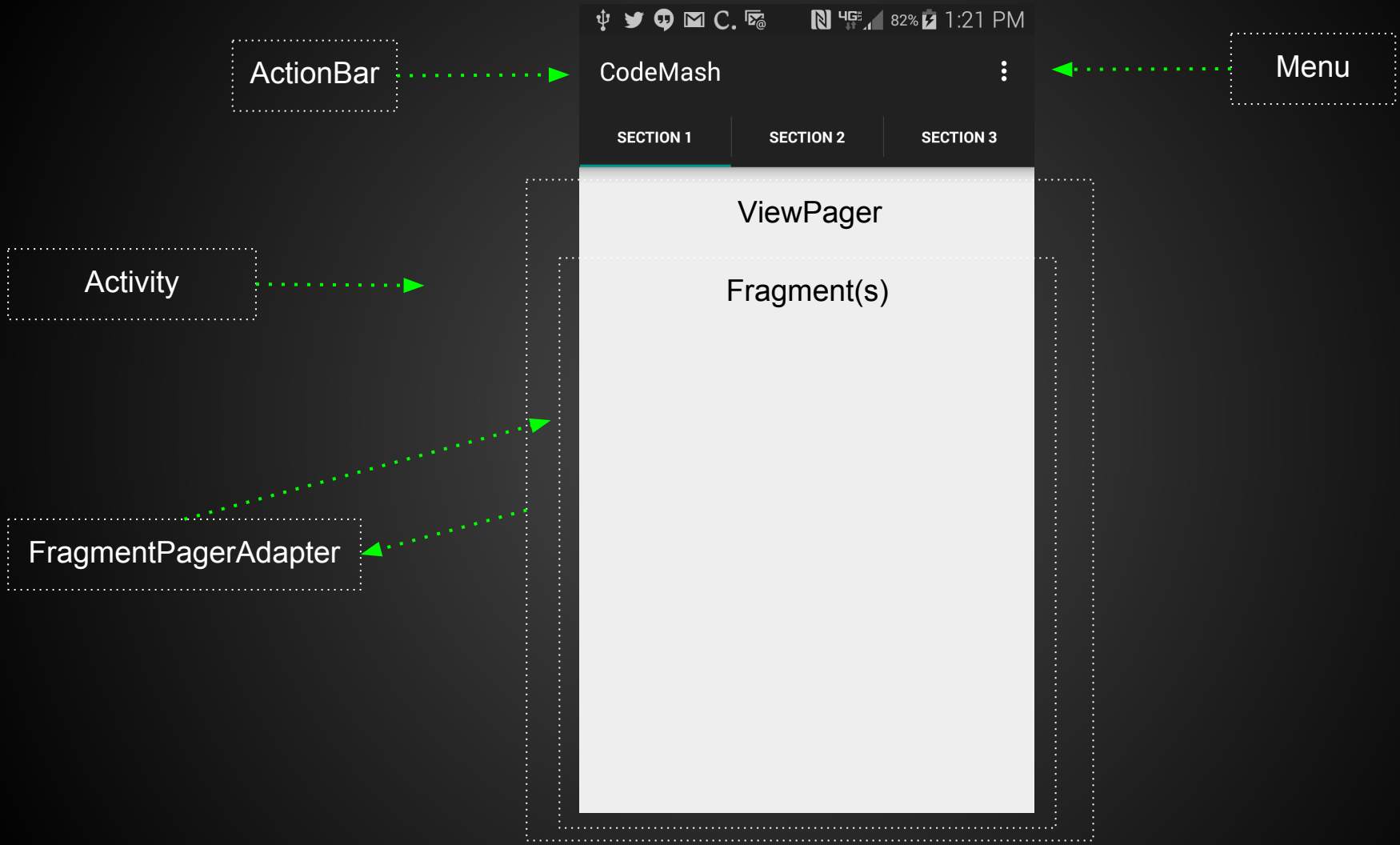
Move the adapter & fragment out



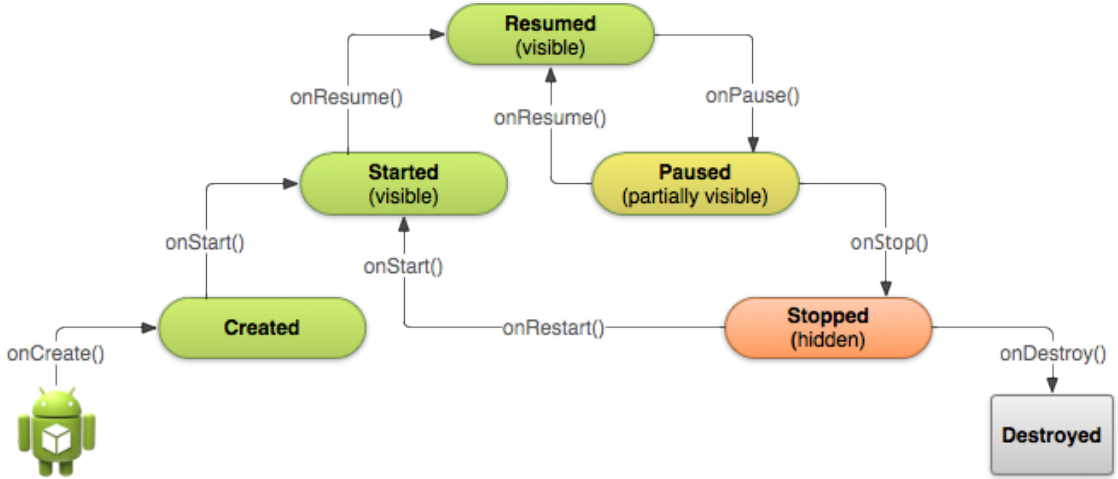


Now let’s take a few minutes to discuss each and their role here

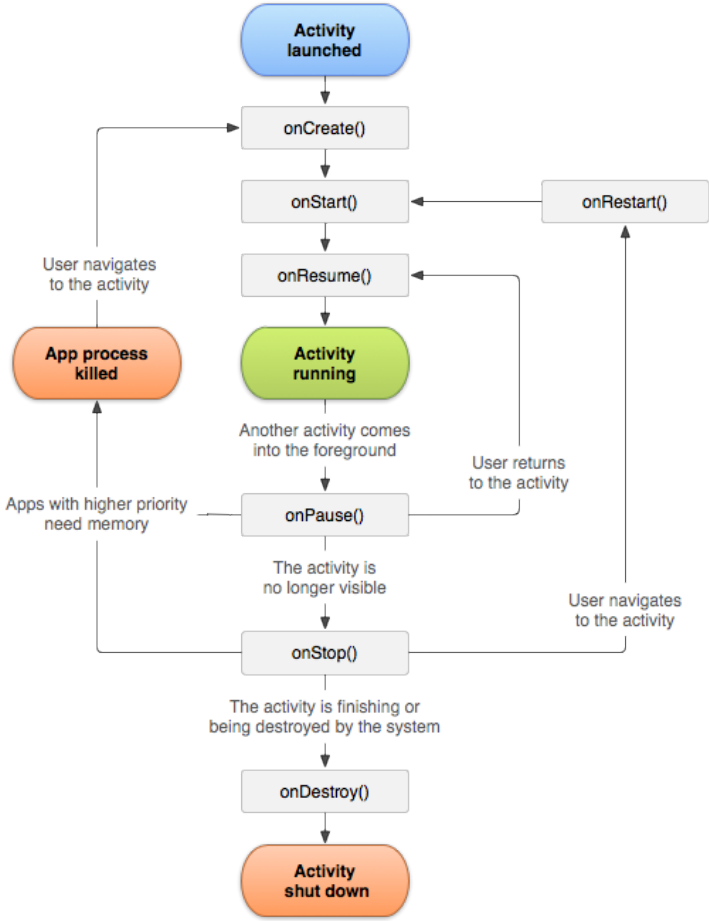
- MainActivity
- SectionsPagerAdapter
- PlaceHolderFragment


ActionBar
CodeMash
Menu
SECTION 1
SECTION 2
SECTION 3
ViewPager
Fragment(s)
Activity
FragmentPagerAdapter

Why's everything in onCreate() ?

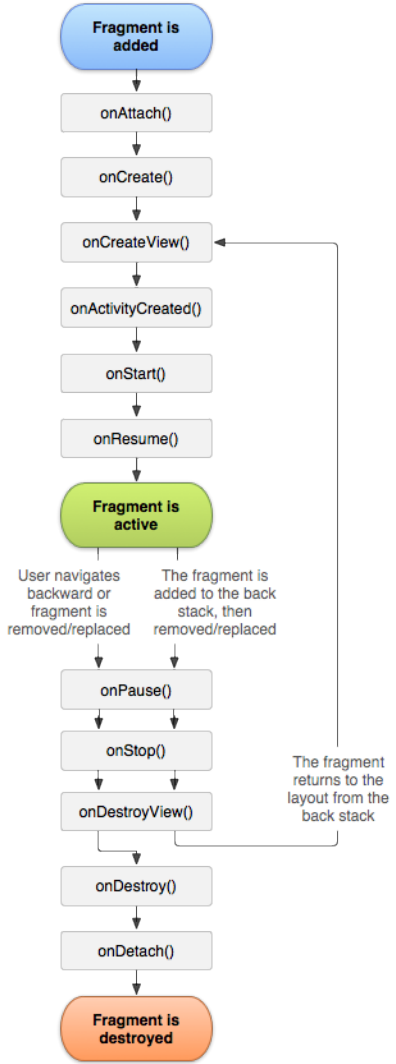


Source: http://developer.android.com/training/basics/activity-lifecycle/starting.html



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");
```

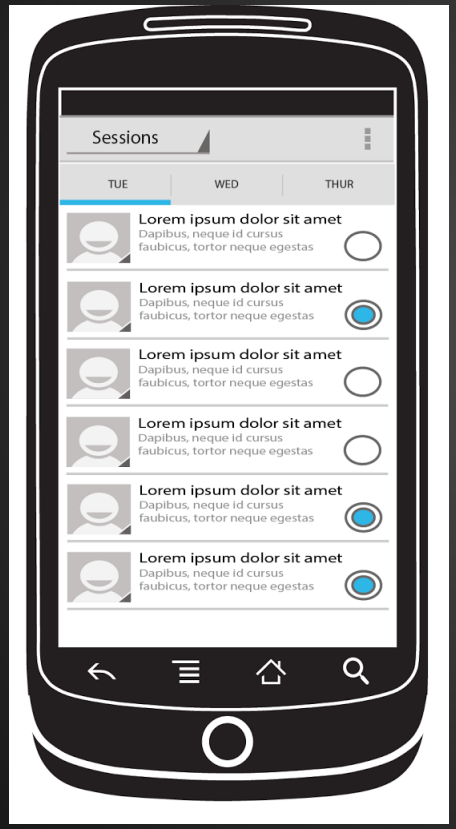
Just to see this in action...

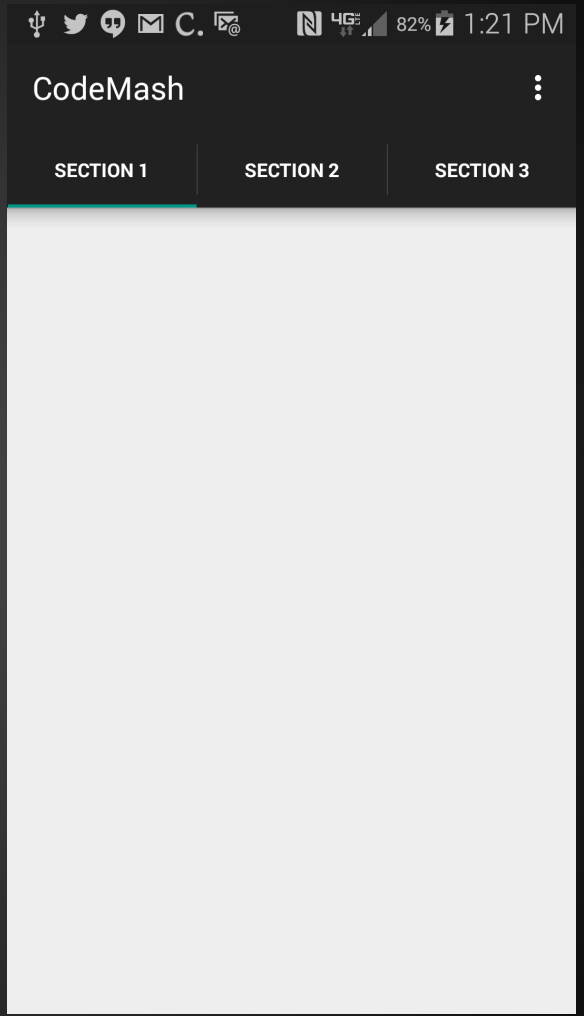
PlaceholderFragment.java

```
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!





Modifying the SectionsPagerAdapter

```java
public class SectionsPagerAdapter extends FragmentPagerAdapter {

    private MainActivity mainActivity;

    public SectionsPagerAdapter(MainActivity mainActivity, FragmentManager fm) {
        super(fm);
        this.mainActivity = mainActivity;
    }
```

```java
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...

PlaceholderFragment.java | SectionsPagerAdapter.java | Day.java

```
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; }
18
19     public boolean isOn(Date date) {
20         Calendar calendar = Calendar.getInstance();
21         calendar.setTime(date);
22         return calendarDayOfWeek == calendar.get(Calendar.DAY_OF_WEEK);
23     }
24 }
25
```

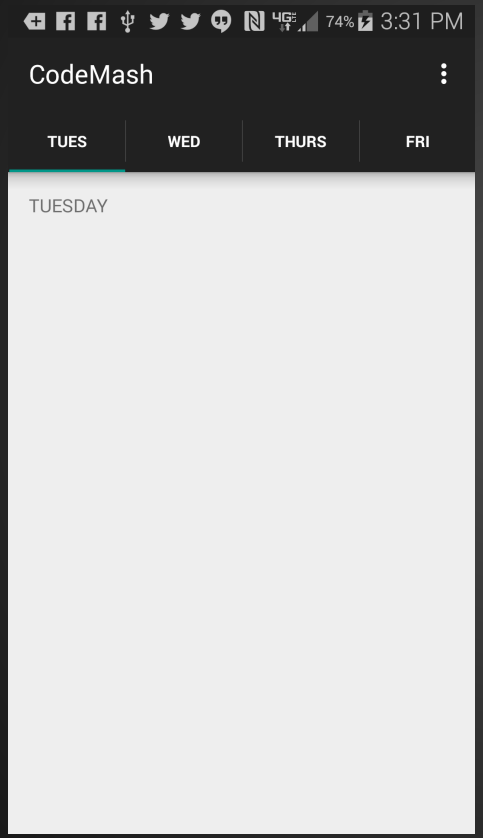
And let’s have our fragments tell us what day they’re on...

```
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.hasArguments()) {
            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...



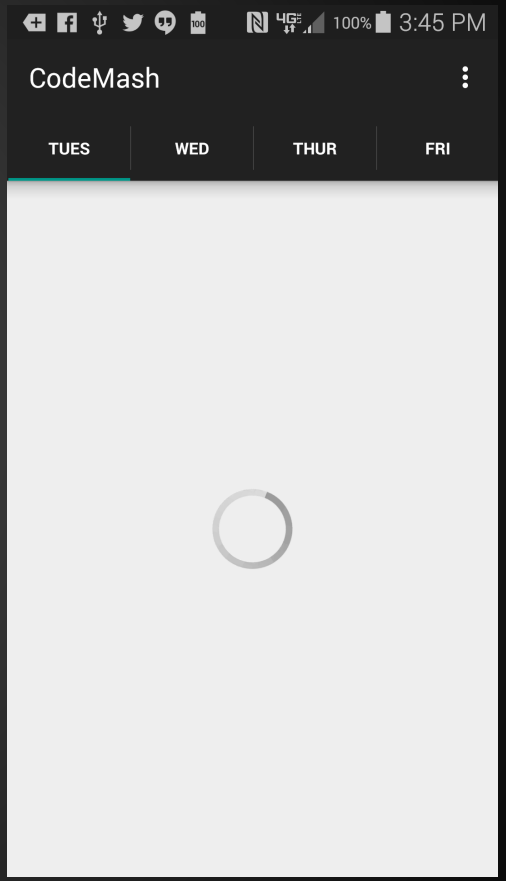
Now let’s get a list displayed on these fragments...

Introducing ListFragment!

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

SessionListFragment.java

```java
package com.brbw.codemash.controllers.fragments;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.ListFragment;

import com.brbw.codemash.models.Day;

public class SessionListFragment extends ListFragment {

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

    private Day day;

    public static Fragment newInstance(Day day) {
        SessionListFragment fragment = new SessionListFragment();
        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.hasArguments()) {
            day = (Day) getArguments().getSerializable(ARG_DAY);
        }
    }

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

}
```



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

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (this.hasArguments()) {
        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.hasArgumentsFor(ARG_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.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.lookupHostByName(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

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!

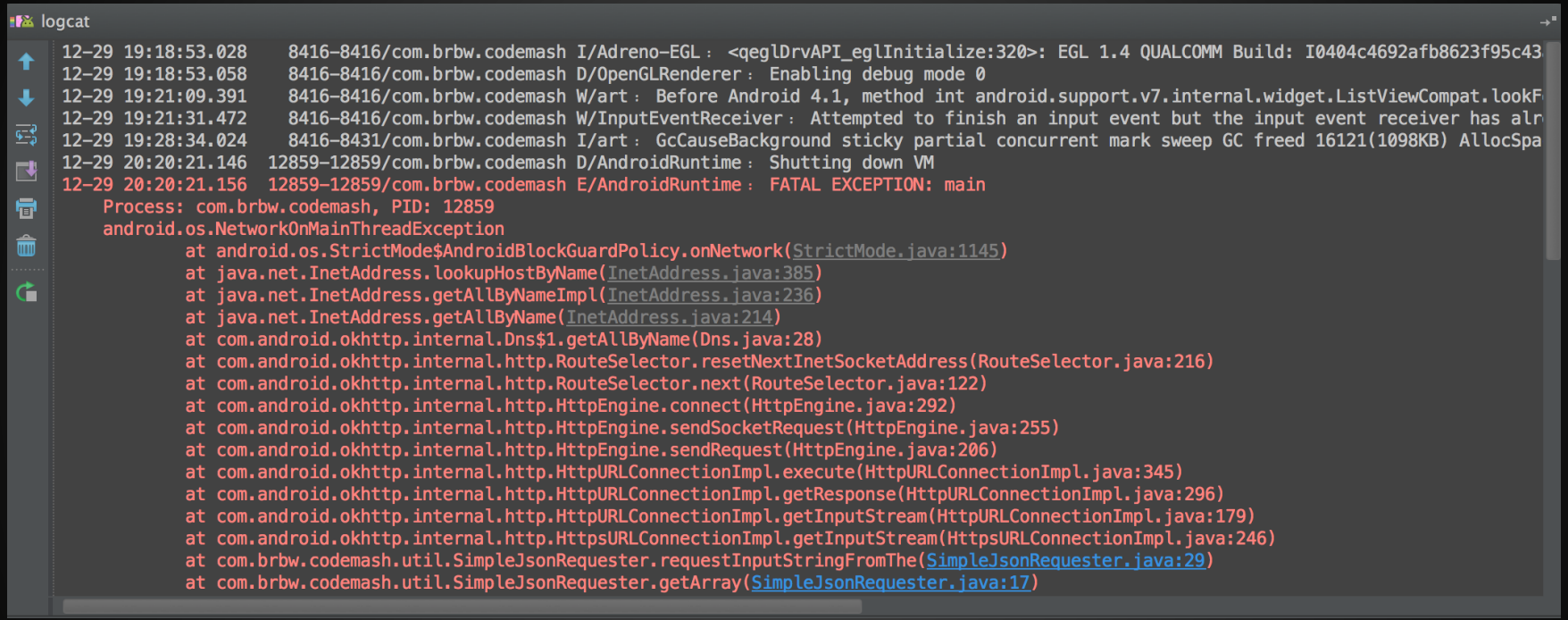
- Retrofit (By Square) http://square.github.io/retrofit/
- GSON https://code.google.com/p/google-gson/
- Guava https://code.google.com/p/guava-libraries/
- In some cases, volley: https://developer.android.com/training/volley/simple.html
- And then some…

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?



(NO)

So let’s talk about AsyncTasks

```java
@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

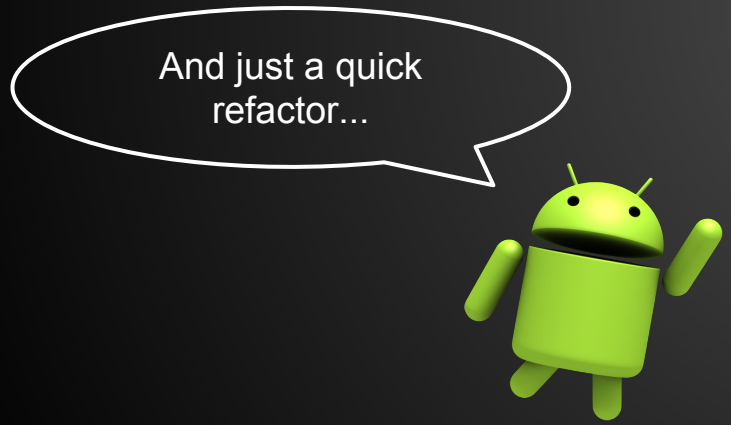
There are a few threading rules that must be followed for this class to work properly:

- 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.)

Lazily ripped from: http://developer.android.com/reference/android/os/AsyncTask.html

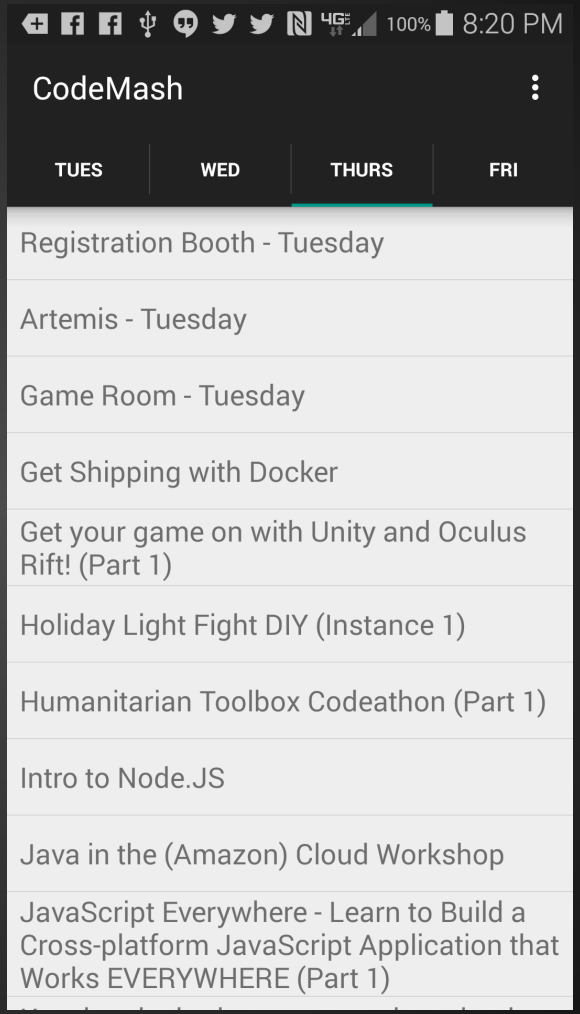
SessionListFragment.java | SessionRetrievalTask.java

```java
package com.brbw.codemash.tasks;

import android.os.AsyncTask;

import com.brbw.codemash.model.CodeMashService;
import com.brbw.codemash.model.Session;
import com.brbw.codemash.util.SimpleJsonRequester;

import java.util.List;

public class SessionRetrievalTask extends AsyncTask<Void,Void,List<Session>>{

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

```




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


    if (this.hasArgumentsFor(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);
}
```

But wait… we're showing ALL sessions on ALL days?!?!



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"
}
```

```java
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;
    }
```

CodeMashApplication.java

```java
package com.brbw.codemash;

import android.app.Application;

import com.brbw.codemash.models.SessionList;
import com.brbw.codemash.util.network.SimpleJsonRequester;

public class CodeMashApplication extends Application {

    private static SessionList sessionList;

    public synchronized static SessionList sessionListInstance() {
        if (sessionList == null) {
            sessionList = new SessionList(new CodeMashService(new SimpleJsonRequester()));
        }
        return sessionList;
    }
}
```

CodeMashApplication.java | AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<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="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".controllers.activities.MainActivity"
```

Now let's modify our AsyncTask and use it!

```java
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 | SessionRetrievalTask.java

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

        if (this.hasArgumentsFor(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(day);
        }
    }
```

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

adapter_session_list_item.xml

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

    <android.support.v7.widget.CardView
        android:id="@+id/card_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent">


    </android.support.v7.widget.CardView>

</LinearLayout>
```

The following classes could not be found:

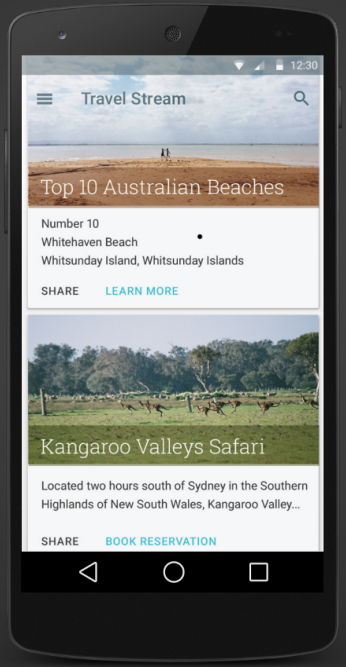
- android.support. (Fix Build Path,

Tip: Try to build the project.

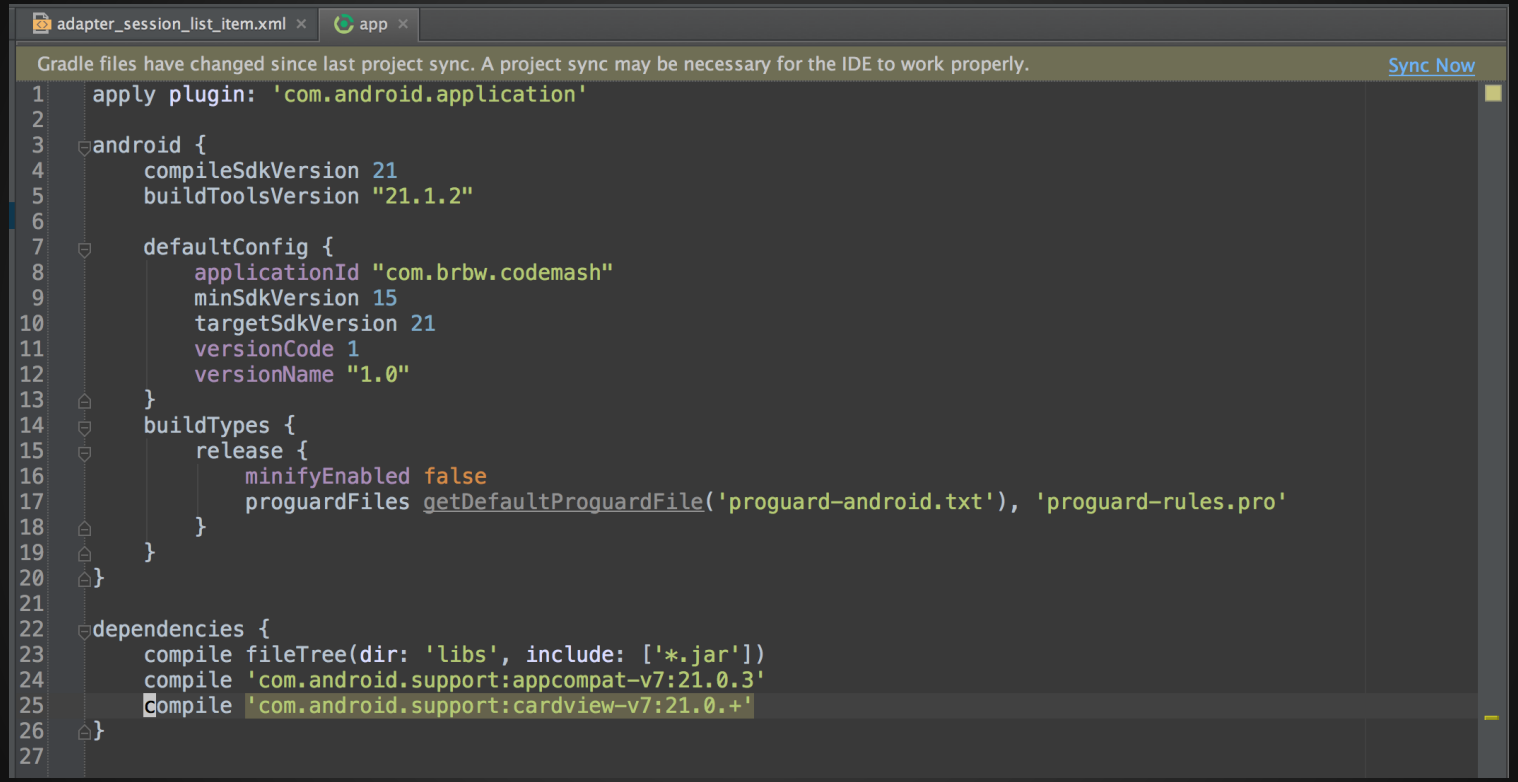
adapter_session_list_item.xml

```xml
<?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">


</android.support.v7.widget.CardView>

```

What’s a Cardview?



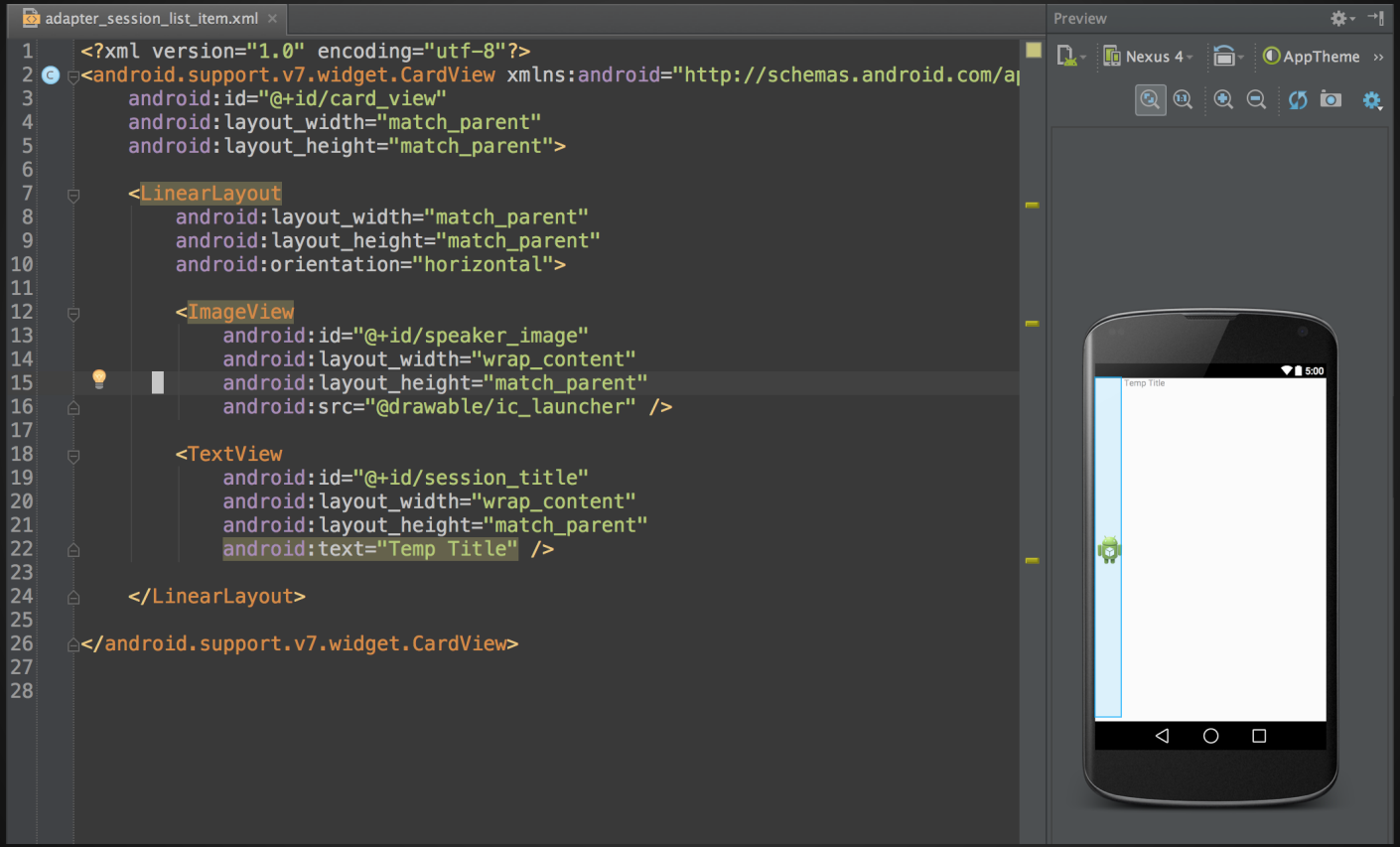
source: https://developer.android.com/training/material/lists-cards.html


adapter_session_list_item.xml
app
Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.
Sync Now
apply plugin: 'com.android.application'
android {
compileSdkVersion 21
buildToolsVersion "21.1.2"
defaultConfig {
applicationId "com.brbw.codemash"
minSdkVersion 15
targetSdkVersion 21
versionCode 1
versionName "1.0"
}
buildTypes {
release {
minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
}
}
}
dependencies {
compile fileTree(dir: 'libs', include: ['*.jar'])
compile 'com.android.support:appcompat-v7:21.0.3'
compile 'com.android.support:cardview-v7:21.0.+'
}

Back to our layout!

adapter_session_list_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView xmlns:android="http://schemas.android.com/a
    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>
```



Now let's populate it!

adapter_session_list_item.xml | SessionListAdapter.java

```
1  package com.brbw.codemash.controllers.adapters;
2
3  import ...
14
15 public class SessionListAdapter extends ArrayAdapter<Session> {
16
17     private static final int LAYOUT_ID = R.layout.adapter_session_list_item;
18
19     public SessionListAdapter(Context context, List<Session> sessions) {
20         super(context, LAYOUT_ID, sessions);
21     }
22
23     @Override
24     public View getView(int position, View convertView, ViewGroup parent) {
25
26         if(convertView == null) {
27             LayoutInflater inflater = (LayoutInflater) getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
28             convertView = inflater.inflate(LAYOUT_ID, parent, false);
29         }
30
31         Session session = getItem(position);
32
33         TextView title = (TextView) convertView.findViewById(R.id.session_title);
34         title.setText(session.getTitle());
35
36         return convertView;
37     }
38 }
39
```

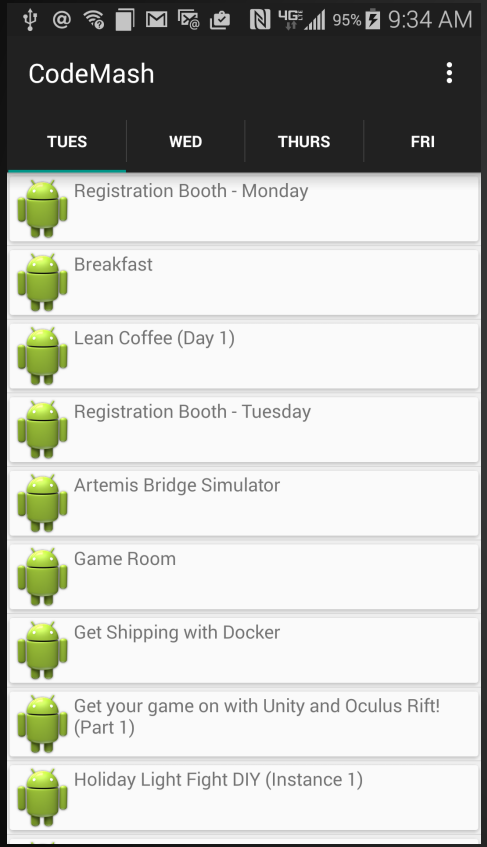
checkout: http://developer.android.com/reference/android/widget/Adapter.html

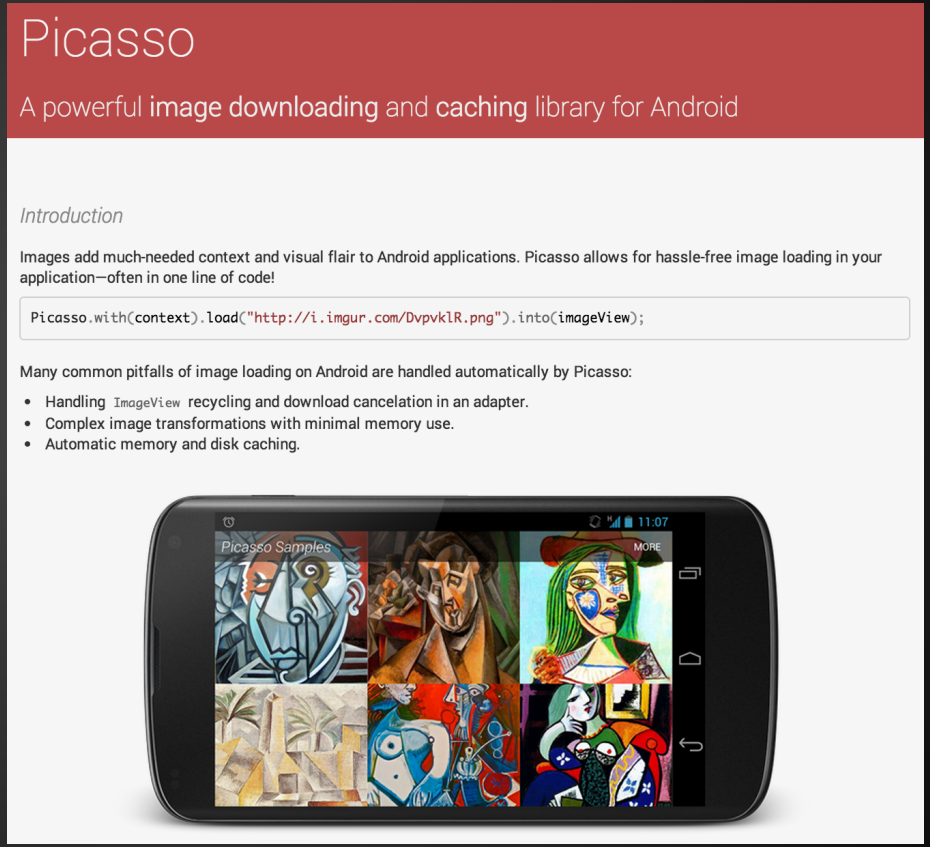
Now let's populate it!

adapter_session_list_item.xml | SessionListAdapter.java

```
1  package com.brbw.codemash.controllers.adapters;
2
3  import ...
14
15 public class SessionListAdapter extends ArrayAdapter<Session> {
16
17     private static final int LAYOUT_ID = R.layout.adapter_session_list_item;
18
19     public SessionListAdapter(Context context, List<Session> sessions) {
20         super(context, LAYOUT_ID, sessions);
21     }
22
23     @Override
24     public View getView(int position, View convertView, ViewGroup parent) {
25
26         if(convertView == null) {
27             LayoutInflater inflater = (LayoutInflater) getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
28             convertView = inflater.inflate(LAYOUT_ID, parent, false);
29         }
30
31         Session session = getItem(position);
32
33         TextView title = (TextView) convertView.findViewById(R.id.session_title);
34         title.setText(session.getTitle());
35
36         return convertView;
37     }
38 }
39
```

But what about those images?...





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

But what about those images?...

```
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'
}
```

Picasso

A powerful **image downloading** and **caching** library for Android

loading in your

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

But what about those images?...

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

    if (convertView == null) {
        LayoutInflater inflater = (LayoutInflater) 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;
}
```

```xml
<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 :)

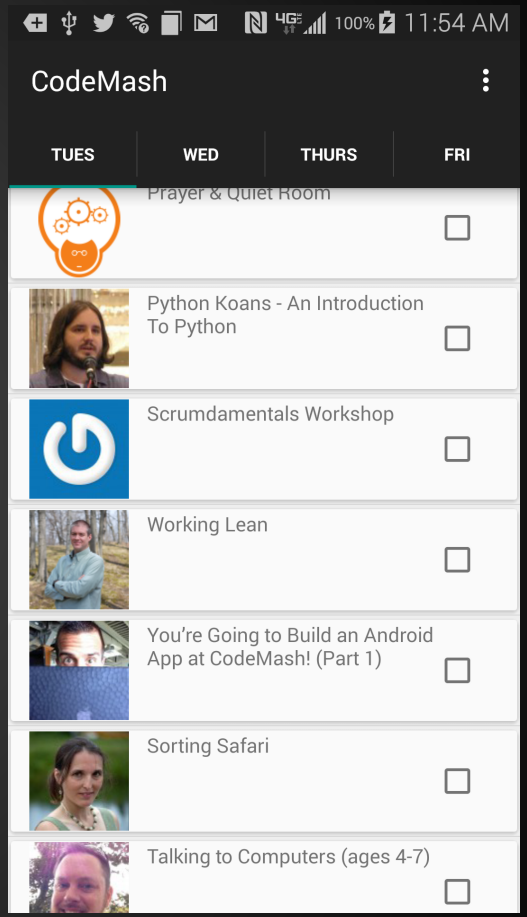
```
public class SessionListAdapter extends ArrayAdapter<Session> {

    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

```
<?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"
    android:descendantFocusability="blocksDescendants">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal"
        android:weightSum="1">

        <ImageView
            android:id="@+id/speaker_image"
            android:layout_width="70dp"
            android:layout_height="70dp"
            android:layout_weight=".10"
            android:src="@drawable/ic_launcher" />

        <TextView
            android:id="@+id/session_title"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight=".85"
            android:text="Temp Title" />

        <CheckBox
            android:id="@+id/session_favorite"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_weight=".05" />

    </LinearLayout>

</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

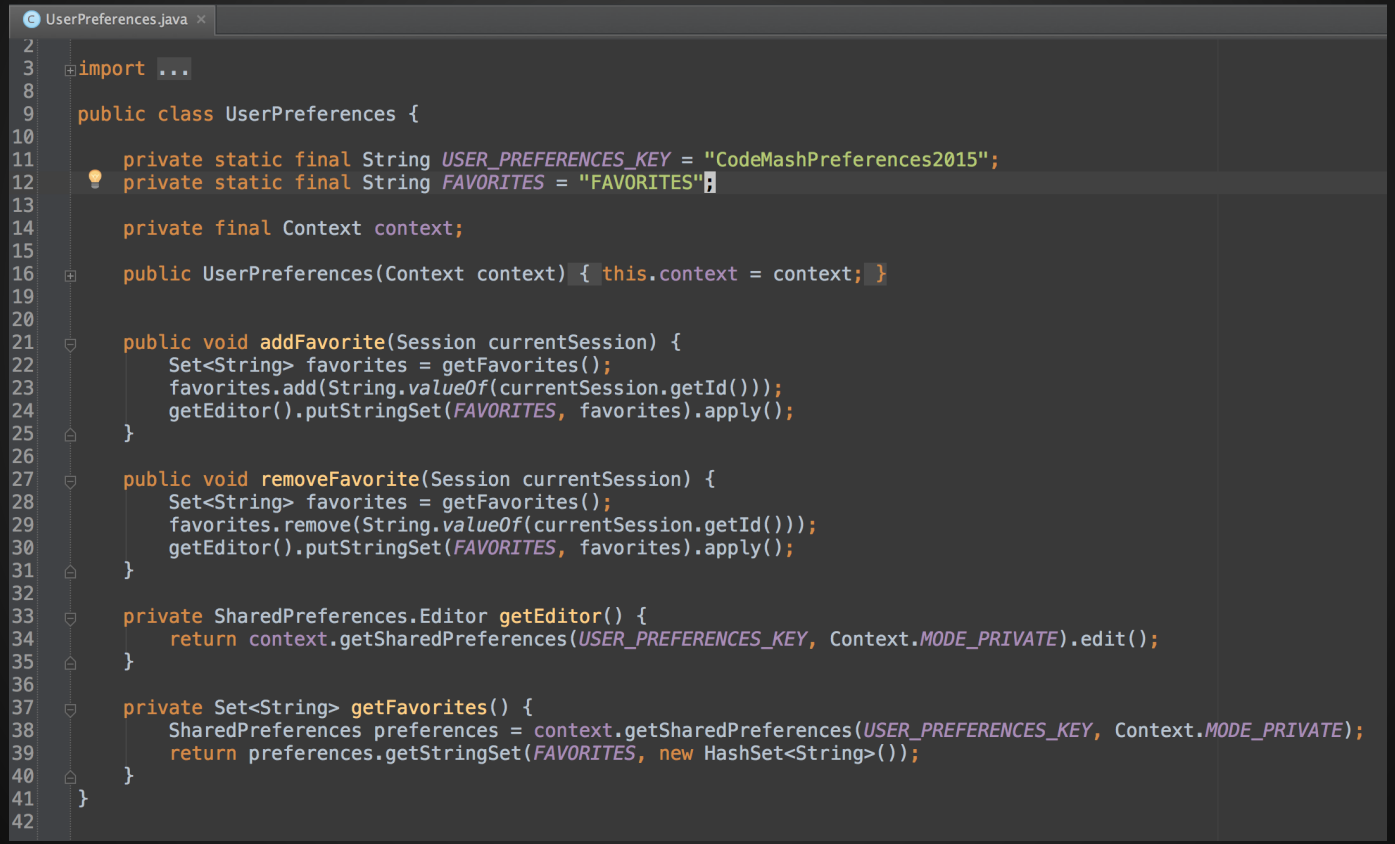
IN THIS DOCUMENT

SEE ALSO

Content Providers and Content Resolvers

source: http://developer.android.com/guide/topics/data/data-storage.html

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



Let’s access this through the application as well

UserPreferences.java | CodeMashApplication.java

```
1  package com.brbw.codemash;
2
3  import ...
8
9  public class CodeMashApplication extends Application {
10
11     private static SessionList sessionList;
12     private static UserPreferences userPreferences;
13
14     @Override
15     public void onCreate() {
16         super.onCreate();
17         userPreferences = new UserPreferences(getBaseContext());
18     }
19
20     public synchronized static SessionList sessionListInstance() {
21         if (sessionList == null) {
22             sessionList = new SessionList(new CodeMashService(new SimpleJsonRequester()));
23         }
24         return sessionList;
25     }
26
27     public synchronized static UserPreferences userPreferencesInstance() {
28         return userPreferences;
29     }
30 }
31
```

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?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) {
            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

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity">

    <item
        android:id="@+id/menu_filter"
        android:icon="@android:drawable/ic_menu_sort_by_size"
        android:title="@string/filter"
        app:showAsAction="always" />

</menu>
```

Let’s start by modifying our menu

menu_main.xml | MainActivity.java

```
                .setTabListener(this));
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.menu_filter) {
            Toast.makeText(this,"That tickled",Toast.LENGTH_LONG).show();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
```

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(); }

    @NonNull
    @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(); }

    @NonNull
    @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?

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

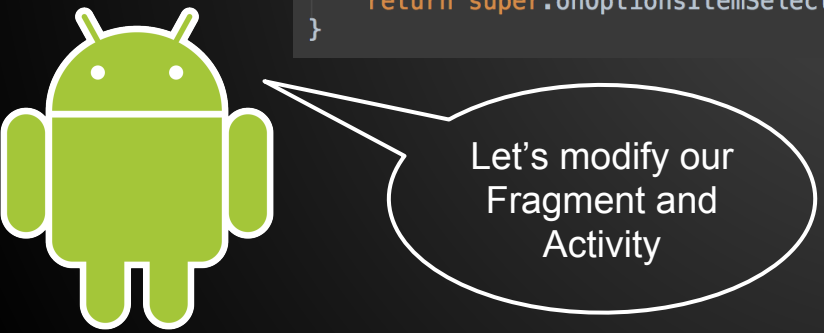
So, back to updating those views (and then some!...)

```
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;
    }

    @NonNull
    @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

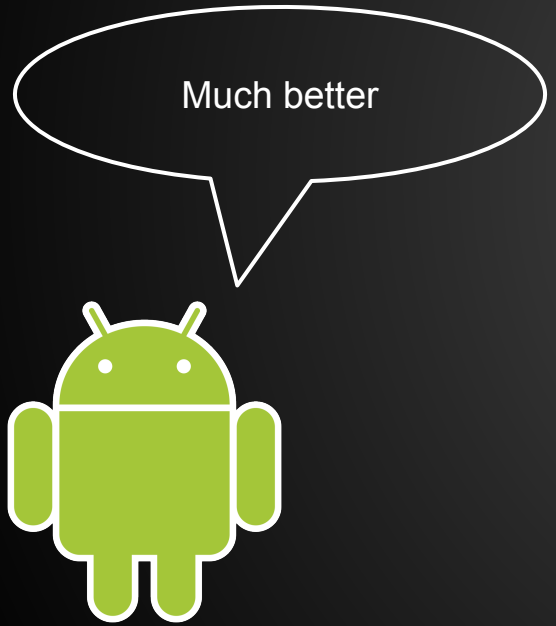
So, back to updating those views (and then some!...)

```
@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);
}
```



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

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




```
@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

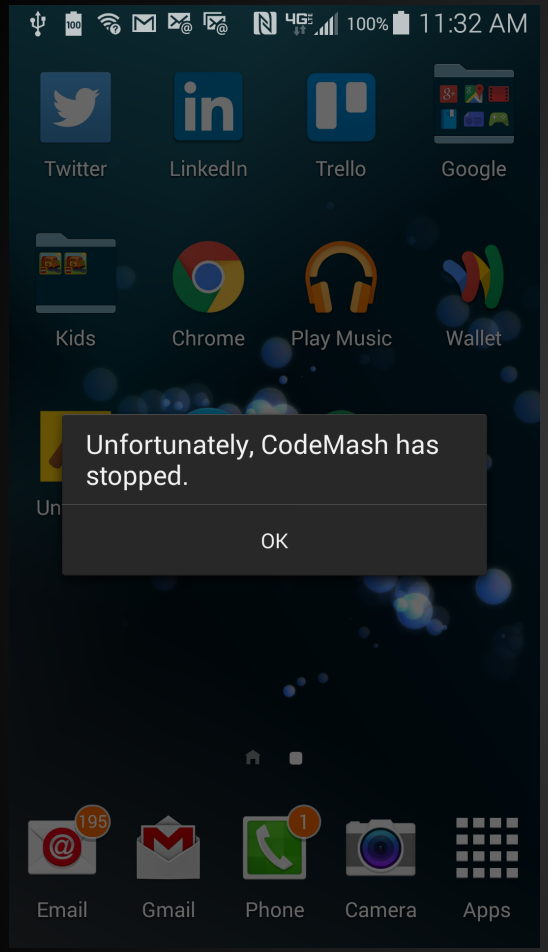
Let’s talk about intents a bit more…
To the docs!
http://developer.android.com/reference/android/content/Intent.html

Intents

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

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


```java
public class SessionDetailsActivity extends ActionBarActivity {

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




```
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.
            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

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

    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

fragment_session_details.xml

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

    <TextView
        android:id="@+id/session_title"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Let’s display it!

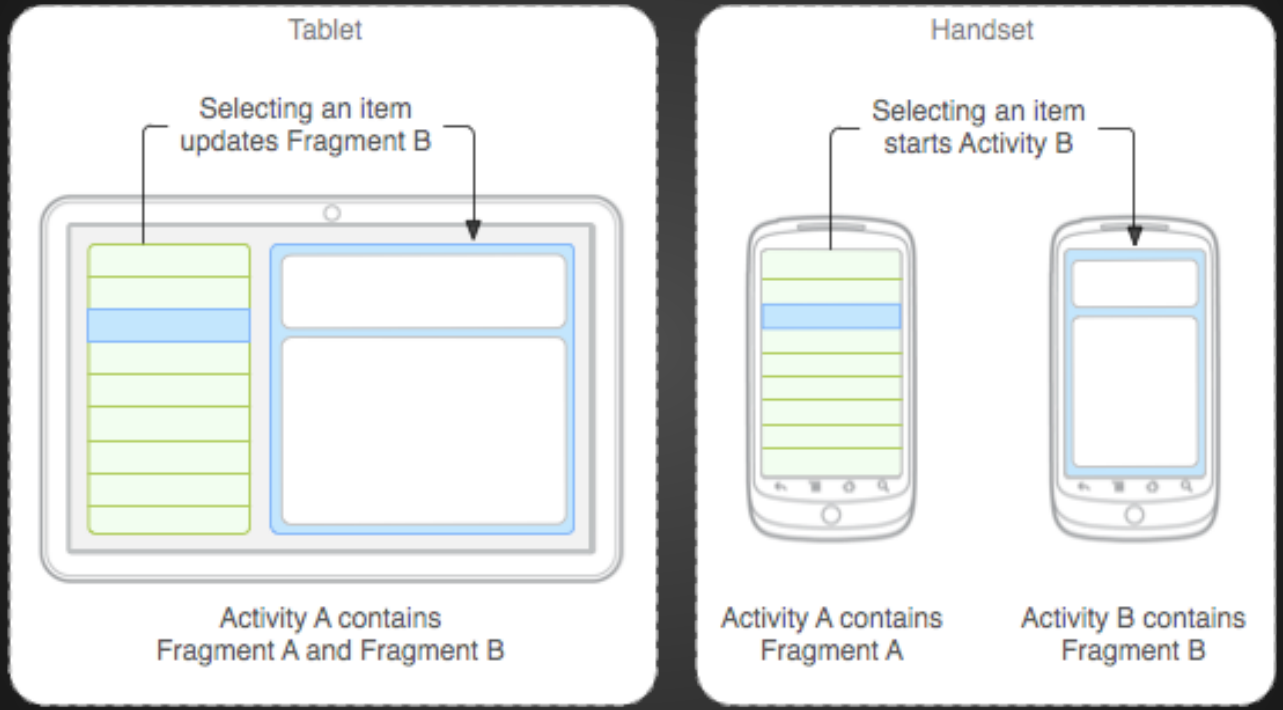
SessionDetailsFragment.java

```
7  import android.view.View;
8  import android.view.ViewGroup;
9
10 import com.brbw.codemash.R;
11 import com.brbw.codemash.models.Session;
12 import com.brbw.codemash.util.ViewHelper;
13
14 public class SessionDetailsFragment extends Fragment {
15
16     private static final String SESSION_ARGS = "SESSION_ARGS";
17
18     public static SessionDetailsFragment newInstance(Session session) {
19         SessionDetailsFragment fragment = new SessionDetailsFragment();
20         Bundle args = new Bundle();
21         args.putSerializable(SESSION_ARGS, session);
22         fragment.setArguments(args);
23         return fragment;
24     }
25
26     @Override
27     public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
28         View view = inflater.inflate(R.layout.fragment_session_details, container, false);
29
30         if (this.hasArgsFor(SESSION_ARGS)) {
31             Session session = (Session) getArguments().getSerializable(SESSION_ARGS);
32             new ViewHelper(view)
33                     .setText(R.id.session_title, session.getTitle());
34         }
35         return view;
36     }
37
38     private boolean hasArgsFor(String args) {
39         return getArguments() != null && getArguments().containsKey(args);
40     }
41 }
42
```

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?



Lazily ripped from: http://developer.android.com/guide/components/fragments.html

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

SessionDetailsActivity.java

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_single_fragment);
        getSupportActionBar().setDisplayShowTitleEnabled(true);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

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

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.session_details, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (R.id.session_share == item.getItemId()) {
            startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(buildTweet())));
            return true;
        }else if(android.R.id.home == item.getItemId()) {
            onBackPressed();
        }
        return super.onOptionsItemSelected(item);
    }

    private String buildTweet() {
        String message = String.format(getString(R.string.twitter_message), session.getTitle());
        String twitterUrl = getString(R.string.twitter_url);
        return String.format(twitterUrl, urlEncode(message));
    }
}
```

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

session_details.xml

```
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

Edit translations for all locales in the translations editor.

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

    <string name="app_name">CodeMash</string>
    <string name="action_settings">Settings</string>
    <string name="tuesday">tues</string>
    <string name="wednesday">wed</string>
    <string name="thursday">thurs</string>
    <string name="friday">fri</string>
    <string name="empty_session_list_message">No sessions available!</string>
    <string name="filter">filter</string>
    <string name="filter_title">Filter Sessions</string>

    <array name="favorites">
        <item>Show All</item>
        <item>Show Favorites Only</item>
    </array>

    <string name="twitter_message">Checking out %s at #CodeMash!</string>
    <string name="twitter_url">https://twitter.com/intent/tweet?text=%s</string>

</resources>
```

And that's it

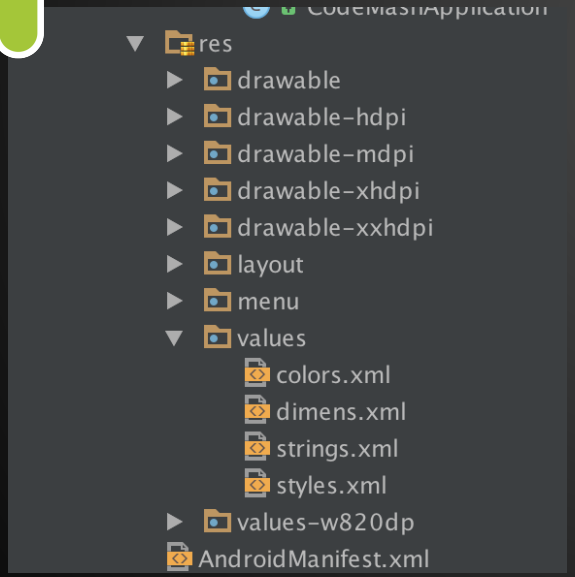
Let’s recap before moving forward

- 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

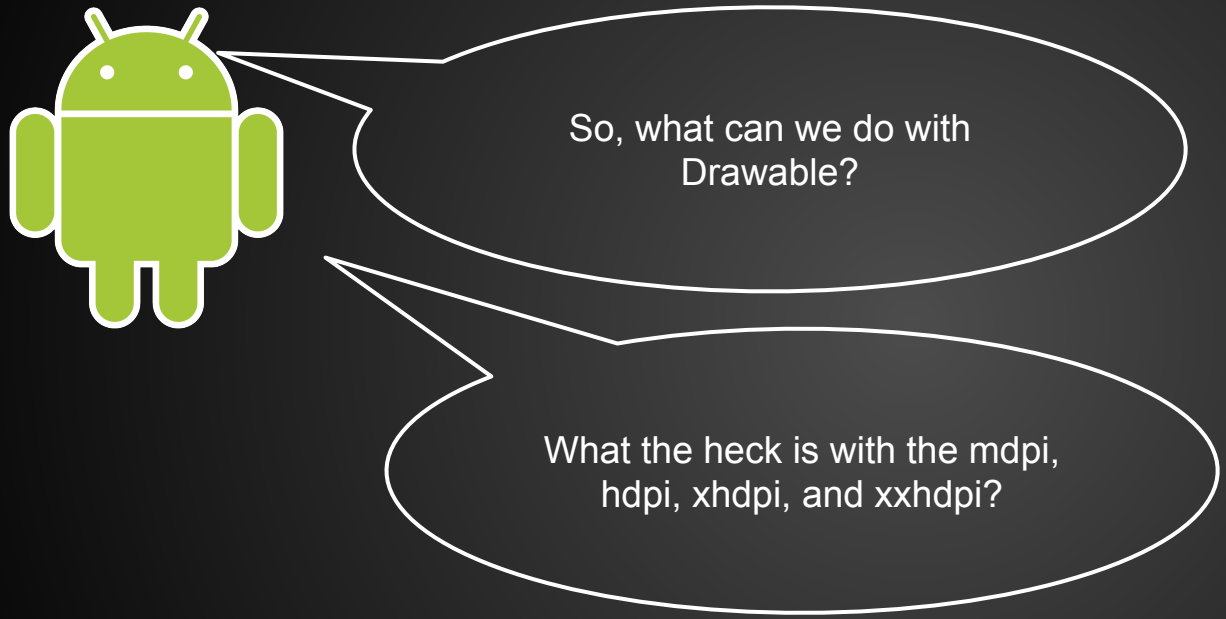
Resources

So, what other things are in the res directory?



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

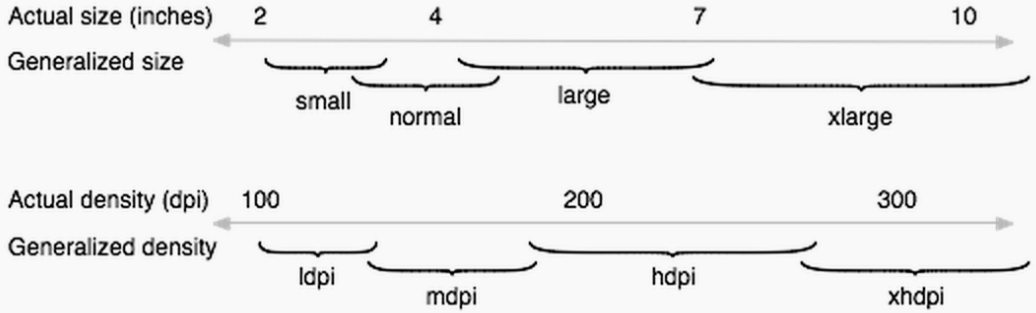
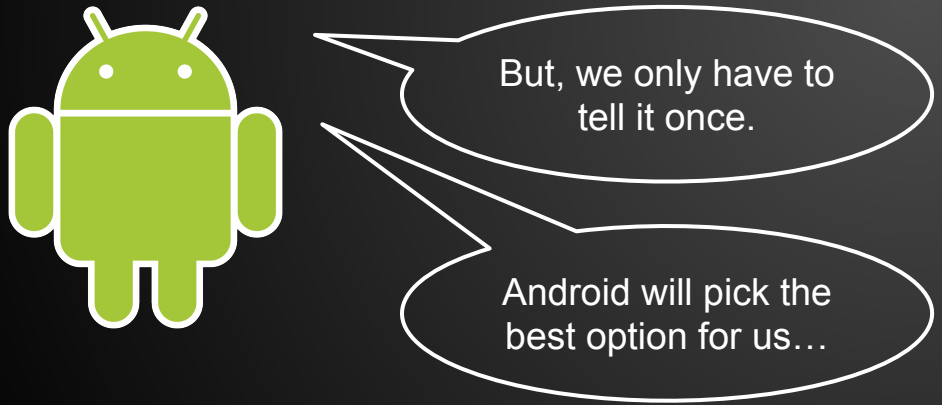


Figure 1. Illustration of how Android roughly maps actual sizes and densities to generalized sizes and densities (figures are not exact).

Lazily ripped from: http://developer.android.com/guide/practices/screens_support.html

Drawable

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



Shapes

So, how do we make this guy?

Shapes

So, how do we make this guy?

```xml
<?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>
```

```xml
<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
<?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>
```

```xml
<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.