Flickr Browser Changes For Android 1.5.1

Introduction

Eventually I will re-create the Flickr videos completely for Android Marshmallow. For now, this document is what you need to do to get the app working with Android Marshmallow.

Extra information

I have created two videos which might be of interest to you. You can watch this, or just follow the information in this guide (note that the videos don't contain everything int his document, so consider the document to be the most up to date).

What has changed in Android Studio 1.5.1

Click Here

and

Changes to be made to cover Android Marshmallow for the flickr app.

Click Here

Below Are The Following Things you will need to change from the original recorded videos

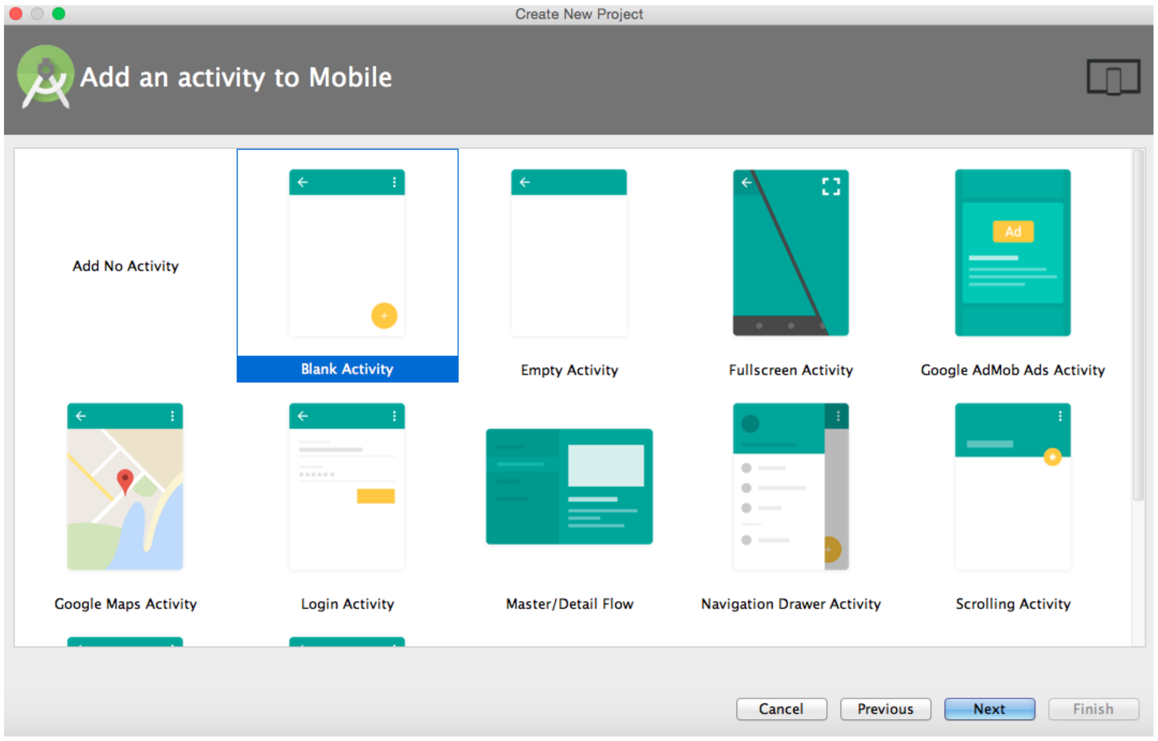
Here are the following items to keep in mind when you are building the app in Android Studio 1.5.1. ALSO IMPORTANT NOTE! Please when the lecture videos comes to the part where it edits the following files please proceed to edit the these following files instead.

activity_main.xml===➔content_main.xml
layout_search.xml===➔content_search.xml
photo_details.xml===➔content_view_photo_details.xml

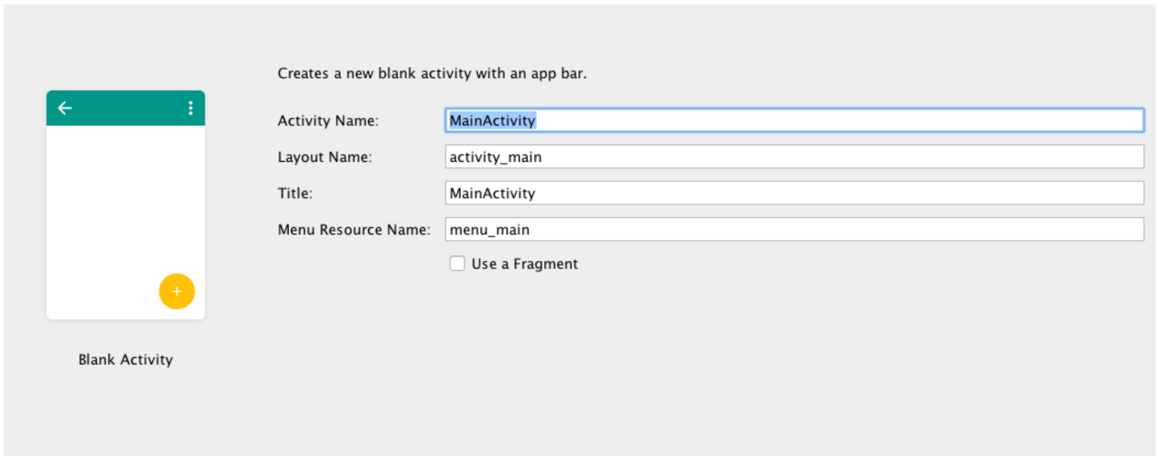
With that out of the way, lets begin!

Things to do when creating the project in <u>Lecture 61: Flickr Api, Feeds, Introduction to Json and More</u>

When creating a new Android Studio Project for the Flickr Browser App, choose **Blank Activity** since **Blank Activity with Fragment** does not exist anymore.



On the next screen make sure the **Use a Fragment** option and left unchecked. And press ok to create the project.


Creates a new blank activity with an app bar.

Activity Name: MainActivity

Layout Name: activity_main

Title: MainActivity

Menu Resource Name: menu_main

☐ Use a Fragment

Commenting out the Floating Action Button and Creating Blank Activity.

For every Blank Activity you create throughout the entire project you need to comment out the following lines of code. Also when creating **blank activity with fragment** please choose instead a Blank Activity and don't check the checkbox for **Use a Fragment** when you come to the screen when your adding the Name

Example if you have a created a Blank Activity called MainActivity you need to comment out the following lines in the MainActivity.java file. Just select the lines below and press CMD+/ or CTRL+/ if your in a Windows. Java files are stored in the **App>Src>Main>Java>Packagename** of the project.

```
FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();
    }
});
```

After commenting the following lines of code the current inside of your MainActivity.java should now look like the following below.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar)
findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
//
//        FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
//        fab.setOnClickListener(new View.OnClickListener()
{
//            @Override
//            public void onClick(View view) {
//                Snackbar.make(view, "Replace with your
own action", Snackbar.LENGTH_LONG)
//                        .setAction("Action",
null).show();
//            }
//        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action
bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action
bar will
        // automatically handle clicks on the Home/Up
button, so long
        // as you specify a parent activity in
AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

Then you need to open up the activity's corresponding activity layout xml file which in this example is **activity_main.xml** and comment out these lines of code. Xml layout files are always stored in **App>Src>Main>Res>layout** of the project.

```
<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_margin="@dimen/fab_margin"
    android:src="@android:drawable/ic_dialog_email"/>
```

You can comment them by selecting the lines of code and press CMD+/ or CTRL+/ for windows.

This is what your **activity_main.xml** contents should look like after commenting.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout

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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context=".MainActivity">

<android.support.design.widget.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/AppTheme.AppBarOverlay">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/AppTheme.PopupOverlay"/>

</android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main"/>

    <!--
<android.support.design.widget.FloatingActionButton-->
```

```
        <!--android:id="@+id/fab"-->
        <!--android:layout_width="wrap_content"-->
        <!--android:layout_height="wrap_content"-->
        <!--android:layout_gravity="bottom|end"-->
        <!--android:layout_margin="@dimen/fab_margin"-->
        <!--
android:src="@android:drawable/ic_dialog_email"/>-->

</android.support.design.widget.CoordinatorLayout>
```

Apply this every time you create a new Blank Activity. You must comment out the following lines of code in both the java file and the corresponding activity xml file of it.

Editing content_main.xml and content xml files in general

When the lecture shows editing **activity the of any xml file** you should instead edit the **corresponding content xml file.** The same applies to any new Blank Activities you create, instead of editing the activity xml file for that activity; you should instead edit the content xml file.

When editing the **content_main.xml file or any content xml file**. Please do not delete the following highlighted in **red**.

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

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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"

android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"

app:layout_behavior="@string/appbar_scrolling_view_behavior
"

tools:context="org.example.android.flickrapp.MainActivity"
    tools:showIn="@layout/activity_main">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"/>
</RelativeLayout>
```

Your **content_main.xml** should now look like the following:

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

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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

app:layout_behavior="@string/appbar_scrolling_view_behavior
"

tools:context="org.example.android.flickrapp.MainActivity"
    tools:showIn="@layout/activity_main">

<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"/>

</RelativeLayout>
```

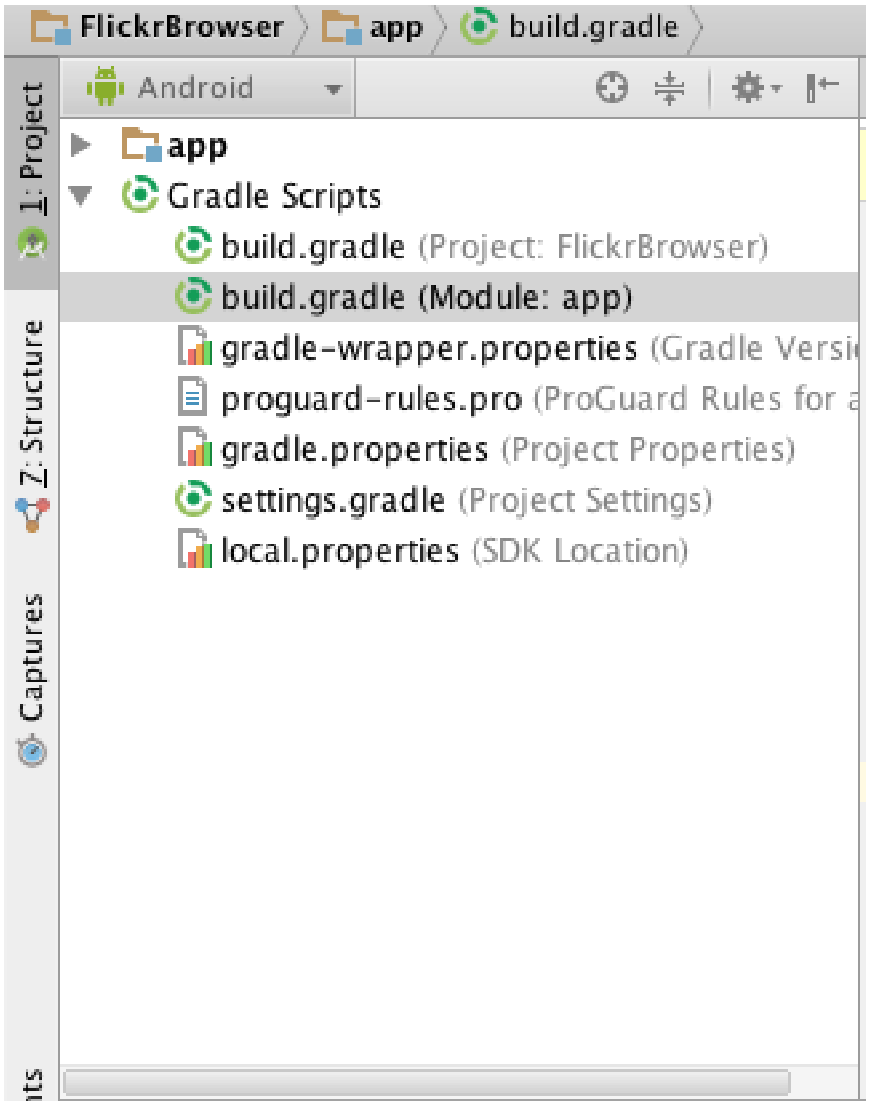
This concludes the change need to be made while on <u>Lecture 61: Flickr Api, Feeds, Introduction to Json and More</u>

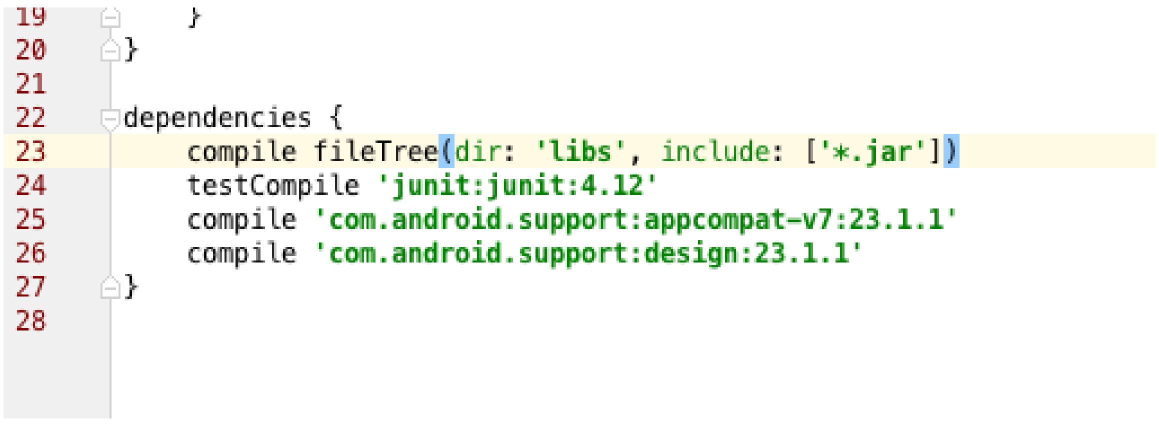
Things to do when creating the project in Lecture 67: Creating The Basic GUI

build.gradle Dependencies

Important Note: *By doing the following step in this document you no longer need to download the picasso jar file and install it into the project.*

In the build gradle file found in **GradleScripts>build.gradle(Module: App)** in the project pane window of Android Studio. Also make sure to set the view to **Android**.


FlickrBrowser
app
build.gradle
Android
1: Project
7: Structure
Captures
app
Gradle Scripts
build.gradle (Project: FlickrBrowser)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Versi
proguard-rules.pro (ProGuard Rules for a
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)



```
19          }
20      }
21
22      dependencies {
23          compile fileTree(dir: 'libs', include: ['*.jar'])
24          testCompile 'junit:junit:4.12'
25          compile 'com.android.support:appcompat-v7:23.1.1'
26          compile 'com.android.support:design:23.1.1'
27      }
28
```

Then make sure to add the following dependencies after line 26 of the **build.gradle** file. If they have not been added yet.

```
compile 'com.android.support:recyclerview-v7:23.1.1'
compile 'com.squareup.picasso:picasso:2.5.2'
compile 'com.android.support:cardview-v7:23.1.1'
```

Then click on Sync Now link on the upper right corner to start syncing the build.gradle.

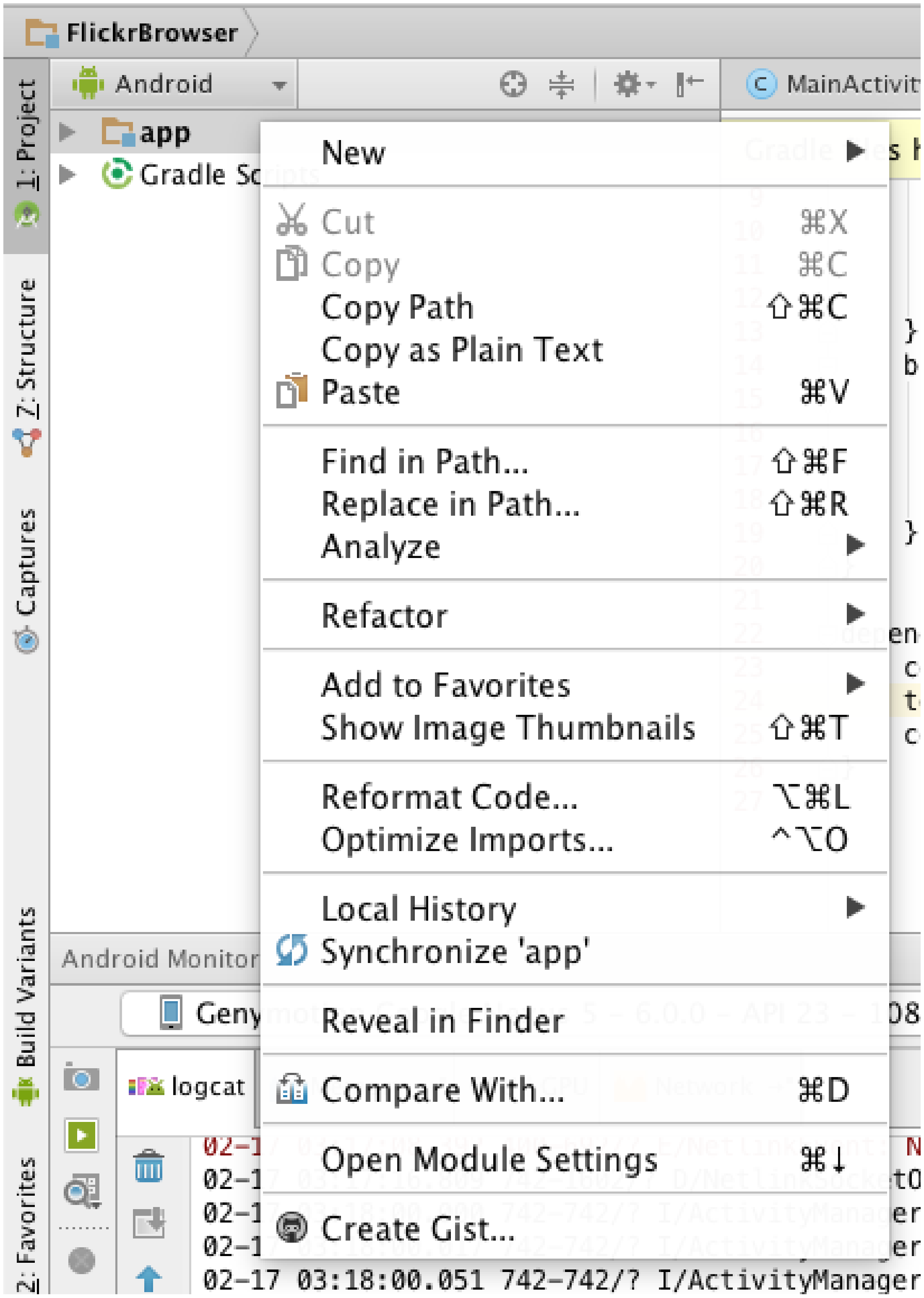
There is an alternative way to adding dependencies first you have to delete the following line from your build.gradle.

```
compile 'com.android.support:appcompat-v7:23.1.1'
```

These are the only items that should be left on your dependencies area of your build.gradle file

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:design:23.1.1'
}
```

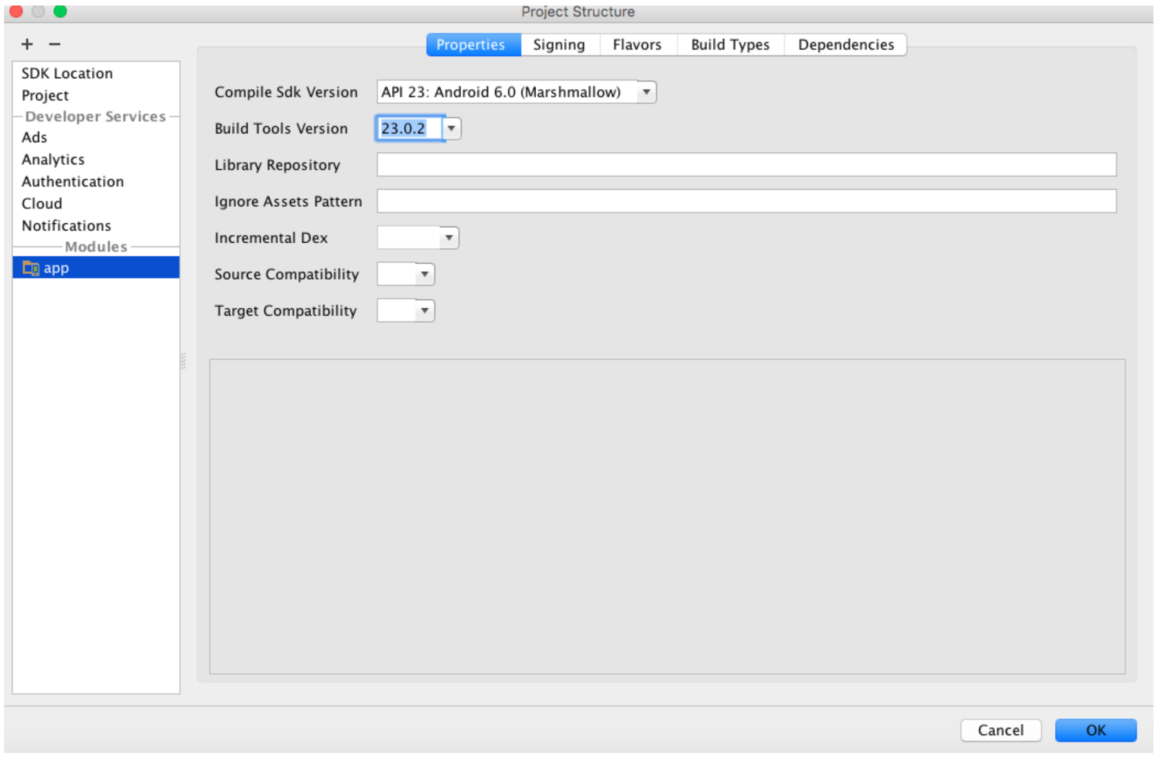
Next, go to your project pane, right click on the top most folder in the labeled **app**.



```
02-17 03:18:00.051 742-742/? I/ActivityManager
```

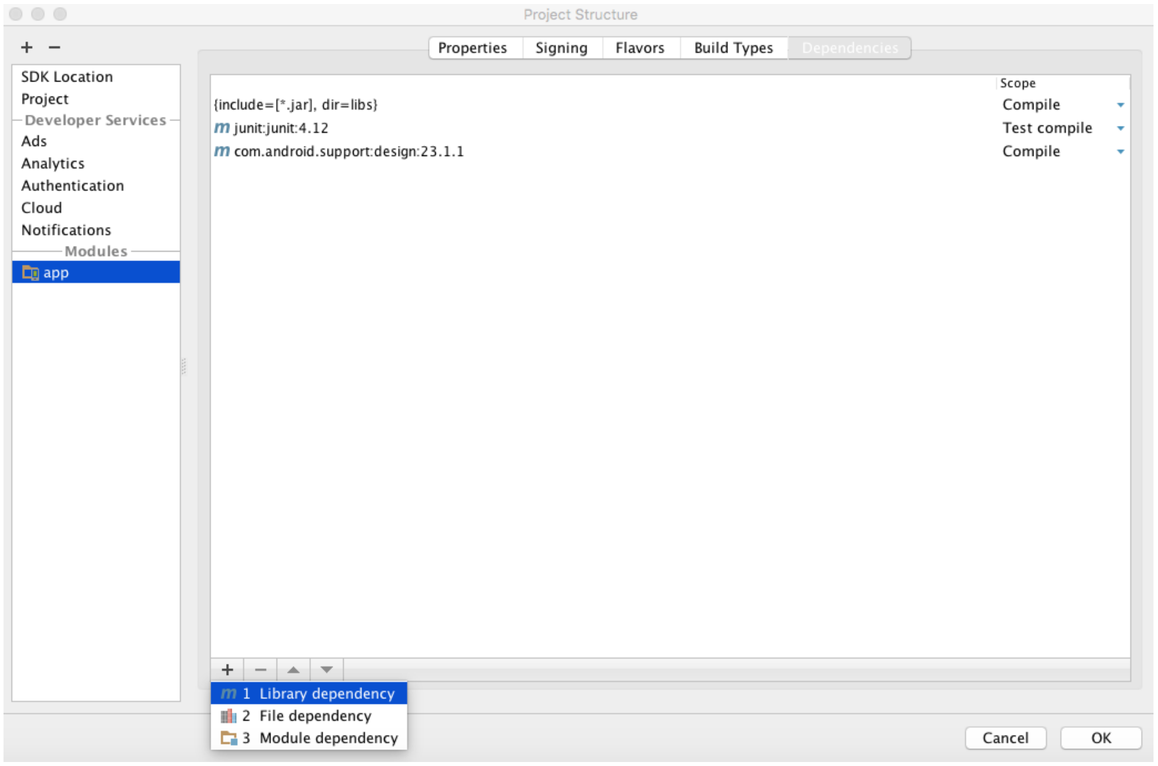
Right click on it and select **Open Module Settings**.

In the open module settings window, select **app** on the **left menu**. Make sure that the latest Build Tools Version is selected, which in this current time of documentation is **23.0.2**.



Compile Sdk Version: API 23: Android 6.0 (Marshmallow)
Build Tools Version: 23.0.2
Library Repository
Ignore Assets Pattern
Incremental Dex
Source Compatibility
Target Compatibility

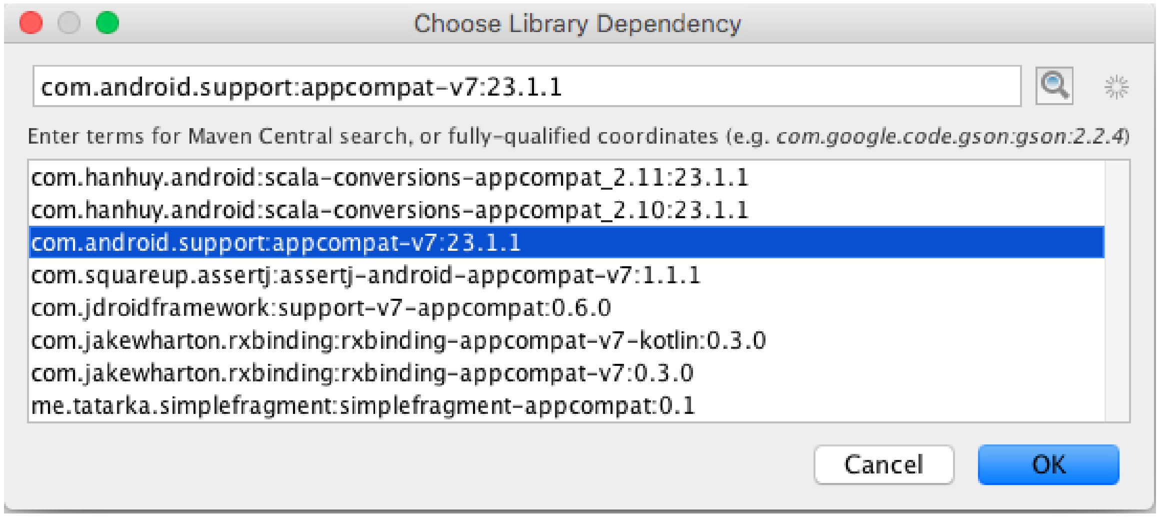
Click the **Dependencies** tab, which is where you we will add the dependencies with correct latest versions. First click the plus sign in the lower left, and choose **Library** dependency.



	Scope
{include=[*.jar], dir=libs}	Compile
junit:junit:4.12	Test compile
com.android.support:design:23.1.1	Compile

A small window will pop out for searching the Library depdencies we want to add. First seach for **appcompat** by typing **app compat** in the search field and press enter. Wait for it to finish searching. It will then bring back a list of all the appcompats it could find.

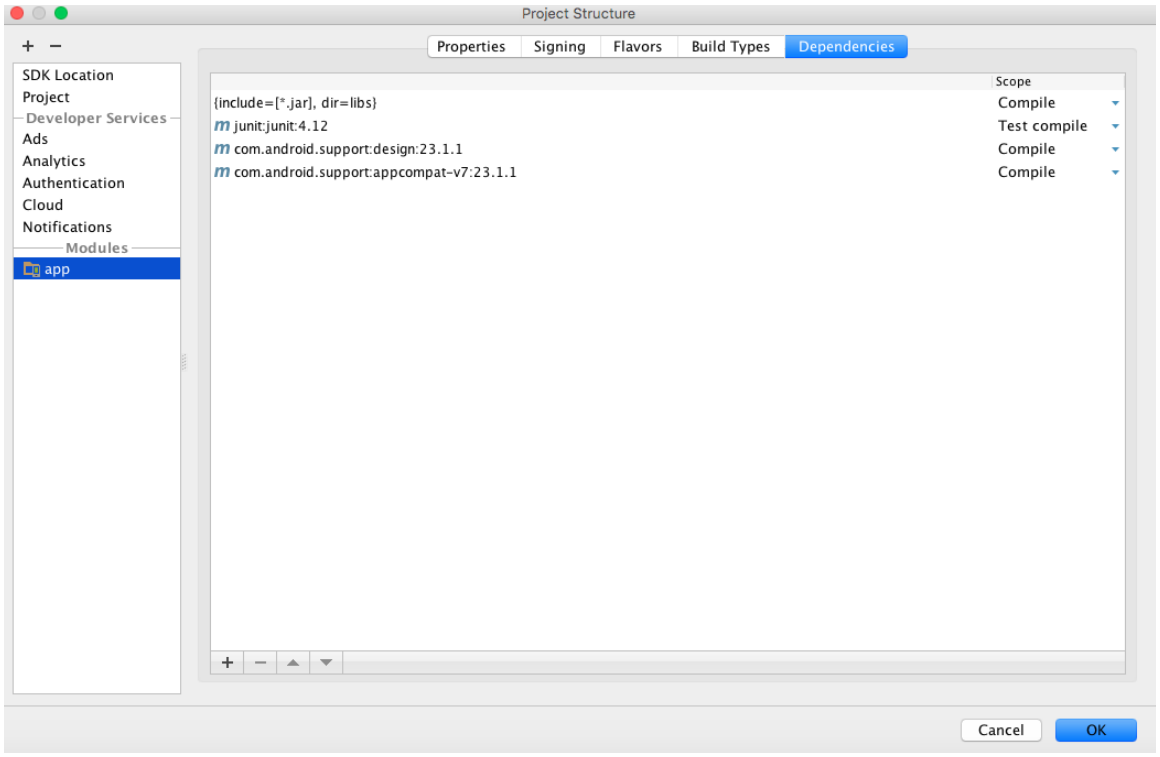
The one you need to choose is the most latest version for the **com.android.support** which in this case is 23.1.1. So select that one and press Ok, it will then get added to the list in the dependencies tab.



com.android.support:appcompat-v7:23.1.1

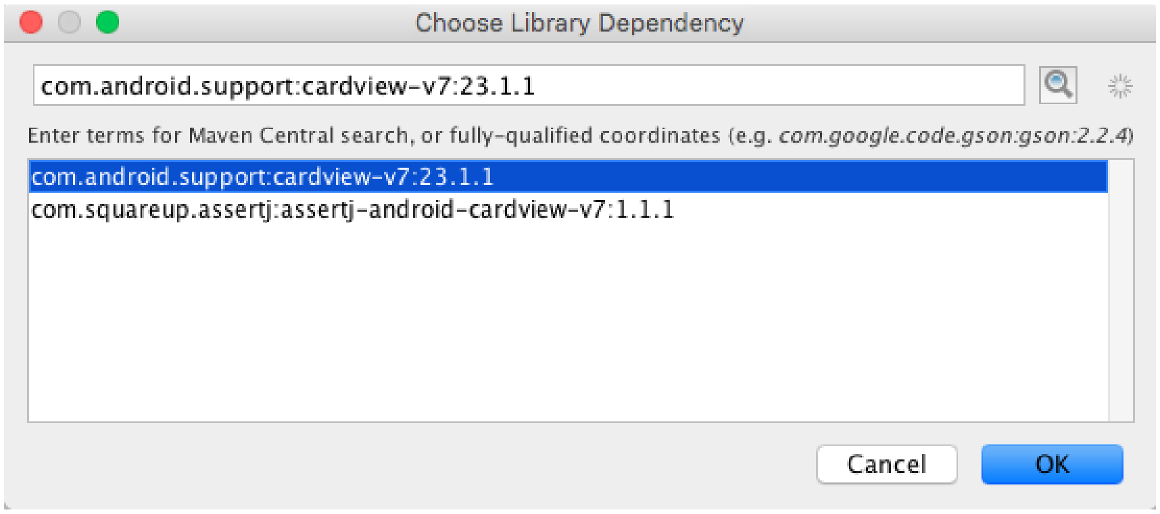
Enter terms for Maven Central search, or fully-qualified coordinates (e.g. *com.google.code.gson:gson:2.2.4*)

```
com.hanhuy.android:scala-conversions-appcompat_2.11:23.1.1
com.hanhuy.android:scala-conversions-appcompat_2.10:23.1.1
com.android.support:appcompat-v7:23.1.1
com.squareup.assertj:assertj-android-appcompat-v7:1.1.1
com.jdroidframework:support-v7-appcompat:0.6.0
com.jakewharton.rxbinding:rxbinding-appcompat-v7-kotlin:0.3.0
com.jakewharton.rxbinding:rxbinding-appcompat-v7:0.3.0
me.tatarka.simplefragment:simplefragment-appcompat:0.1
```



	Scope
{include=[*.jar], dir=libs}	Compile
junit:junit:4.12	Test compile
com.android.support:design:23.1.1	Compile
com.android.support:appcompat-v7:23.1.1	Compile

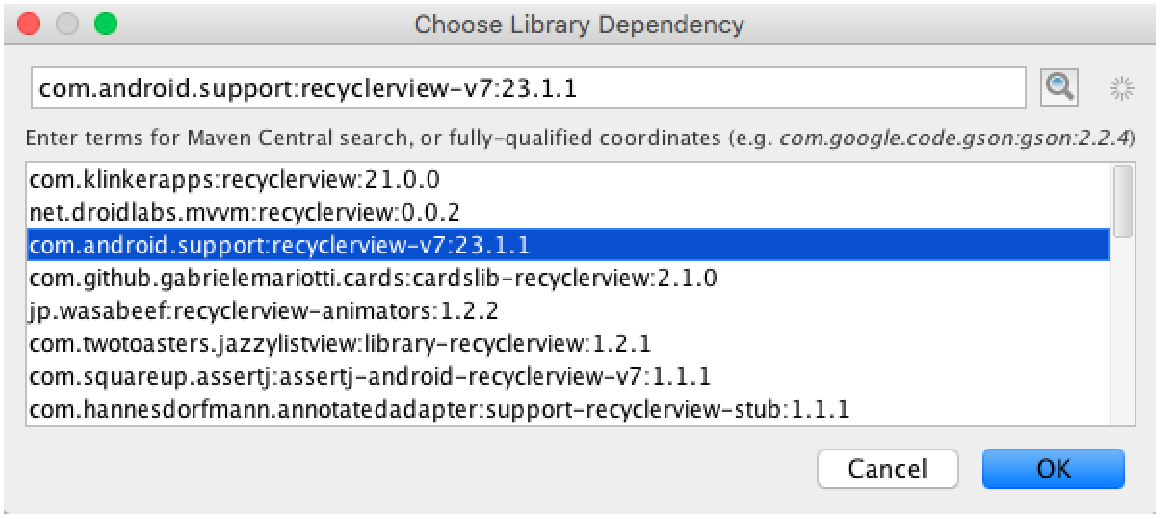
Next, we need to add dependencies for **cardview** and **Recyclerview**. So search for those again and add them. Be careful thought because you might pick the wrong ones, so make sure the ones you have selected have the **com.android.support** on it.



com.android.support:cardview-v7:23.1.1

Enter terms for Maven Central search, or fully-qualified coordinates (e.g. *com.google.code.gson:gson:2.2.4*)

com.android.support:cardview-v7:23.1.1
com.squareup.assertj:assertj-android-cardview-v7:1.1.1

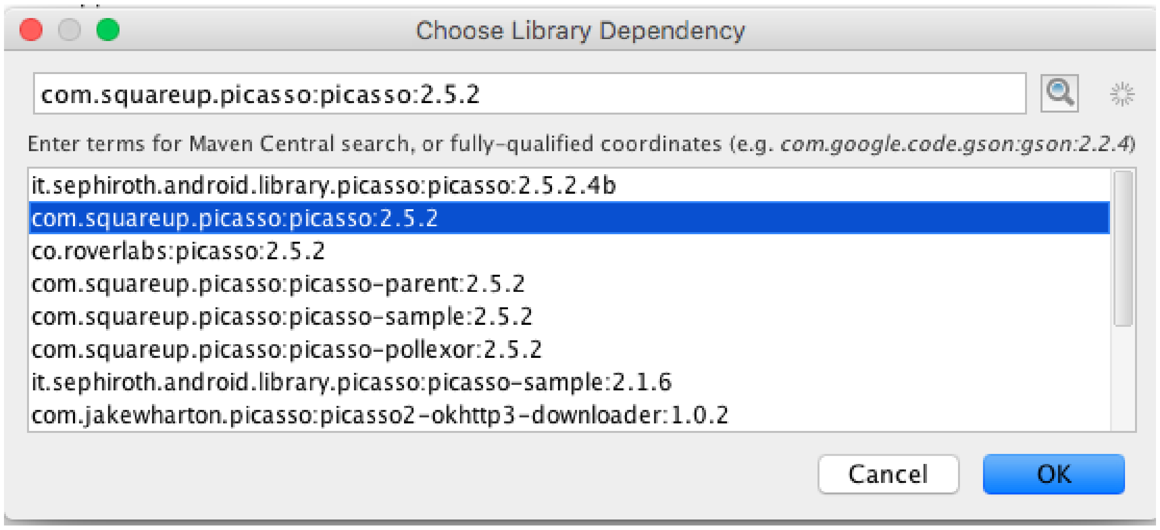


com.android.support:recyclerview-v7:23.1.1

Enter terms for Maven Central search, or fully-qualified coordinates (e.g. *com.google.code.gson:gson:2.2.4*)

```
com.klinkerapps:recyclerview:21.0.0
net.droidlabs.mvvm:recyclerview:0.0.2
com.android.support:recyclerview-v7:23.1.1
com.github.gabrielemariotti.cards:cardslib-recyclerview:2.1.0
jp.wasabeef:recyclerview-animators:1.2.2
com.twotoasters.jazzylistview:library-recyclerview:1.2.1
com.squareup.assertj:assertj-android-recyclerview-v7:1.1.1
com.hannesdorfmann.annotatedadapter:support-recyclerview-stub:1.1.1
```

And lastly we need to add the dependency for **picasso** so type that into the search box and add it. **Be sure to pick the one that starts with com.squareup.picasso and of course pick the latest version.**

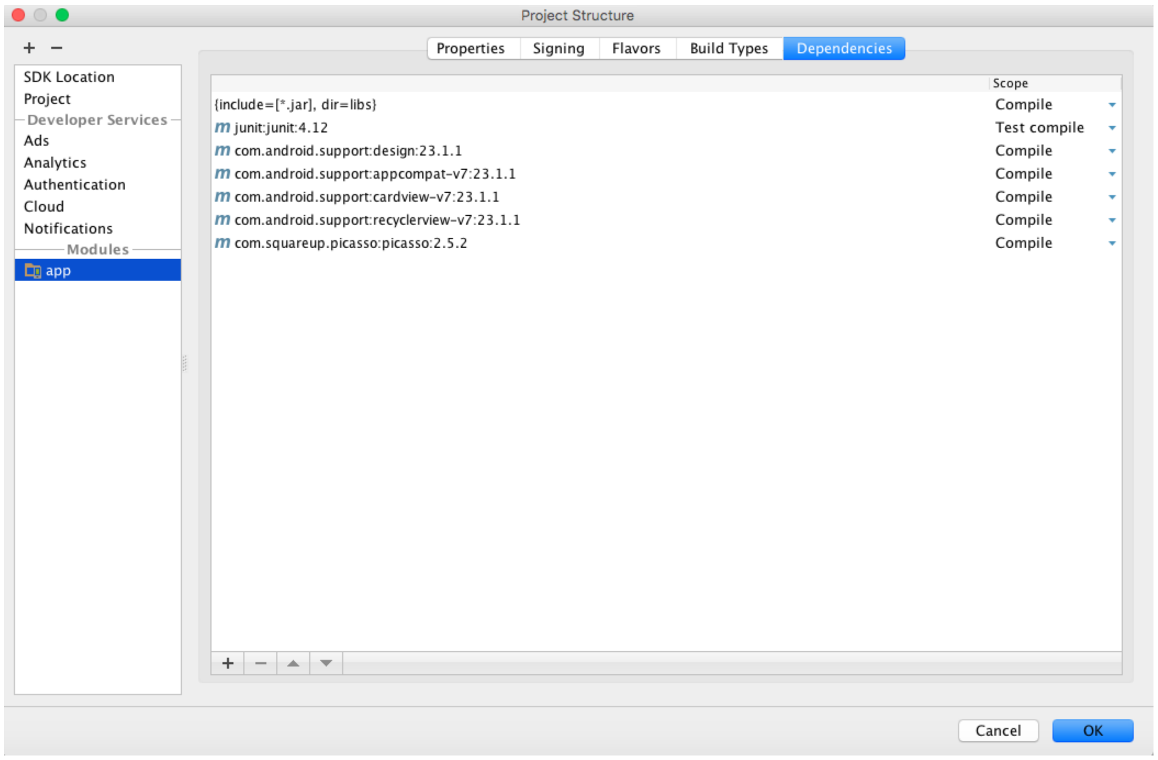


com.squareup.picasso:picasso:2.5.2

Enter terms for Maven Central search, or fully-qualified coordinates (e.g. *com.google.code.gson:gson:2.2.4*)

```
it.sephiroth.android.library.picasso:picasso:2.5.2.4b
com.squareup.picasso:picasso:2.5.2
co.roverlabs:picasso:2.5.2
com.squareup.picasso:picasso-parent:2.5.2
com.squareup.picasso:picasso-sample:2.5.2
com.squareup.picasso:picasso-pollexor:2.5.2
it.sephiroth.android.library.picasso:picasso-sample:2.1.6
com.jakewharton.picasso:picasso2-okhttp3-downloader:1.0.2
```

Once you have all 4 depdencies showing up the in the window press Ok to add them to the build.gradle file.



	Scope
{include=[*.jar], dir=libs}	Compile
junit:junit:4.12	Test compile
com.android.support:design:23.1.1	Compile
com.android.support:appcompat-v7:23.1.1	Compile
com.android.support:cardview-v7:23.1.1	Compile
com.android.support:recyclerview-v7:23.1.1	Compile
com.squareup.picasso:picasso:2.5.2	Compile

The build.gradle file will add the dependencies and it will automatically start syncing them. Wait for it to finish the syncing process and your done.

Your **build.gradle** contents should now look like the following below.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.2"

    defaultConfig {
        applicationId "org.example.android.flickrbrowser"
        minSdkVersion 16
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:design:23.1.1'
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:cardview-v7:23.1.1'
    compile 'com.android.support:recyclerview-v7:23.1.1'
    compile 'com.squareup.picasso:picasso:2.5.2'
}
```

In the end you should have Appcompat, RecyclerView, CardView, and Picasso in your **build.gradle** dependencies.

This concludes the change need to be made while on Lecture 67: Creating The Basic GUI

Things to do when creating the project Lecture 68: Continue Creating The Basic GUI

You no longer need to comment out the following code on MainActivity.java shown in the lecture:

```
        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction()
                    .add(R.id.container, new 
PlaceholderFragment())
                    .commit();
        }
```

As we created this project with a Blank Activity without a fragment.

This concludes the change need to be made while on Lecture 68: Continue Creating The Basic GUI

Things to do when creating the project Lecture 69: Finish Off The GUI Interface

Editing The content_main.xml

In this lecture you will be editing the **content_main.xml** file instead of the **activity_main.xml** file as show in the video. Your **content_main.xml** should contain currently the following:

Please do not delete the lines highlighted in red. Only delete the lines that are not highlighted.

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

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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

app:layout_behavior="@string/appbar_scrolling_view_behavior
"

tools:context="org.example.android.flickrbrowser.MainActivi
ty"
    tools:showIn="@layout/activity_main">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"/>

</RelativeLayout>
```

Your **content_main.xml** should now only contain the following:

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

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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
app:layout_behavior="@string/appbar_scrolling_view_behavior
"

tools:context="org.example.android.flickrbrowser.MainActivi
ty"
    tools:showIn="@layout/activity_main">

</RelativeLayout>
```

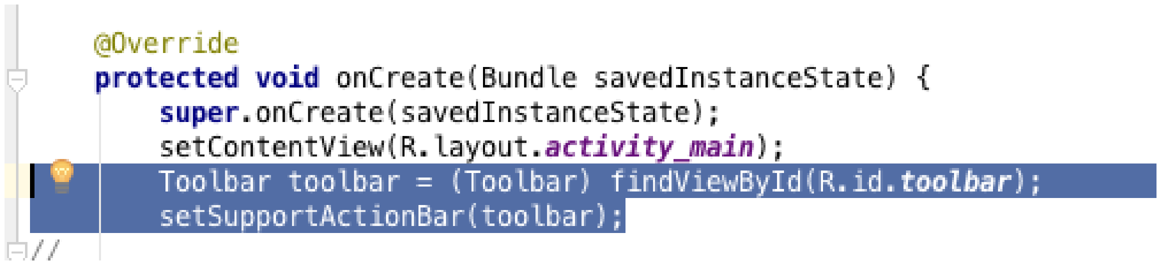
Then follow the lecture on the items to be added in. So the final **content_main.xml** should be. The codes added are highlighted in red.

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

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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="10dp"

app:layout_behavior="@string/appbar_scrolling_view_behavior
"

tools:context="org.example.android.flickrbrowser.MainActivi
ty"
    tools:showIn="@layout/activity_main">

    <view
        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class="android.support.v7.widget.RecyclerView"/>

</RelativeLayout>
```

We are now done with changes to **content_main.xml**

Avoid Deleting These Lines In MainActivity.java

Next is the **MainActivity.java.** When you get to the part of the lecture of editing the MainActivity.java file please **DO NOT** delete the following lines of code highlighted.



```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
//
```

No Need To Re-Add The Picasso Dependencies

Because on the earlier part of the documentation you no longer need to insert picasso dependencies on the build.gradle file.

This concludes the change need to be made while on <u>Lecture 69: Finish Off The GUI Interface</u>

Things to do when creating the project <u>Lecture 70: Creating The App's Theme</u>

Editing The colors.xml

In this lecture you will find that you already have a colors.xml file in that case just replace all the contents in their as to what is shown in the lecture.

No Need For AppCompat And CardView

Since we already added the appcompat and cardview dependencies in the earlier video you can skip this part of the lecture.

Editing The styles.xml

In this lecture you will find that you already have a **styles.xml** file in that case just replace all the contents in their as to what is shown in the lecture.

This concludes the change need to be made while on <u>Lecture 70: Creating The App's Theme</u>

Things to do when creating the project Lecture 71: Setting Up The Toolbar and Styles

Editing The v21 styles.xml

In this lecture you will find that you already have a styles.xml file in the v21 folder in that case just replace all the contents in their as to what is shown in the lecture.

This concludes the change need to be made while on Lecture 71: Setting Up The Toolbar and Styles

Things to do when creating the project Lecture 72: Let's Create The BaseActivity Class

Editing The BaseActivity Class

When editing the **BaseActivity.java** file in this lecture make sure you use extends **AppCompatActivity** instead of **ActionBarActivity.** Since ActionBarActivity has been depecreated and not in use.

So your extends should be AppCompatActivity.

```
public class BaseActivity extends AppCompatActivity {

    private Toolbar mToolbar;

    protected Toolbar activateToolbar() {
        if(mToolbar == null) {
            mToolbar = (Toolbar)
findViewById(R.id.app_bar);
            if(mToolbar != null) {
                setSupportActionBar(mToolbar);
            }
        }
        return mToolbar;
    }
}
```

Comment The Lines In activity_main.xml

Please comment out the lines of code in activity_main.xml file to avoid problems.

```
<android.support.design.widget.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/AppTheme.AppBarOverlay">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/AppTheme.PopupOverlay"/>

</android.support.design.widget.AppBarLayout>
```

Your activity_main.xml should now look like the following:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout

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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"

tools:context="org.example.android.flickrbrowser.MainActivi
ty">

    <!--<android.support.design.widget.AppBarLayout-->
        <!--android:layout_width="match_parent"-->
        <!--android:layout_height="wrap_content"-->
        <!--android:theme="@style/AppTheme.AppBarOverlay">-
->

        <!--<android.support.v7.widget.Toolbar-->
            <!--android:id="@+id/toolbar"-->
            <!--android:layout_width="match_parent"-->
            <!--
android:layout_height="?attr/actionBarSize"-->
            <!--android:background="?attr/colorPrimary"-->
            <!--
app:popupTheme="@style/AppTheme.PopupOverlay"/>-->

    <!--</android.support.design.widget.AppBarLayout>-->

    <include layout="@layout/content_main"/>

    <!--
<android.support.design.widget.FloatingActionButton-->
        <!--android:id="@+id/fab"-->
        <!--android:layout_width="wrap_content"-->
        <!--android:layout_height="wrap_content"-->
        <!--android:layout_gravity="bottom|end"-->
        <!--android:layout_margin="@dimen/fab_margin"-->
        <!--
android:src="@android:drawable/ic_dialog_email"/>-->

</android.support.design.widget.CoordinatorLayout>
```

Removing Some Line of Code in the AndroidManifest.xml

Open up the AndroidManifest.xml file in **App>Src>Main** and look for the following line of code. It should be inside the first <activity> tag for .MainActivity. Look for the highlighted code

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/Theme.Flickr">
    <activity
        android:name=".MainActivity"
        android:label="@string/app_name"
        android:theme="@style/AppTheme.NoActionBar">
        <intent-filter>
            <action
android:name="android.intent.action.MAIN"/>

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

Delete the line of code highlighted in **red** and your current AndroidManifest.xml file should only look like this for now.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest package="org.example.android.flickrbrowser"

xmlns:android="http://schemas.android.com/apk/res/android">

    <uses-permission
android:name="android.permission.INTERNET"/>
    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/Theme.Flickr">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN"/>
```

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

</manifest>
```

Commenting Some Lines Of Code In MainActivity.java

Open up the MainActivity.java file and then comment the following lines of code found on onCreate method.

```
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
```

The onCreated method of your MainActivity.java should now look like the following.

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
//        Toolbar toolbar = (Toolbar)
findViewById(R.id.toolbar);
//        setSupportActionBar(toolbar);
//
//        FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
//        fab.setOnClickListener(new View.OnClickListener()
{
//            @Override
//            public void onClick(View view) {
//                Snackbar.make(view, "Replace with your
own action", Snackbar.LENGTH_LONG)
//                        .setAction("Action",
null).show();
//            }
//        });

//        GetRawData theRawData = new
GetRawData("https://api.flickr.com/services/feeds/photos_pu
blic.gne?tags=android,lollipop&format=json&nojsoncallback=1
");
        activateToolbar();

        mRecyclerView = (RecyclerView)
findViewById(R.id.recycler_view);
        mRecyclerView.setLayoutManager(new
LinearLayoutManager(this));

        ProcessPhotos processPhotos = new
ProcessPhotos("google",true);
        processPhotos.execute();
    }
```

This concludes the change need to be made while on
<u>Lecture 72: Let's Create The BaseActivity Class</u>

Things to do when creating the project Lecture 73: Create The Layout XML For The SearchActivity Class

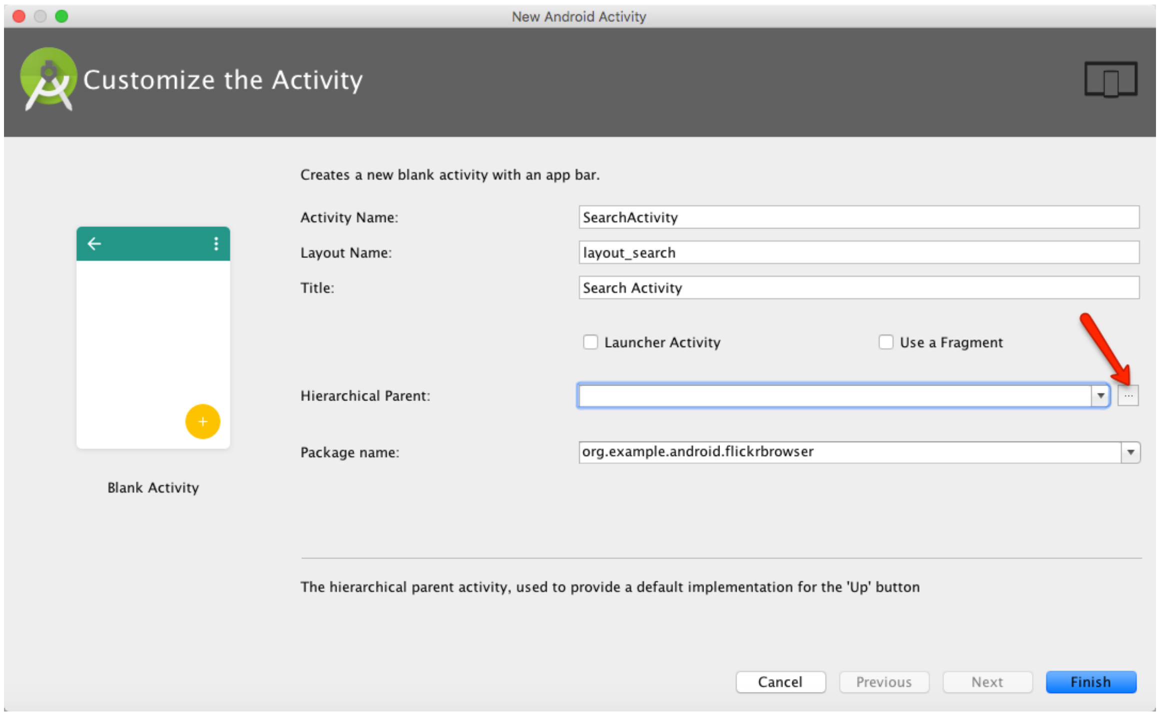
NO Need To Delete The fragment_main.xml

In this lecture you don't need to delete the fragment_main.xml since the fragment was not checked when creating the Blank Activity of this project.

Creating The SearchActivity.java

When creating the SearchActivity.java just proceed with the creation despite no option to add the **menu_search.xml** file. Then do the following edits

When adding the Hierarchical Parent make sure to click the browse button and select MainActivity.



Creates a new blank activity with an app bar.

Activity Name: SearchActivity

Layout Name: layout_search

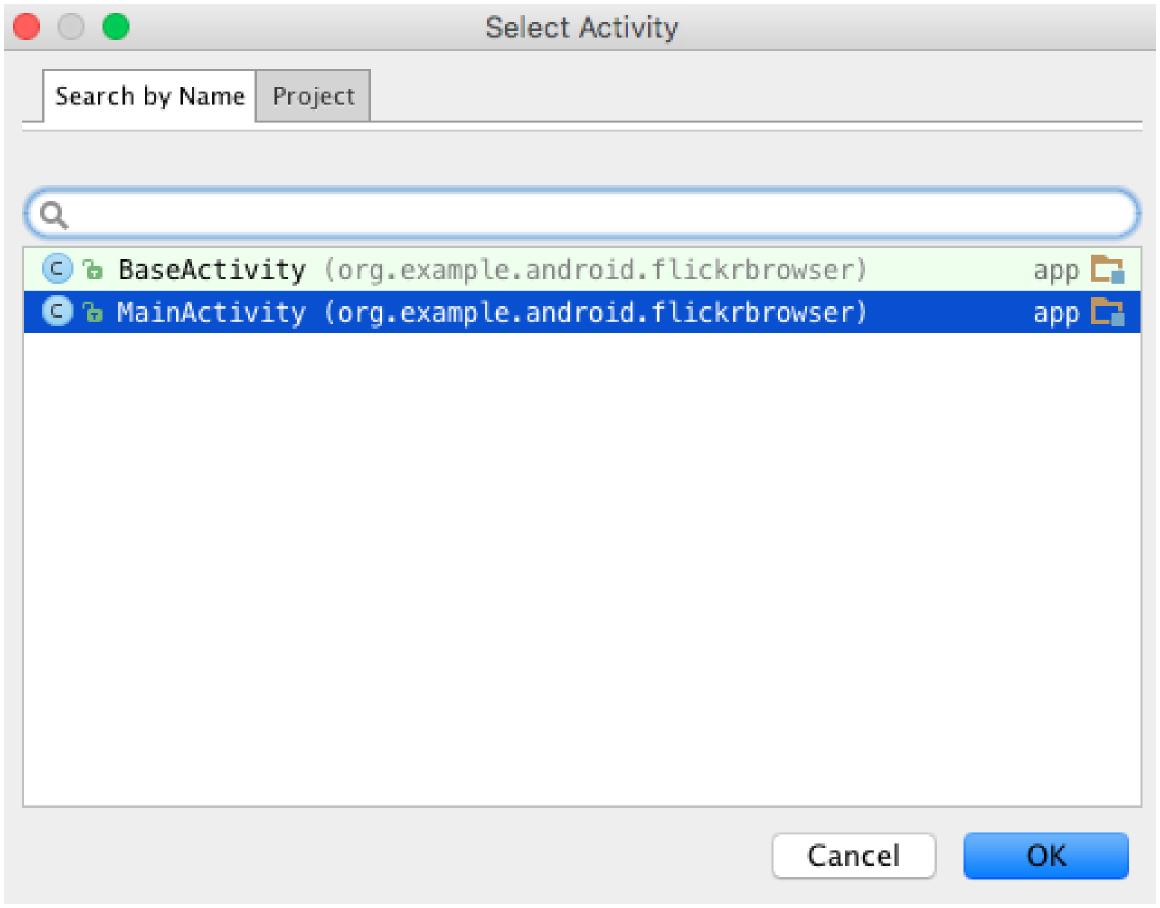
Title: Search Activity

☐ Launcher Activity ☐ Use a Fragment

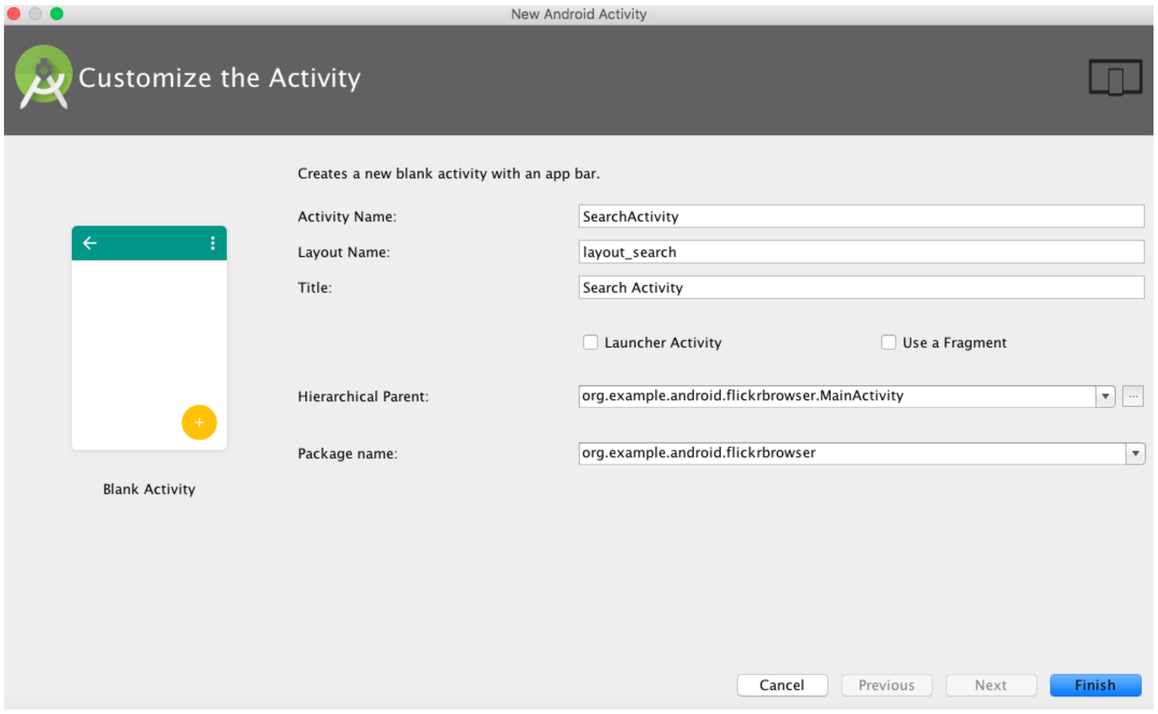
Hierarchical Parent:

Package name: org.example.android.flickrbrowser

The hierarchical parent activity, used to provide a default implementation for the 'Up' button



BaseActivity (org.example.android.flickrbrowser) app
MainActivity (org.example.android.flickrbrowser) app



Customize the Activity

Creates a new blank activity with an app bar.

Activity Name:	SearchActivity
Layout Name:	layout_search
Title:	Search Activity
	☐ Launcher Activity ☐ Use a Fragment
Hierarchical Parent:	org.example.android.flickrbrowser.MainActivity
Package name:	org.example.android.flickrbrowser

Congrats you have now created the SearchActivity.

Commenting out the Floating Action Button and Creating Blank Activity.

For every Blank Activity you create throughout the entire project you need to comment out the following lines of code.

Example if you have a created a Blank Activity called SearchActivity you need to comment out the following lines in the SearchActivity.java file. Just select the lines below and press CMD+/ or CTRL+/ if your in a Windows. Java files are stored in the **App>Src>Main>Java>Packagename** of the project.

```
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolbar);

FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();
    }
});
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
```

After commenting the following lines of code the current inside of your MainActivity.java should now look like the following below.

```
public class SearchActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_search);
//        Toolbar toolbar = (Toolbar)
findViewById(R.id.toolbar);
//        setSupportActionBar(toolbar);
//
//        FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
//        fab.setOnClickListener(new View.OnClickListener()
{
//            @Override
//            public void onClick(View view) {
//                Snackbar.make(view, "Replace with your
own action", Snackbar.LENGTH_LONG)
//                      .setAction("Action",
null).show();
//          }
//      });
//
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    }

}
```

Then you need to open up the activity's corresponding activity xml file which in this example is **layout_search.xml** and comment out these lines of code. Xml layout files are always stored in **App>Src>Main>Res>layout** of the project.

```
<android.support.design.widget.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/Theme.Flickr.AppBarOverlay">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/Theme.Flickr.PopupOverlay"/>

</android.support.design.widget.AppBarLayout>

<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_margin="@dimen/fab_margin"
    android:src="@android:drawable/ic_dialog_email"/>
```

You can comment them by selecting the lines of code and press CMD+/ or CTRL+/ for windows.

This is what your **activity_main.xml** contents should look like after commenting.

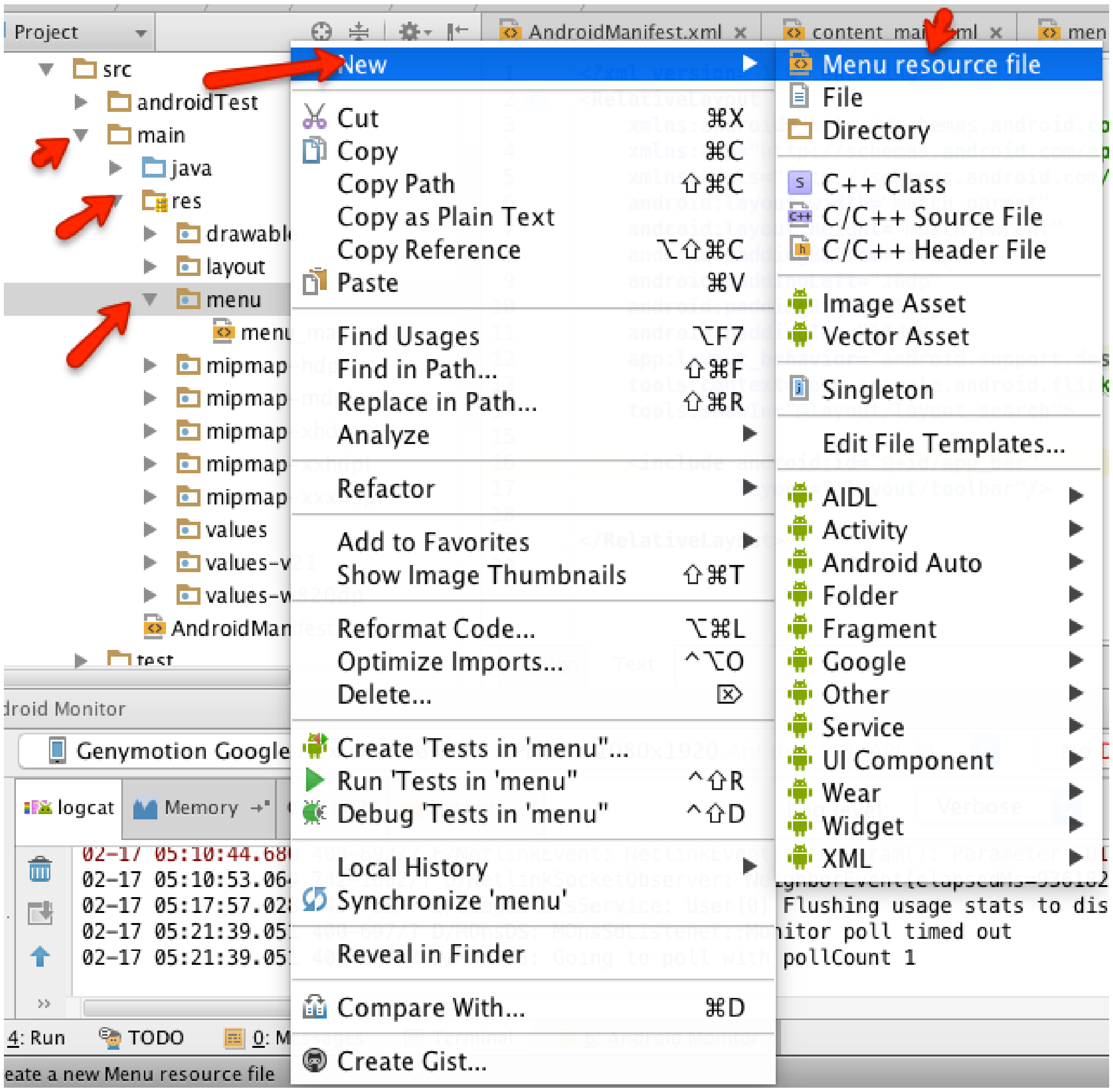
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout

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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"

tools:context="org.example.android.flickrbrowser.SearchActi
vity">

    <!--<android.support.design.widget.AppBarLayout-->
        <!--android:layout_width="match_parent"-->
        <!--android:layout_height="wrap_content"-->
        <!--
android:theme="@style/Theme.Flickr.AppBarOverlay">-->

        <!--<android.support.v7.widget.Toolbar-->
            <!--android:id="@+id/toolbar"-->
            <!--android:layout_width="match_parent"-->
            <!--
android:layout_height="?attr/actionBarSize"-->
            <!--android:background="?attr/colorPrimary"-->
            <!--
app:popupTheme="@style/Theme.Flickr.PopupOverlay"/>-->

    <!--</android.support.design.widget.AppBarLayout>-->

    <include layout="@layout/content_search"/>

    <!--
<android.support.design.widget.FloatingActionButton-->
        <!--android:id="@+id/fab"-->
        <!--android:layout_width="wrap_content"-->
        <!--android:layout_height="wrap_content"-->
        <!--android:layout_gravity="bottom|end"-->
        <!--android:layout_margin="@dimen/fab_margin"-->
        <!--
android:src="@android:drawable/ic_dialog_email"/>-->

</android.support.design.widget.CoordinatorLayout>
```

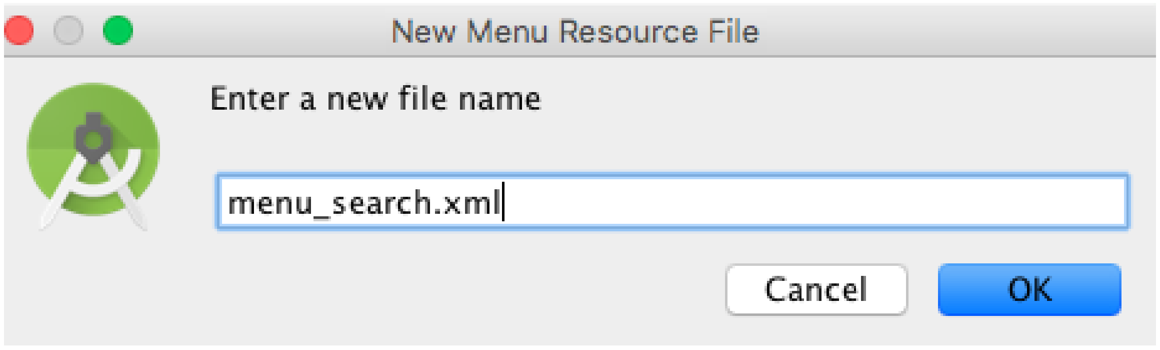
Apply this every time you create a new Blank Activity. You must comment out the following lines of code in both the java file and the corresponding activity xml file of it.

Creating The menu_search.xml

Since when creating the SearchActivity it didn't gave us the option to add a menu xml file you need to create it manually so go to the **Src>Main>Res>Menu.** Right click and create a new resource file and call it **menu_search.xml**



```
02-17 05:10:44.68
02-17 05:10:53.06
02-17 05:17:57.02                Flushing usage stats to dis
02-17 05:21:39.05                itor poll timed out
02-17 05:21:39.05                pollCount 1
```



menu_search.xml

Once you have created the file then add in the contents of it that is show in the lecture.

This concludes the change need to be made while on <u>Lecture 73: Create The Layout XML For The SearchActivity Class</u>

Things to do when creating the project Lecture 74: Continue Working On The SearchActivity Class

No onCreateOptionsMenu on SearchActivity.java

In this lecture you will by adding some code to the onCreateOptionsMenu method but with the new version of Android Marshmallow and Android Studio it seems it doesn't add it when creating a new Blank Activity. So you need to add the following code below the **onCreate** method of the SearchActivity.java

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar
if it is present.
    getMenuInflater().inflate(R.menu.menu_search, menu);

    return true;
}
```

So your SearchActivity.java should now look like the following:

```
public class SearchActivity extends BaseActivity {

    private SearchView mSearchView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout_search);
//          Toolbar toolbar = (Toolbar)
findViewById(R.id.toolbar);
//          setSupportActionBar(toolbar);
//
//          FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
//          fab.setOnClickListener(new View.OnClickListener()
{
//              @Override
//              public void onClick(View view) {
//                  Snackbar.make(view, "Replace with your
own action", Snackbar.LENGTH_LONG)
//                          .setAction("Action",
null).show();
//              }
//          });
//
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        activateToolbarWithHomeEnabled();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action
bar if it is present.
        getMenuInflater().inflate(R.menu.menu_search,
menu);

        return true;
    }

}
```

This concludes the change need to be made while on <u>Lecture 74: Continue Working On The SearchActivity Class</u>

Things to do when creating the project Lecture 77: Create the RecyclerItemClickListener Class

Additional Method for the RecyclerItemClickListener Class

When you finish going through the lecture on creating the **RecyclerItemClickListener** Class on the lecture as of the time of this documentation. You need to add this empty method in before the closing bracket of the entire class.

```
@Override
public void onRequestDisallowInterceptTouchEvent(boolean
disallowIntercept) {

}
```

Your RecyclerItemClickListener class should now look like this in then.

```
public class RecyclerItemClickListener implements
RecyclerView.OnItemTouchListener {

    public static interface OnItemClickListener
    {
        public void onItemClick(View view, int position);
        public void onItemLongClick(View view, int
position);
    }

    private OnItemClickListener mListener;
    private GestureDetector mGestureDetector;

    public RecyclerItemClickListener(Context context, final
RecyclerView recyclerView, OnItemClickListener listener) {
        mListener = listener;
        mGestureDetector = new GestureDetector(context, new
GestureDetector.SimpleOnGestureListener() {
            public boolean onSingleTapUp(MotionEvent e) {
                return true;
            }

            public void onLongPress(MotionEvent e) {
                View childView =
recyclerView.findChildViewUnder(e.getX(),e.getY());
                if(childView != null && mListener != null)
{
                    mListener.onItemLongClick(childView,
recyclerView.getChildPosition(childView));
                }
            }
        });
    }

    public boolean onInterceptTouchEvent(RecyclerView view,
MotionEvent e) {
        View childView = view.findChildViewUnder(e.getX(),
e.getY());
        if(childView != null && mListener != null &&
mGestureDetector.onTouchEvent(e)) {

mListener.onItemClick(childView,view.getChildPosition(child
View));
        }
        return false;
    }

    public void onTouchEvent(RecyclerView view, MotionEvent
motionEvent) {
    }

    @Override
    public void
onRequestDisallowInterceptTouchEvent(boolean
disallowIntercept) {

    }
}
```

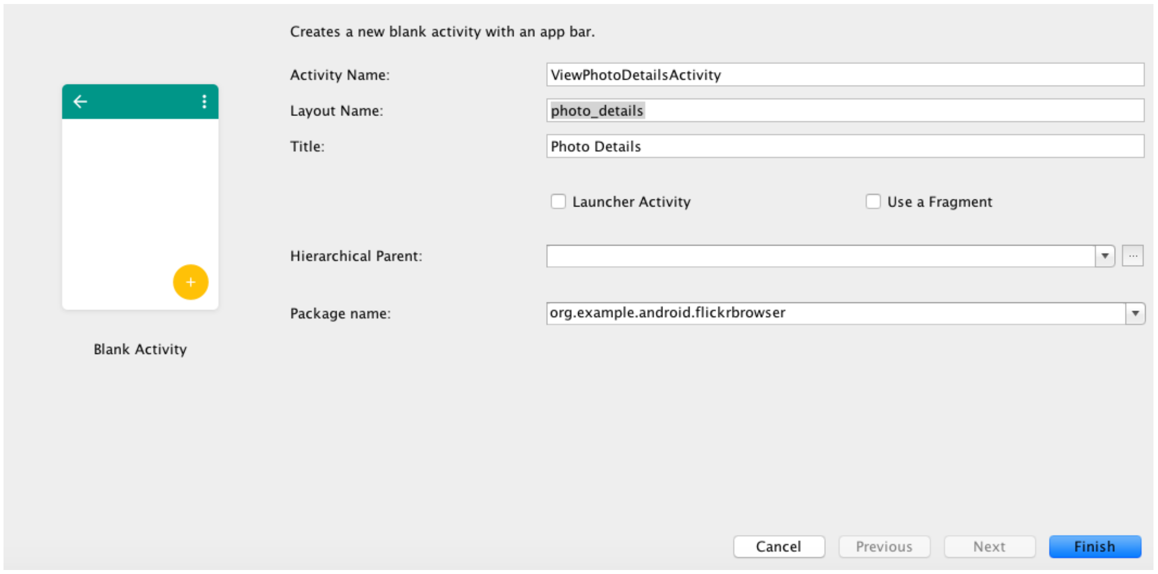
This concludes the change need to be made while on Lecture 77: Create the RecyclerItemClickListener Class

Things to do when creating the project Lecture 78: Implement the ViewPhotoDetailsActivity class

Creating The ViewPhotoDetailsActivity.java

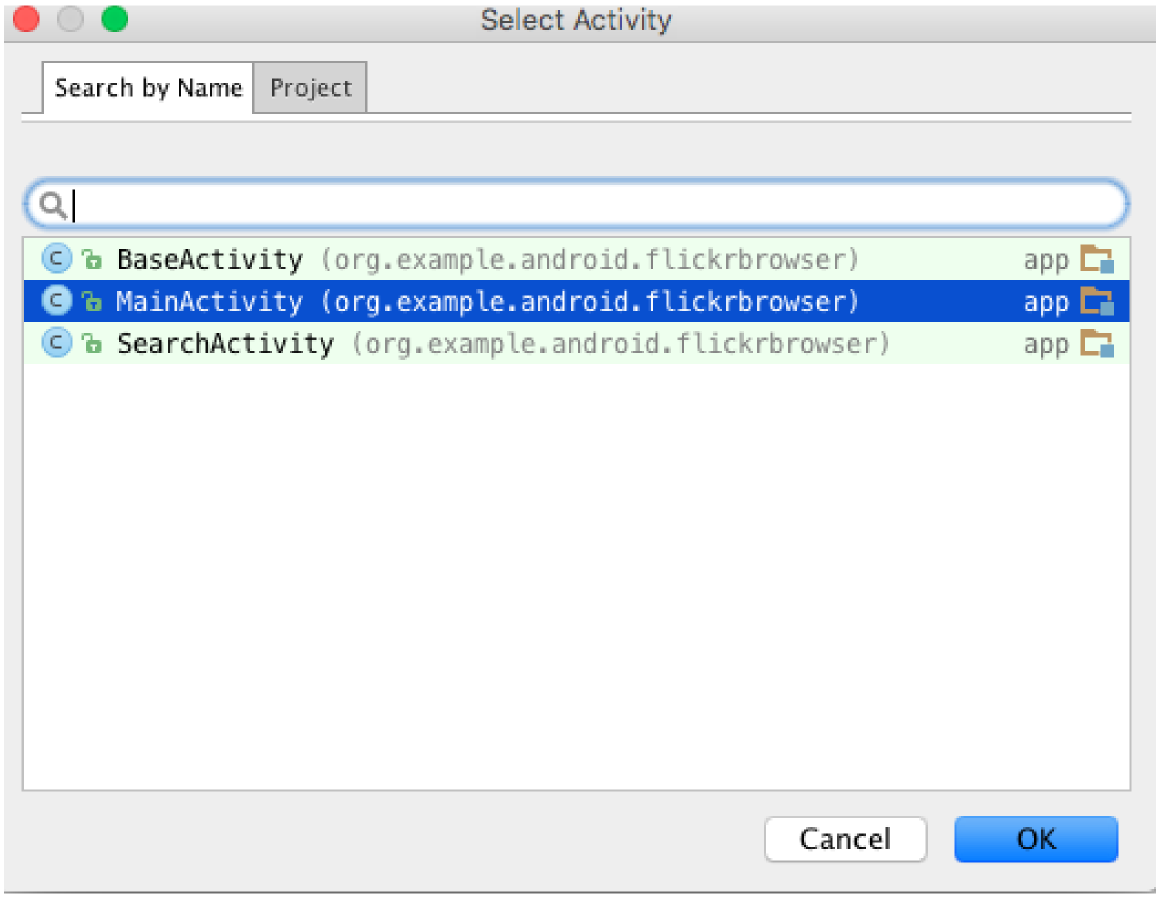
When creating the SearchActivity.java just proceed with the creation despite no option to add the **menu_photo_details.xml** file. Then do the following edits

When adding the Hierarchical Parent make sure to click the browse button and select MainActivity.

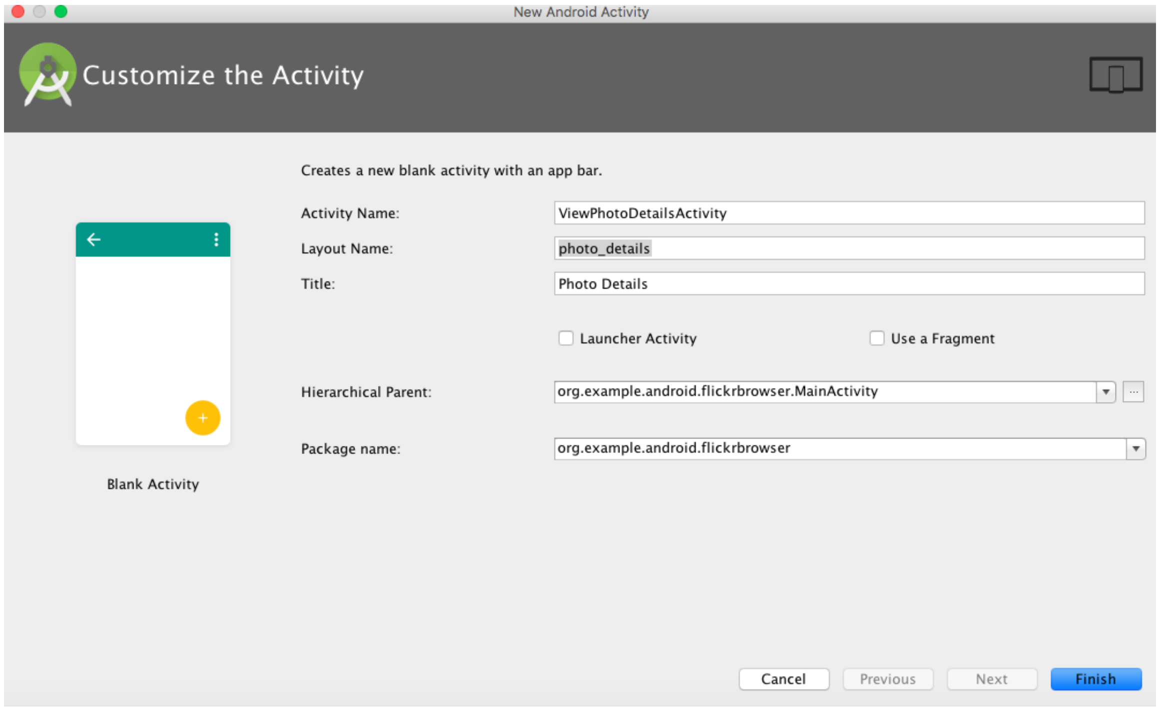


Creates a new blank activity with an app bar.

Activity Name: ViewPhotoDetailsActivity
Layout Name: photo_details
Title: Photo Details
Launcher Activity [ ] Use a Fragment [ ]
Hierarchical Parent:
Package name: org.example.android.flickrbrowser


Select Activity

BaseActivity (org.example.android.flickrbrowser) app
MainActivity (org.example.android.flickrbrowser) app
SearchActivity (org.example.android.flickrbrowser) app


New Android Activity

Customize the Activity

Creates a new blank activity with an app bar.

Activity Name: ViewPhotoDetailsActivity
Layout Name: photo_details
Title: Photo Details

☐ Launcher Activity ☐ Use a Fragment

Hierarchical Parent: org.example.android.flickrbrowser.MainActivity

Package name: org.example.android.flickrbrowser

Congrats you have now created the ViewPhotoDetailsActivity.

Commenting out the Floating Action Button and Creating Blank Activity.

For every Blank Activity you create throughout the entire project you need to comment out the following lines of code.

Example if you have a created a Blank Activity called **ViewPhotoDetailsActivity** you need to comment out the following lines in the SearchActivity.java file. Just select the lines below and press CMD+/ or CTRL+/ if your in a Windows. Java files are stored in the **App>Src>Main>Java>Packagename** of the project.

```
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolbar);

FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Snackbar.make(view, "Replace with your own action",
Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();
    }
});
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
```

After commenting the following lines of code the current inside of your MainActivity.java should now look like the following below.

```
public class ViewPhotoDetailsActivity extends
AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.photo_details);
//          Toolbar toolbar = (Toolbar)
findViewById(R.id.toolbar);
//          setSupportActionBar(toolbar);
//
//          FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
//          fab.setOnClickListener(new View.OnClickListener()
{
//              @Override
//              public void onClick(View view) {
```

```
//                 Snackbar.make(view, "Replace with your
own action", Snackbar.LENGTH_LONG)
//                         .setAction("Action",
null).show();
//             }
//         });
//
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    }

}
```

Then you need to open up the activity's corresponding activity xml file which in this example is **photo_details.xml** and comment out these lines of code. Xml layout files are always stored in **App>Src>Main>Res>layout** of the project.

```
<android.support.design.widget.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/Theme.Flickr.AppBarOverlay">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/Theme.Flickr.PopupOverlay"/>

</android.support.design.widget.AppBarLayout>

<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_margin="@dimen/fab_margin"
    android:src="@android:drawable/ic_dialog_email"/>
```

You can comment them by selecting the lines of code and press CMD+/ or CTRL+/ for windows.

This is what your **photo_details.xml** contents should look like after commenting.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
```

```
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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"

tools:context="org.example.android.flickrbrowser.ViewPhotoD
etailsActivity">

    <!--<android.support.design.widget.AppBarLayout-->
        <!--android:layout_width="match_parent"-->
        <!--android:layout_height="wrap_content"-->
        <!--
android:theme="@style/Theme.Flickr.AppBarOverlay">-->

        <!--<android.support.v7.widget.Toolbar-->
            <!--android:id="@+id/toolbar"-->
            <!--android:layout_width="match_parent"-->
            <!--
android:layout_height="?attr/actionBarSize"-->
            <!--android:background="?attr/colorPrimary"-->
            <!--
app:popupTheme="@style/Theme.Flickr.PopupOverlay"/>-->

    <!--</android.support.design.widget.AppBarLayout>-->

    <include layout="@layout/content_view_photo_details"/>

    <!--
<android.support.design.widget.FloatingActionButton-->
        <!--android:id="@+id/fab"-->
        <!--android:layout_width="wrap_content"-->
        <!--android:layout_height="wrap_content"-->
        <!--android:layout_gravity="bottom|end"-->
        <!--android:layout_margin="@dimen/fab_margin"-->
        <!--
android:src="@android:drawable/ic_dialog_email"/>-->

</android.support.design.widget.CoordinatorLayout>
```

Apply this every time you create a new Blank Activity. You must comment out the following lines of code in both the java file and the corresponding activity xml file of it

This concludes the change need to be made while on
<u>Lecture 78: Implement the ViewPhotoDetailsActivity class</u>

Things to do when creating the project Lecture 79: Setup The Layout XML For The ViewPhotoDetailsActivity Class

Editing The content_view_photos_details.xml

In this lecture instead of editing the photo_details.xml you will be editing the content_view_photo_details.xml file.

Please do not remove the following lines that highlighted in red while editing it.

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

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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"

android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"

app:layout_behavior="@string/appbar_scrolling_view_behavior
"

tools:context="org.example.android.flickrbrowser.ViewPhotoD
etailsActivity"
    tools:showIn="@layout/photo_details">

</RelativeLayout>
```

You can edit the other parts but don't edit the ones highlighted for this lecture in the content_view_photo_details.xml file.

This concludes the change need to be made while on Lecture 79: Setup The Layout XML For The ViewPhotoDetailsActivity Class