

Android Troubleshooting/Debugging

Updated: 10th September, 2018

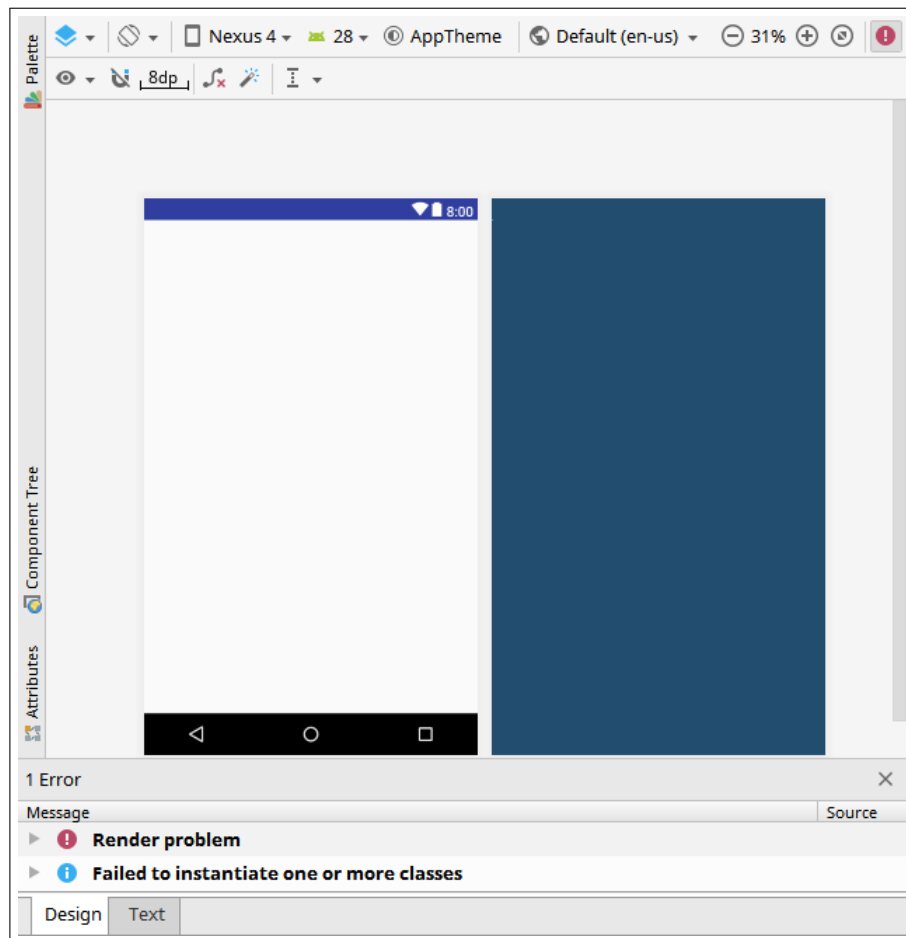
Contents

1	The UI Drag-and-Drop Editor	1
1.1	Broken AppCompatActivity Theme	1
1.2	Missing Device Information	3
1.3	No UI Constraints – Everything in the Top-Left Corner	3
2	Opening Projects	4
3	The Emulator	4
3.1	KVM/Acceleration Error Messages	4
3.2	Waiting for Target Device to Come Online	5
4	Editor/Compile Errors	5
4.1	Cannot Resolve Symbol 'R' / Package 'R' does not exist	5
4.2	Cannot Resolve Symbol 'RecyclerView' / 'LinearLayoutManager'	6
4.3	Missing Build Dependencies	6
5	Debugging	7
5.1	Crash on Startup	7
5.2	Debugging with Logcat	7
5.3	Debugging with Breakpoints	9
5.4	android.view.InflateException	9
5.5	android.content.ActivityNotFoundException	9
5.6	java.lang.ClassNotFoundException	10
5.7	android.content.res.Resources\$NotFoundException: String resource ID ...	10
5.8	java.lang.NullPointerException	11

1 The UI Drag-and-Drop Editor

1.1 Broken AppCompatActivity Theme

When trying to design your UI layout, you may get an empty screen:



(The red button in the top-right shows that the IDE knows something is wrong.)

The actual problem appears to be a bug in the AppCompatActivity library. You will need to select one of several workarounds:

Option A. Change the theme from “AppTheme” to something else (“Light”, etc.).

Option B. Change the AppCompatActivity definition. In “app/src/main/res/values/styles.xml”, find this line:

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
```

And prefix the value of parent with “Base.”:

```
<style name="AppTheme" parent="Base.Theme.AppCompat.Light.DarkActionBar">
```

Option C. Change the version of the AppCompatActivity library being used. At the time of writing, the most recent version (28.0.0 RC 2) still has the bug, but there is an older version (28.0.0 Alpha 1) that does not¹.

Find the following line in “app/build.gradle”:

```
implementation 'com.android.support:appcompat-v7:28.0.0-rc02'
```

And then change it to:

¹Available versions are listed at developer.android.com/topic/libraries/support-library/revisions.

```
implementation 'com.android.support:appcompat-v7:28.0.0-alpha1'
```

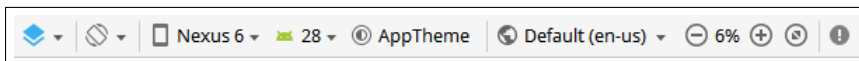
Then perform a “project sync” as advised:

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

1.2 Missing Device Information

In this case, nothing at all is displayed in the drag-and-drop editor (other than the tools and buttons around the outside), and two of the drop-down menus will be empty or nearly empty.

Normally, you should see this:



Notice the “Nexus 6” and “28” – these are the two drop-down menus that *should* let you select a device type and API (so you can preview your UI on different kinds of devices while you’re designing it; this is different from the emulator devices).

I have only seen this happen once, and I haven’t fully discovered why it happened. However, the problem appeared to be specific to the *current project*. Therefore, one workaround is to create a new project and copy across your code and XML from the old one.

1.3 No UI Constraints – Everything in the Top-Left Corner

You won’t notice this until you run your app. If you’re working with ConstraintLayout, but you haven’t specified any constraints, all your UI elements will bunch up in the top-left corner.

This is true even if you’ve dragged them all to the right place in the editor. Unfortunately, that manual placement is actually meaningless! In order to *actually* position your UI elements, you have to add constraints.

If you click on one of your UI elements, it probably looks like this:



The four circle-connectors can be dragged to the screen edges, or to corresponding circles on other UI elements. Doing this will add a constraint, which causes the UI element to be held in place, relative to whatever you dragged it to. Some more specifics:

- Constraints can be vertical or horizontal. Each UI element should have at least one (possibly two) of each.

- You can centre an element within a gap (or across the whole screen) by adding two opposing constraints; e.g. drag the left connector to the left screen edge, and the right connector to the right screen edge. You can also “bias” the centring, so that the element appears at (say) 25% of the way from one point to the other.
- Constraints have margins. By default, there’s a small gap between the element and what it’s constrained to. You can adjust the size of the margins, though typically stick to 8dp or 16dp. (If you want to have dynamically-sized margins, then you probably need to re-think your constraints.)

The squares on the corners will adjust the element’s size, but in most cases you don’t want to do this manually. It’s usually better to set the element’s “layout_width” and “layout_height” attributes to one of the following:

- “wrap_content” – automatically sets the element’s size to the minimum needed to fit everything it contains.
- “0dp” or “match_constraint” (equivalent) – automatically sets the element’s size to fill the available space.

(You can make the width and height behave differently, with one set to wrap_content and the other to match_constraint, if desired.)

2 Opening Projects

On the Bentley B314 Linux lab machines, the “Open File or Project” window may temporarily freeze. Generally, this is because it’s trying to show you a directory tree, and unfortunately this involves reading the contents of /home, which can be a very slow network operation in our particular setup.

To avoid waiting, you might like to use the command-line to open projects; e.g.:

```
[user@pc]$ /local/android/studio ~/AndroidStudioProjects/MyApplication
```

(Where “~/AndroidStudioProjects/MyApplication” should be the directory containing your app. You will need to adjust it as appropriate.)

3 The Emulator

3.1 KVM/Acceleration Error Messages

On the lab machines, if you’re getting messages related to KVM, this is a system administration issue. Try a different machine, and please let the lecturer / sysadmin know which machine(s) exhibited the problem.

If you’re using your own machine, you may need to follow these instructions:

Windows: developer.android.com/studio/run/emulator-acceleration#vm-windows.

MacOS: developer.android.com/studio/run/emulator-acceleration#vm-mac.

Linux: developer.android.com/studio/run/emulator-acceleration#vm-linux.

3.2 Waiting for Target Device to Come Online

It's normal to see this message while the virtual device is booting up. If the IDE finishes building your app before the device is ready to receive it, the IDE has to wait.

However, sometimes the IDE continues "waiting" forever, even when the device is ready. The root cause of this problem is not entirely clear, but Mitch Bartlett describes several steps you can try in this article:

Mitch Bartlett (2018), "Android Emulator Stuck at 'Waiting for target to come online'", <https://www.technipages.com/android-emulator-stuck-waiting-for-target-to-come-online>.

The recommended approaches are:

Option A. Stop and restart the emulator (via "Tools" → "AVD Manager");

Option B. Exit Docker (if using MacOS);

Option C. Wipe the virtual device (via the AVD Manager);

Option D. Uninstall and reinstall the emulator (via "Tools" → "SDK Manager").

(I have observed that creating a new virtual device often solves the problem too, though it's likely that wiping the existing device is an easier way to achieve the same effect.)

4 Editor/Compile Errors

4.1 Cannot Resolve Symbol 'R' / Package 'R' does not exist

The editor highlights references to "R" in red, and the app fails to compile. There are several known causes of this:

- (a) You have an error in one of your resource XML files, probably a misspelt attribute name or value. For instance:

```
<TextView
    android:layout_widthzzz="wrap_content"
    ...
```

In this case, in the build display area you should also see various error messages, including:

- "error: attribute '...' not found";
- "error: '...' is incompatible with attribute ...";
- "failed linking file resources";
- "AAPT2 error: check logs for details".

After getting these errors, the editor and compiler will no longer be able to find the "R" class, because it no longer exists. It's auto-generated, and the logic for auto-generating it has been broken.

If you look in the relevant XML file, the editor should shade/highlight any misspelt attributes. Fix these, and then rebuild the project.

- (b) You are missing a build dependency on:
“com.android.support.constraint:constraint-layout”. See [Missing Build Dependencies](#) below.

(Normally the IDE will set things up initially so that you can use ConstraintLayout without any dramas, but there are situations in which you need to add it manually.)

- (c) You have moved your Java files to a different package and not also updated the manifest file: app/src/main/AndroidManifest.xml. The manifest specifies which package the R class is to be created in, as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.curtin.myapp">
    ...
```

If “R” ends up in the wrong package, then technically it exists but the compiler won’t be able to find it.

Update the manifest file, and rebuild the project.

4.2 Cannot Resolve Symbol ‘RecyclerView’ / ‘LinearLayoutManager’

You are missing a build dependency on “com.android.support:recyclerview”. See [Missing Build Dependencies](#) below.

(When you use the drag-and-drop editor to add a RecyclerView, it generally asks you if you’d like to add the corresponding build dependency. But if you type the XML in manually or acquire it from elsewhere, you may also need to add the dependency manually.)

4.3 Missing Build Dependencies

Android apps frequently depend on libraries that must be added to the project. Two commonly used in MAD are:

- “com.android.support.constraint:constraint-layout”; and
- “com.android.support:recyclerview”.

If you need to add build dependencies, you can do so in two ways:

- (a) Select “File” → “Project Structure”, then select the “app” module (or whatever module name is shown), and then the “Dependencies” tab. You should see the lists of dependencies. If the required dependencies are not there, you can select “+”, and then “Library dependency”, and then choose from among the options shown.
- (b) Directly edit the build script app/build.gradle (noting that there are at least *two* files called build.gradle, and you want the one in the app/ directory).

There should be a “dependencies { ... }” block as follows:

```
...  
dependencies {  
    ...  
    implementation 'com.android.support.constraint:constraint-layout:1.1.2'  
    implementation 'com.android.support:recyclerview-v7:28.0.0-rc02'  
}
```

(The only trouble here is that you need to know which versions of the libraries to depend on. Usually the latest ones, whatever they are, is ideal.)

5 Debugging

5.1 Crash on Startup



If your app compiles but doesn't start (and the emulator is running, and the IDE isn't still “waiting”), then it has probably crashed on startup before displaying anything. To help confirm that it is actually a problem with your code (and not the IDE, emulator, etc.), try re-running the app from the device itself. You can do this in the same way you would launch (or re-launch) any app:

- Access the apps menu (by dragging the bottom edge of the screen upwards).
- Find your app and click on it.

If crashing is the issue, Android will eventually show you a “My Application keeps stopping” dialog box.

If you get this, you know that it's a *crash*, and you need to start debugging.

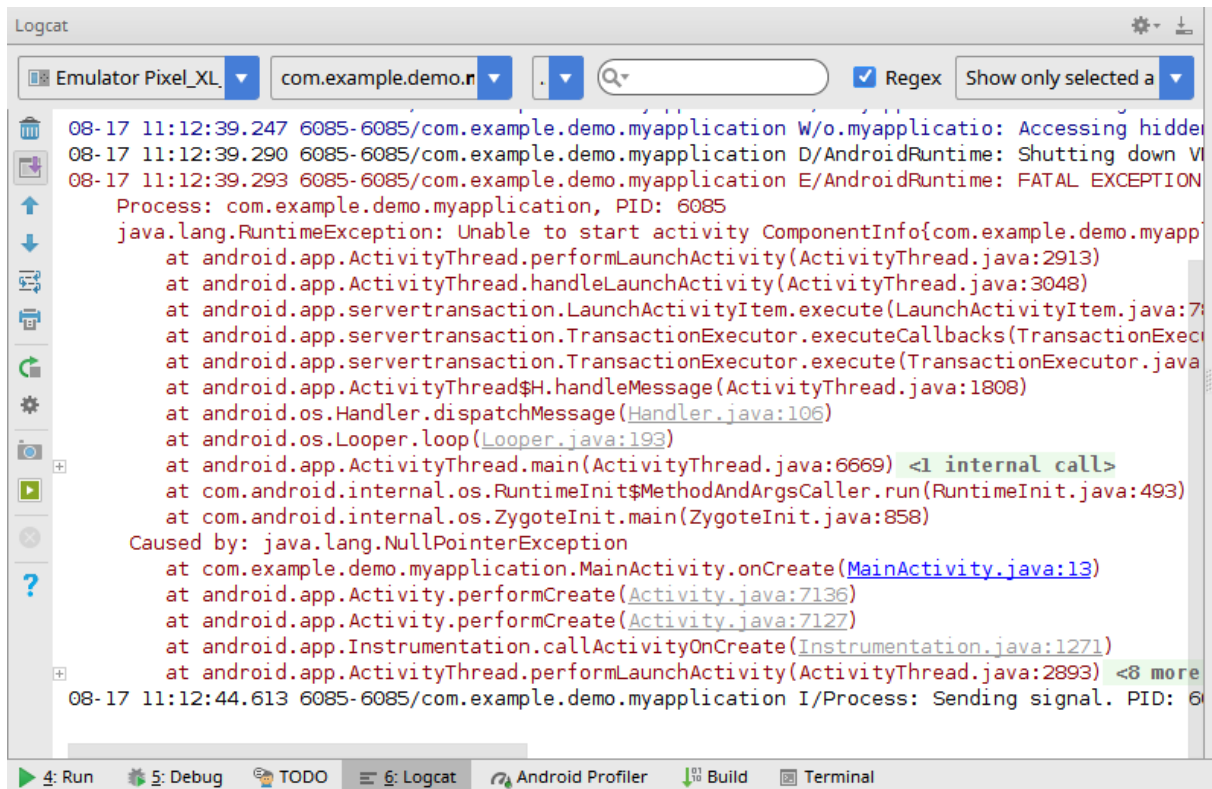
My Application keeps stopping

-  App info
-  Close app

5.2 Debugging with Logcat

Find the logcat² display by clicking on the “Logcat” tab at the bottom.

²Logcat is a command-line tool run by Android Studio in the background.



If your application crashes, you should see the relevant exception messages and stack trace here. In this example, we can see:

- There was a RuntimeException generated somewhere within the standard Android classes.
- But this was *caused* by a NullPointerException in MainActivity.onCreate(), on line 13 of MainActivity.java. That's where to start looking for the problem.

You can add your own messages to the logcat display (to reproduce the effect of running System.out.println() in a “normal” Java application):

```
public MyActivity extends AppCompatActivity
{
    private static final String TAG = "MyActivity";
    ...

    // Inside some method:
    Log.d(TAG, "Setting displayed text...");
}
```

The Log class has various one-letter static methods for logging messages. Log.d() is for debugging messages. These can be inserted into your code temporarily, but it's actually good practice to have *permanent* logging statements. You can also use Log.e() for errors, Log.w() for warnings, Log.i() for information, etc.

The first “tag” parameter is to help distinguish log messages originating from different places. Some other activity (or indeed any other class) would define its own separate tag value.

You can also supply an exception object as an optional third parameter:


```
try
{
    ...
}
catch(SomeHorribleException e)
{
    Log.e(TAG, "A horrible thing happened.", e);
}
```

You can find the official guide on Android logging here:
<https://developer.android.com/studio/debug/am-logcat>.

5.3 Debugging with Breakpoints

Use the IDE to its full potential:

- (a) Set a breakpoint on any given line of code by either (1) clicking next to a line number, (2) pressing Ctrl+8, or (3) choosing “Run” → “Toggle Line Breakpoint”. A red dot should appear next to the line.

- (b) Choose the “Debug” option (the  icon) to run your app, or press Shift+F9. Note: the normal “Run” option ignores breakpoints.

- (c) Once your app hits the breakpoint, it will pause, and the IDE will show you the method call stack, and the actual current values of all fields and variables.

Use this to figure out what’s going wrong. You’ll hopefully find it quicker and easier than inserting Log.d() everywhere.

- (d) While debugging, you can select various arrow icons at the top of the debug display to step through your code incrementally, or resume normal execution. You can also add/remove breakpoints during debugging.

5.4 android.view.InflateException

If the logcat screen shows a stacktrace with “android.view.InflateException” at the top, the problem will (most likely) be in your layout XML files. InflateException happens when the UI could not be “inflated” (i.e. read from XML and instantiated into a tree of View objects). Basically, you’ve done something wrong in one of these files. Perhaps you’ve mis-named a tag, or used incorrect ID syntax?

5.5 android.content.ActivityNotFoundException

You may encounter this exception if you haven’t declared all your activities in your AndroidManifest.xml file. The IDE *normally* keeps this file up-to-date, and the build process detects *some* types of errors at compile-time. However, neither of these are guarantees, and so you need to know how to deal with the file yourself.

Basically, AndroidManifest.xml should look something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication">

    <application ...>
        <activity android:name=".MyActivity1">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MyActivity2">
        </activity>
        <activity android:name=".MyActivity3">
        </activity>
    </application>
</manifest>
```

Notice that there is an `<activity ...></activity>` element for each Activity, and your package attribute at the top must also be correct.

5.6 java.lang.ClassNotFoundException

If the error message complains about not being able to find an activity class, it may be that the package attribute in AndroidManifest.xml is wrong. The IDE should probably be smart enough to show a warning message about this.

5.7 android.content.res.Resources\$NotFoundException: String resource ID ...

This may happen if you attempt to provide an integer to the `setText()` method (in `TextView` or `EditText`). In short, you must convert your `int` to a `String` first:

```
TextView textView = ...;
int value = ...;

// Do this:
textView.setText(String.valueOf(value));

// NOT this:
textView.setText(value);
```

Why? Remember that Java has method overloading. If you call `setText()` with an integer, that's a completely different method from the one that takes a `String`. The integer

version of `setText()` assumes its parameter is a resource reference, not simply a value to be displayed.

5.8 java.lang.NullPointerException

`NullPointerException` is a common flow-on effect from various other problems. You need more information before being able to guess why it's happening. Start by identifying the line number on which it occurs, then figure out what thing is null (a field, variable, method return result, etc.), and then backtrack to determine why.

That being said, in Android programming, `NullPointerException`s often happen in connection with `findViewById()` (and other methods that work in a similar fashion, like `findFragmentById()`). This returns null if it can't find the specified ID, which may occur for a couple of different reasons.

(a) You're looking for the wrong thing.

Consider this:

```
private EditText textField;  
...  
textField = (EditText) findViewById(R.id.myTextField);  
String s = textField.getText().toString();
```

The IDE will warn you if “myTextField” is not a valid ID, but it can't check whether that ID is actually part of the current UI layout or not, because that information only exists at runtime. It could be a valid ID for a *different* layout not currently being used (intended for a different activity/fragment/list/etc.).

In this case, `findViewById()` will return null, so `textField` becomes null, and then the call to `getText()` on the next line will trigger a `NullPointerException`.

(b) You're looking in the wrong place.

Consider this:

```
private Button btn;  
...  
btn = (Button) findViewById(R.id.myButton);  
btn.setOnClickListener(new View.OnClickListener()  
{  
    @Override  
    public void onClick(View view)  
    {  
        EditText textField = (EditText) view.findViewById(R.id.myTextField);  
        ...  
    }  
});
```

When the button is pressed, we want to do something with an `EditText` element. However, the above code will almost certainly trigger a `NullPointerException`, because of what we're doing with the `View` parameter.

An `onClick()` method's `View` parameter refers to the UI element that triggered the event, which in this case is a button. So, in this case, `view == btn` (they refer to the same object). Then, by writing `view.findViewById(...)`, we're actually looking for the text field *within* the button element itself, and of course it isn't there.

How to fix it? We want to be careful about where we're searching (not just what we're looking for).

We can (in this case) just remove the "view." from in front of `findViewById(...)`. However, it's actually more efficient (and probably cleaner) to obtain all necessary IDs up-front:

```
private Button btn;
private EditText textField;
...
btn = (Button) findViewById(R.id.myButton);
textField = (EditText) findViewById(R.id.myTextField);
btn.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View view)
    {
        ...
    }
});
```