Android Reference

By John Bentley

IDES

Eclipse (Deprecated)

In Eclipse, press Ctrl + Shift + O to create missing “imports”.

To print out debug information write the the LogCat with android.util.Log

```
import android.util.Log;

private void getLocales() {
          final String LOG_TAG ="au.com.softmake.LOG";

          int i = 1;
          String logMessage = "";
          for (Locale locale : Locale.getAvailableLocales()) {
                    logMessage = String.format("%03d %8s : %s%n", i++, locale.toString(),
locale.getDisplayName());
                    Log.i(LOG_TAG, logMessage);
          }
}
```

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages) *https://developer.android.com/reference/android/util/Log.html*

In practice use logging functions from a Standard Applciation Library that wraps your logging tag.

```
au.com.softmake.standardapplibrary.Log.i("onServiceConnected Start", this);
```

C:\Users\John\Documents\Sda\Code\Android\Libraries\StandardAppLibrary\src\au\com\softmake\standardapplibrary\Log.java

To import Android SDK samples into eclipse:

- Open an existing workspace.
- File > New > Other ...
- Android > Android Sample Project

(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/) *http://stackoverflow.com/questions/12209661/unable-to-import-sample-project-from-sdk*

To import existing projects into a workspace for which you are having trouble importing.

- Copy files to a disk location outside of your workspace root.
- File > Import > Android > Existing Code Into Workspace. Next.
- Navigate to the directory of the files to import.
- Tick "Copy projects into workspace".
- Finish.

If you can't build Eclipse try the following: Check errors in the menu layout.

Android Studio

Setup

Setup Environment

See:

(Bentley 2021, "Android And Java SDA", C:\Users\John\Documents\CustomData\AppDataUserSaved\Brainforest\DevAndComputer\Android\AndroidAndJavaSda.pdb), Manual Updgrades

Setup App

See:

(Bentley 2021, "Android And Java SDA", C:\Users\John\Documents\CustomData\AppDataUserSaved\Brainforest\DevAndComputer\Android\AndroidAndJavaSda.pdb), Setup Android App (creation, building, and running).

Libraries

Run app as Java

Sometimes you just want to run a small bit of Java code in your app, to test it. You can do this as follows:

- Create a class in your app with a main method (ensure you get the signature right);

```
package au.com.softmake.mythirdapp;

import au.com.softmake.mythirdapp.recyclerviewdemo.Country;

public class QuickTest {
    public static void main(String[] args) {
        Country country = new Country("Afghanistan", "AF", 69.1761f, 34.5228f);
        System.out.println(country.toString());
    }
}
```

- Run > Edit configurations ...
  - Application > [+] ...
  - Name: RunAppAsJava
  - Main class: au.com.softmake.mythirdapp.QuickTest
  - Use classpath of module: app
- Select this configuration in the run dropdown box. Click on the green triangle (or Shift + F10) to run it.

Debug

To print out debug information write to the LogCat with android.util.Log but use one's standard library as a log tag

```
// Main app
import android.util.Log;

...
Log.i( au.com.softmake.standardapplibrary.java.SalGlobals.LOG_TAG, "Hello");
...

// au.com.softmake.standardapplibrary.java.SalGlobals.LOG_TAG
public class SalGlobals {
    public final static String LOG_TAG = "au.com.softmake.LOG";
}
```

```
// LogCat
06-24 03:44:53.942    2105-2105/au.com.softmake.myfirstapp02 I/au.com.softmake.LOG﹕ Hello
```

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages) *https://developer.android.com/reference/android/util/Log.html*

You could also use System.out to print to LogCat (not the Console) but you lose out on the advantages of log tagging. Ensure you debug, not run, to get this output. Ensure you output a newline

```
// In App
System.out.println("Nice"); // Ok because new line
System.out.printf("%s", person.toString()); // Not Ok. No newline
System.out.printf("%s\n", person.toString()); // Ok. Has newline.

// LogCat
06-24 03:44:53.942    2105-2105/au.com.softmake.myfirstapp02 I/System.out﹕ Nice
```

You can log to the Console with an Action Breakpoint (Shift + Click Gutter)

(JetBrains 2014, "IntelliJ IDEA > Getting Started", https://www.jetbrains.com/help/idea/getting-started.html) *https://www.jetbrains.com/help/idea/debugging-your-first-java-application.html*

If your are not able to stop at breakpoints try

- Run > Attach debugger to Android process ...

(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/) "android debugger does not stop at breakpoints", *https://stackoverflow.com/questions/5293415/android-debugger-does-not-stop-at-breakpoints/48615795#48615795*

JavaDocs

Troubleshooting Tools > Generate JavaDoc. If your Java docs won't generate due to "package android.os does not exist" errors then add a -bootclasspath command line argument that points to the android.jar.

```
// Errors
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\standardapplibrary\s
rc\main\java\au\com\softmake\standardapplibrary\android\Files.java:11: error: package
android.os does not exist
import android.os.Environment

// Fix
// Tools > Generate JavaDoc ...
// Other command line arguments:
-bootclasspath "C:\Program Files\Android\sdk\platforms\android-19\android.jar"
```

(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/) *http://stackoverflow.com/questions/5200234/javadoc-in-eclipse-failing-to-recognize-packages*

Rename Project

Rename Project (Needs refinement).

- In Android Studio. Click on Project Root. Refactor > Raname to "MyNewApp".
- In Android Studio. Open your package root "com.example.myoldapp" and Refactor > Rename to "mynewapp" (to effect a new package root of "com.example.mynewapp").
- /res/string/ change the app_name string to "My New App".
- In Android Studio. Click on Project Root. Ctrl + Shift + R for Replace in Path.
- Text to Find "MyOldApp" to "MyNewApp".
- Directory, Recursively.
- Preserve case: ticked
- "Do you want to replace this occurrence?" All files.
- Do not reload project when prompted. Instead Exit Android Studio.
- In Windows Explorer (or your OS's file system) rename your project directory "MyOldApp" to "MyNewApp".
- Open "MyNewApp" in Android Studio. Quick Start > Open Project ... Surf to newly named directory.
- Verify successful by loading into an Emulator.

Libraries

Check the latest version

For support libraries (e.g. Android support library; Contraint Layout Library), check the latest Google Maven Repository:

- What the Google Maven Repository supports:
  (Google, n.d., "Android Studio User Guide",
  https://developer.android.com/studio/intro) "Google's Maven Repository",
  https://developer.android.com/studio/build/dependencies.html#google-maven
- The latest version, see the directory of the Google Maven Repository:
  https://dl.google.com/dl/android/maven2/index.html

Jetpack

https://developer.android.com/jetpack

AndroidX Library

Use AndroidX over the deprecated "Android Support Library".

> All new Support Library development will occur in the AndroidX library.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html) , "AndroidX Overview", https://developer.android.com/jetpack/androidx/

Migrate from "Android Support Library" to AndroidX:

- {Android Studio} > Refactor > Migrate to AndroidX.

  (Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html), AndroidX, Migrating to AndroidX, https://developer.android.com/jetpack/androidx/migrate#migrate

- Verify `gradle.properties`

```
android.enableJetifier=true
android.useAndroidX=true
```

  (Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html), AndroidX, Migrating to AndroidX, https://developer.android.com/jetpack/androidx/migrate#migrate

- Verify build.gradle (Module: app) support library artifacts changed to android. Use the listed "artificate mappings" (see rule source); and "Check the latest version" above.

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

implementation 'androidx.appcompat:appcompat:1.0.2'
implementation "androidx.preference:preference:1.1.0"
```

  (Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html), AndroidX, Migrating to AndroidX, https://developer.android.com/jetpack/androidx/migrate#artifact_mappings

- In res/layout/ files Search and replace (Edit > Find > Replace in Path) the following common strings (and other class mappings, use the html link or CSV download in sources below):

```
androidx.constraintlayout.ConstraintLayout to
androidx.constraintlayout.widget.ConstraintLayout

android.support.design.widget.CoordinatorLayout to
androidx.coordinatorlayout.widget.CoordinatorLayout

android.support.design.widget.AppBarLayout to
com.google.android.material.appbar.AppBarLayout

android.support.v7.widget.Toolbar to
androidx.appcompat.widget.Toolbar

android.support.v7.widget.RecyclerView to
androidx.recyclerview.widget.RecyclerView

android.support.v7.widget.CardView to
androidx.cardview.widget.CardView

android.support.design.widget.FloatingActionButton to
com.google.android.material.floatingactionbutton.FloatingActionButton
```

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html), AndroidX, Migrating to AndroidX, Class Mappings, https://developer.android.com/jetpack/androidx/migrate#class_mappings & https://developer.android.com/topic/libraries/support-library/downloads/androidx-class-mapping.csv
(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/), "Error inflating class androidx.constraintlayout.widget.ConstraintLayout", https://stackoverflow.com/questions/50783810/error-inflating-class-androidx-constraintlayout-widget-constraintlayout

- Preferences

```
// build.gradle (Module:app)
dependencies {
...
implementation 'androidx.appcompat:appcompat:1.1.0'
implementation "androidx.preference:preference:1.1.0"
...
}
```

(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/), https://stackoverflow.com/a/58886654/872154

Custom Libraries

The Authorative version of "Custom Libraries" is now at:

(Bentley 2019, "AndroidJavaGradle-CustomLibrarySetup.Docx", AndroidJavaGradle-CustomLibrarySetup.docx) AndroidJavaGradle-CustomLibrarySetup.docx

An earlier version of "Custom Libraries" section online at Stackoverflow, *How do we reference custom Android and Java Libraries living outside the directory of our Main Android Project?*, https://stackoverflow.com/questions/49268039/how-do-we-reference-custom-android-and-java-libraries-living-outside-the-directo/49268123#49268123 . This version still has public utility but my private thoughts at AndroidJavaGradle-CustomLibrarySetup.docx are the master.

(Deprecated) Android Support Library

Basics

For support of newer android features in previous android versions use the Android Support Library.

(Google n.d., "Support Library | Android Developers", Accessed 2021-02-13. https://developer.android.com/topic/libraries/support-library) *https://developer.android.com/topic/libraries/support-library*
https://developer.android.com/topic/libraries/support-library/setup

To use the Android Support Library:

- Download the Android Support Library via the Android SDK Manageer > Extras.
- Decide on the features you want backward compatibility for.
- Identify whether the Android Support Library(s) you want come with resources or not.
- Add the Android Support Library(s) to your project.
- Adjust your code to utilize the Android Support Library.

Add Android Support Library(s) to your project depends on whether the support library comes with resources or not. See Adding libraries without resources; and Adding libraries with resources.

(Google n.d., "Support Library | Android Developers", Accessed 2021-02-13. https://developer.android.com/topic/libraries/support-library) *https://developer.android.com/topic/libraries/support-library/setup#add-library*

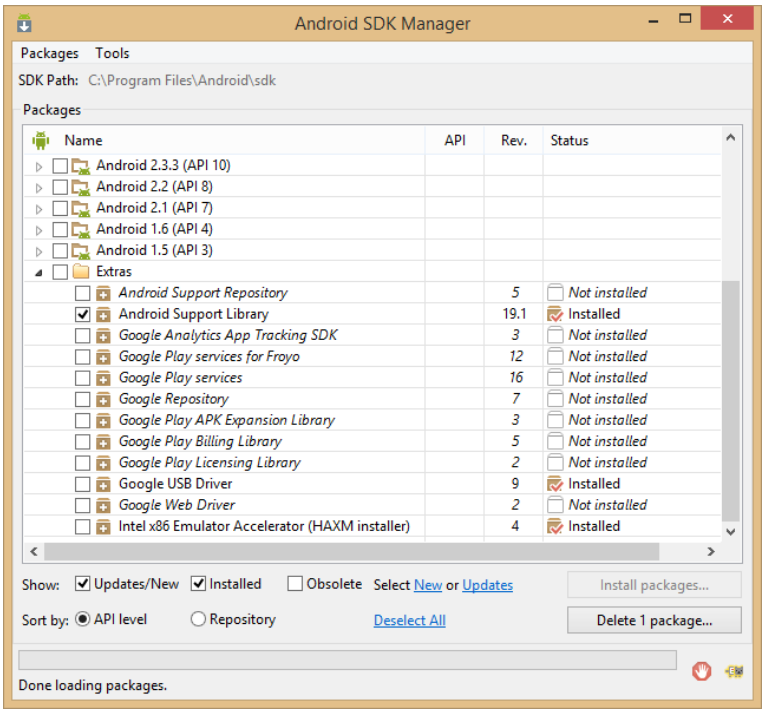
Checklist for Debugging Support Library Reference Problems

android-support-v4.jar (a library without resources)

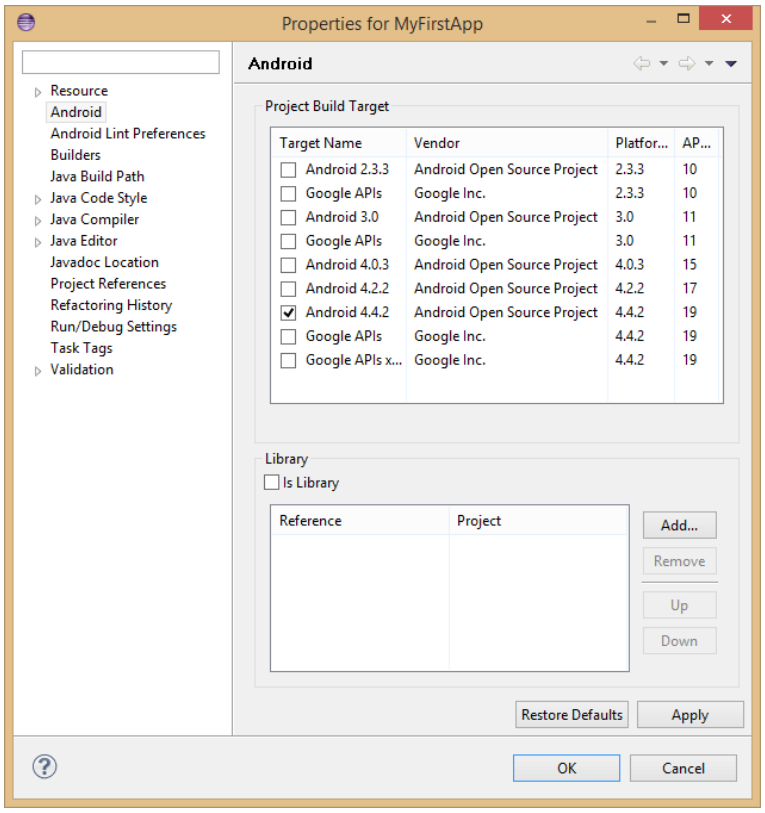
- Ensure `android-support-v4.jar` is at `MyApp/libs/android-support-v4.jar`. If not copy from "C:\Program

Files\Android\sdk\extras\android\support\v4\android-support-v4.jar" by dragging this to MyApp/libs/ *within Eclsipse*.

- If missing from \extras\android\support\v4\ then download from the Android SDK Manager.

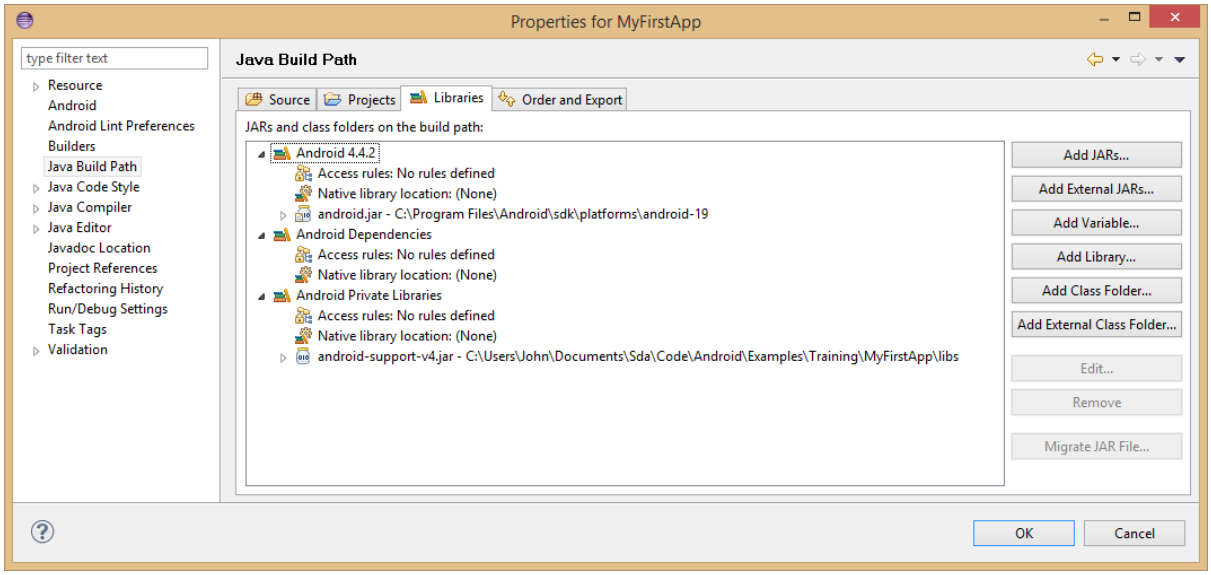


- Eclipse MyApp/libs/android-support-v4.jar > Right click > Build Path > Add to Build Path ...
- Eclipse > MyApp RightClick > Properties > Android. Check there is no Library Reference.

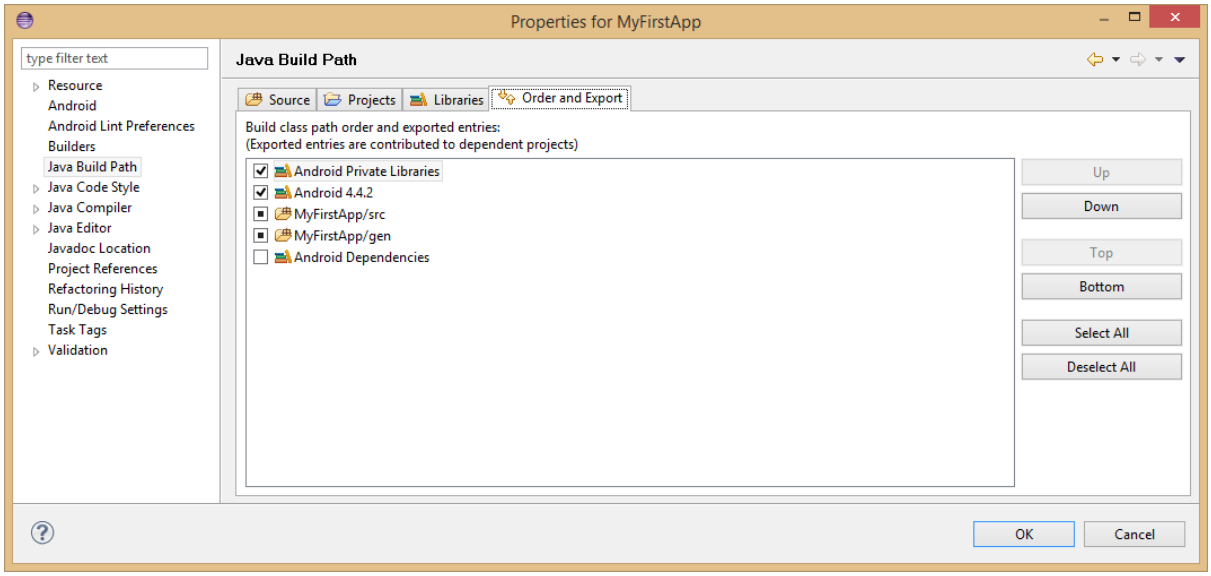


- Eclipse > MyApp RightClick > Properties > Java Build Bath.
  Projects Tab = None.

  Libraries Tab, as follows



- Order and Export Tab, as follows



- Eclipse > Project > Build ...
- You may have to comment out "showAsAction" settings in .xml files in order for the project to build first (to sort references out).

App Fundamentals

.apk

The android SDK compiles your Java code into an Android package, a single file with an .apk extension.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html) *http://developer.android.com/guide/components/fundamentals.html*

Java, Android, and Android Studio

When using Java with Android Studio use the OpenJDK implementation of Java. This is the embeded version:

- File > Project Structure ... > SDK Location > JDK Location: "Embedded JDK C:\Program Files\Android\Android Studio\jre"

This is what is officially recommended.

(Google, n.d., "Android Studio User Guide", https://developer.android.com/studio/intro) Android Studio Users Guide, Meet Android Studio, Configure the IDE, Set the JDK Version, https://developer.android.com/studio/intro/studio-config#jdk

The relationship between Java, Android, and Android Studio is as follows:

- Among the several Java implementations (in additions to Oracle's, OpenJDK, and (the deprecated) Apache Harmony) is Android's (Google's). This java implementation lives inside `android.jar` (or the source at C:\Users\John\AppData\Local\Android\sdk\sources\android-23\java\net\URL.java).
- Historically Android's implementation was based on Apache Harmony. In the future it will be based on Open JDK.
- The separate Oracle JDK you download (and you could just as well use Open JDK) is merely used, in an Android development context, in order to use the tools that come with it (the javac compiler for example). Those java tools are required by the Android Studio IDE. No such tools come with Android's java implementation, it just implements a Java API library, which explains why you need a third party JDK.
- In an Android context `java.*` classes will always come from (and be compiled with respect to) `android.jar`. So, for example, the right reference documentation for `java.net.HttpURLConnection` is http://developer.android.com/reference/java/net/HttpURLConnection.html rather than https://docs.oracle.com/javase/7/docs/api/java/net/HttpURLConnection.html. For although the public java API (the methods, fields, etc you can use) for each implementation will generally be the same there might be small differences (over and above internal implementation details).

Based on comments by Jake Wharton, 2016-04-01, *https://www.reddit.com/r/androiddev/comments/4cp62q/is_the_android_developers_doc_wrong_for_a_url/d1l9pje (see my child comment there).*

App Components

Four types of app components:

- Activities: a single screen with a UI.
- Services: runs in the background to perform long-running operation or to perform work for remote processes.
- Broadcast receivers: responds to system wide announcenment, originating from the system itself (e.g. screen off) or an app.
- Content providers: manages app data, whether private to the app or shared with other apps.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/components/fundamentals.html

Any app can start another app's component. For Activities, Services, and Broadcast Receivers this is done with an `intent`. Content providers are handled by Content resolvers, rather than intents.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/components/fundamentals.html

Chips, Systems, CPUs, and Instruction Sets

System on a Chip, or "System On Chip":

- Atom. By Intel.
- Snapdragon. By Qualcomm
- Exynos. By Samsung.
- Apple A9. By Apple.

Instruction sets:

- ARM. E.g. ARMv7-A (32 bit); ARMv8-A (64 Bit).

  In 2013, 10 billion were produced[32] and "ARM-based chips are found in nearly 60 percent of the world's mobile devices.

  https://en.wikipedia.org/wiki/ARM_architecture#History

Example device architecture:

Device	System On Chip	CPU	Instruction Set	32bit V 64bit
Samsung Galaxy S5	Snapdragon 801 (by Qualcomm)	ARM Cortex-A15 MPCore	ARMv7-A	32bit
Samsung Galaxy S7	Exynos 8890 (by Samsung)	Exynos M1 "Mongoose"+ Cortex-A53	ARMv8-A	64bit.
Apple iPhone 6S	A9 (by Apple)	1.85 GHz dual-core 64-bit ARMv8-A[2][3] "Twister"	ARMv8-A	64bit

System Image Choice

Which "System Image(s)" to choose in Android SDK Manager? Chose:

- "**Google APIs Intel x86 Atom_64 System Image**" or, for earlier APIs, "Google APIs Intel x86 Atom System Image". This supports API functionality such as Cloud Messaging, Maps, Firebase Realtime database, etc. Also in Android Studios's Virtual Device Manager this is promoted as the image to download
- (Deprecated) "**Intel x86 Atom_64 System Image**" or, for earlier APIs, "Intel x86 Atom System Image"; and

Only Intel x86 images support HAXM emulation (the fast emulation). And we are moving toward 64 bit architectures.

https://software.intel.com/en-us/android/articles/how-to-develop-and-evaluate-64-bit-android-apps-on-intel-x86-platforms

AndroidManifest.xml

Intro

The Android Manifest "presents essential information about your app to the Android System". This includes (amongst other things) permissions, a description of your apps components (activities, services, broadcast receivers, and content providers), linked libraries, and Android API level support.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/topics/manifest/manifest-intro.html

Every application must have a `AndroidManifest.xml` that lives in the root of the project directory.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/components/fundamentals.html#Manifest

It must be named "`AndroidManifest.xml`".

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/topics/manifest/manifest-intro.html

All app components must be declared in the Manifest.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/components/fundamentals.html#DeclaringComponents

Some of the `AndroidManifest.xml` entries are better automatically managed by Gradle. If they are managed by Gradle, you leave them out of AndroidManifest.xml (gradle will insert these into AndroidManifest.xml at build time).

AndroidManifest.xml File conventions

AndroidManifest.xml has conventions particular to it as follows ...

If an element contains anything at all it contains other elements, not character data.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/topics/manifest/manifest-intro.html#filec

Manifest elements reference a Java class through the name attribute. When this is done the Java class name must include it's full package name. This can be done explicitly or through a short hand. To use the shorthand: prefix the class name with a dot "."; and declare the package in the "package" attribute of the manifest.

```
<manifest package="com.example.project". . . >
    <application . . . >
        <!-- Explicit package name -->
        <service android:name="com.example.project.SecretService" . . . >
            . . .
        </service>
        <!-- Implicit package name using the shorthand technique. Dot "." prefix. -->
        <service android:name=".CoolService" . . . >
            . . .
        </service>
        . . .
```

```
    </application>
</manifest>
```

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/topics/manifest/manifest-intro.html#filec

If an element's attribute can have multiple values this is expressed by repeating the element/attribute rather than have a space separated list of values in the one attribute.

```
<intent-filter . . . >
    <action android:name="android.intent.action.EDIT" />
    <action android:name="android.intent.action.INSERT" />
    <action android:name="android.intent.action.DELETE" />
    . . .
</intent-filter>
```

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/topics/manifest/manifest-intro.html#filec

If an attribute in AndroidManifest.xml needs to reference a resource then the following format is used:

`@[package]type/name`

`package` is omitted if the resource is in the same package.

```
<application android:icon="@drawable/ic_launcher" ...
```

Attribute values that reference values from a theme use "?" in palce of "@". That is, the following format is in play: `?[package]type/name`

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/topics/manifest/manifest-intro.html#filec

When an attribute value is a string, and the string conatins characters that must be escaped, use double backslashes "\\" to do the escaping. E.g. "\\n" for newline or "\\uxxxx" for a unicode character.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/topics/manifest/manifest-intro.html#filec

Particular usages worth highlighting

Some elements have `icon`, `label` and `description` attributes. When set any children elements automatically inherit the values of the parent attribute, unless overridden. E.g. If icon and label are set for the application, all the activities inherit this, unless overridden.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/topics/manifest/manifest-intro.html#filec

For permission usage, both custom and framework, see Custom Permssions and Enforce permissions.

Supporting Different Devices

Different device features

Declare application feature requirements that a device **must** have with `uses-feature` and `required="true"` in AndroidManifest.xml.

```
<manifest ... >
    <uses-feature android:name="android.hardware.camera.any"
                  android:required="true" />
    ...
</manifest>

<!-- This results in a user being unable to install your app from the Google Play Store on a
device that lacks a camera -->
```

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/components/fundamentals.html#DeclaringRequirements

For application features that are optional, set `required="false"` and gracefully degrade in code.

```
<manifest ... >
    <uses-feature android:name="android.hardware.sensor.compass"
        android:required="false" />

PackageManager pm = getPackageManager();
if (!pm.hasSystemFeature(PackageManager.FEATURE_SENSOR_COMPASS)) {
    // This device does not have a compass, turn off the compass feature
    disableCompassFeature();
}
```

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/practices/compatibility.html

Device feature reference:

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/topics/manifest/uses-feature-element.html#features-reference

Different android versions

Check the current platform popularity:

http://developer.android.com/about/dashboards/index.html

Platform Version (e.g. Android 4.1.2), API level (e.g. 16), and version code (e.g. JEALLY_BEAN) reference:

http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels

You should always set the `android:targetSdkVersion` as high as possible (the latest android version available) and test your app on the corresponding platform version.

```
<!-- In AndroidManifest.xml -->

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="19" />
```

The minSdkVersion attribute declares the minimum version with which your app is compatible and the targetSdkVersion attribute declares the highest version on which you've optimized your app.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/training/basics/firstapp/running-app.html
(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/training/basics/supporting-devices/platforms.html

Graceful degradation when lower android versions can't quite handle your functionality by using Build.VERSION_CODES or API level integers.

```
if (Build.VERSION.SDK_INT < Build.VERSION_CODES.HONEYCOMB) {
    // Running on something older than API level 11, so disable
    // the drag/drop features that use ClipboardManager APIs
    disableDragAndDrop();
}

// OR

if (Build.VERSION.SDK_INT < 11) {
...
```

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/practices/compatibility.html#Versions

Screen support

Android categorizes device screens using two general properties: size and density.

There are four generalized sizes: small, normal, large, xlarge

And the following generalized densities: low (ldpi), medium (mdpi), high (hdpi), extra high (xhdpi), xxhdpi (extra-extra high).

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html) "Screen compatibility overview", https://developer.android.com/guide/practices/screens_support

Layout and Screen sizes

Layout and screen size folder structure

```
MyProject/
    res/
        layout/             # default (portrait)
            main.xml
        layout-land/        # landscape
            main.xml
        layout-large/       # large (portrait)
            main.xml
        layout-large-land/  # large landscape
            main.xml
```

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html) "Screen compatibility overview", https://developer.android.com/guide/practices/screens_support

Reference your layout in code in a screen independent manner ...

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

```
}
```

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html) "Screen compatibility overview", https://developer.android.com/guide/practices/screens_support

Google Play filter for screen size

You can exclude some devices from installing your app from the Google Play Store using `<supports-screens>` in AndroidManifest.xml

http://developer.android.com/google/play/filters.html

Bitmaps

You should always provide bitmap resources that are properly scaled to each of the generalized density buckets: low, medium, high and extra-high density (there is no mention of extra-extra high in the documentation).

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html) "Screen compatibility overview", https://developer.android.com/guide/practices/screens_support

To generate these images, you should start with your raw resource in vector format and generate the images for each density using the following size scale:

- xhdpi: 2.0
- hdpi: 1.5
- mdpi: 1.0 (baseline)
- ldpi: 0.75

Bitmap Folder Structure.

```
MyProject/
    res/
        drawable-xhdpi/
            awesomeimage.png
        drawable-hdpi/
            awesomeimage.png
        drawable-mdpi/
            awesomeimage.png
        drawable-ldpi/
            awesomeimage.png
```

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html) "Screen compatibility overview", https://developer.android.com/guide/practices/screens_support

Low-density (ldpi) resources aren't always necessary. When you provide hdpi assets, the system scales them down by one half to properly fit ldpi screens.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html) "Screen compatibility overview", https://developer.android.com/guide/practices/screens_support

String Internationalization

Internationalise strings like this. Note that region codes are prefixed with an "r".

```
MyProject/
    res/
       values/
           strings.xml
```

```
       values-es/
           strings.xml
       values-fr/
           strings.xml
       values-fr-rBE/
           strings.xml
```

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html), "Support different languages and cultures", https://developer.android.com/training/basics/supporting-devices/languages.html

A locale consists of a language code followed by an optional region code.

- Language code. Two (preferred) or three lowercase letters. From ISO 639-2 (but ISO 639-1 two letter codes preferred). http://www.loc.gov/standards/iso639-2/php/code_list.php
- Region code. This generally distinguishes regional differences within one language. Two (preferred) or three UPPERCASE letters. From ISO 3166 or UN M.49. http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

```
      ar : Arabic
   ar_MA : Arabic (Morocco)
      ca : Catalan
   ca_AD : Catalan (Andorra)
      de : German
   de_AT : German (Austria)
   de_BE : German (Belgium)
   de_CH : German (Switzerland)
   de_DE : German (Germany)
   de_LI : German (Liechtenstein)
   de_LU : German (Luxembourg)
      en : English
  en_150 : English (Europe)
   en_AU : English (Australia)
   en_GB : English (United Kingdom)
      eo : Esperanto
      es : Spanish
  es_419 : Spanish (Latin America)
   es_ES : Spanish (Spain)
      fr : French
   fr_BE : French (Belgium)
   fr_CH : French (Switzerland)
   fr_FR : French (France)
      zh : Chinese
 zh_HANS : Chinese (HANS)
zh_HANS_CN : Chinese (China,CN)
zh_HANS_HK : Chinese (Hong Kong,HK)
zh_HANS_MO : Chinese (Macau,MO)
zh_HANS_SG : Chinese (Singapore,SG)
```

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages)
http://developer.android.com/reference/java/util/Locale.html

To display a list of the locales supported by a particular Android version open up an emulator and open the "Custom Locale" app, or run the following from your Android App, debug, and examine the LogCat.

```
private void getLocales() {
           final String LOG_TAG ="au.com.softmake.LOG";

           int i = 1;
           String logMessage = "";
           for (Locale locale : Locale.getAvailableLocales()) {
                       logMessage = String.format("%03d %8s : %s%n", i++, locale.toString(),
locale.getDisplayName());
                       Log.i(LOG_TAG, logMessage);
           }
}
```

In Logcat filter by your LOG_TAG value. You could then copy the logcat to the clipboard, then to a text editor.

To set the locale for testing purposes use the "Custom Locale" app in the Android Eclipse.

This can also be done through code.

Non-technical filtering on Google Play

Filtering for non-technical reasons (such as geographic locale) is always handled in the Google Play developer console.

Filtering for technical compatibility (such as required hardware components) is always based on information contained within your APK file.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html)
http://developer.android.com/guide/practices/compatibility.html#Versions

Coding Component Basics

Get context

Summary for getting context.

When inside the following Class	Use
Activity	`this`
Fragment	`getActivity()`
Other	`getContext();getApplicationContext();`

When a method parameter requires a context, try `getContext()` first then use `this` if getContext() is not available.

```
public class DatabaseActivity extends Activity {

    NorthwindDBHelper mDBHelper = new NorthwindDBHelper(this);

...

public class ExampleProvider extends ContentProvider {

 ...

    @Override
    public boolean onCreate() {

        mDBHelper = new NorthwindDBHelper(getContext());

...
```

getContext() is made avaible in classes that don't extend Context. In classes that do extend Context then "this" provides the Context and getContext() is not available.

Such cases are thought to be mutually exclusive. That is, there probably does not exist a class that extends Context and makes getContext() available.

(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/) http://stackoverflow.com/questions/6229052/difference-in-context-this-and-getcontext

In a fragment use getActivity(), as an Activity is inherited from a context.

Some methods come for free, being inherited ultimately from the Context object. So when you move methods from an Activity, which work without qualification, into a fragment: you may have to qualify the method with a context.

```
// In Activity (which inherits from Context)
fileOutputStream = openFileOutput(file.getName(), Context.MODE_PRIVATE);

// In Fragment
fileOutputStream = getActivity().openFileOutput(file.getName(), Context.MODE_PRIVATE);
```

(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/) Stackoverflow > The method OpenFileOutput() is undefined ! > Answer by Pontus Gagge , 2010-10-25 > http://stackoverflow.com/a/4015886/872154
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\ShareFilesDemoActivity.java

To get context in a

Return the App Version

Return App Version. In this example into a preference fragment ...

```
// Res\xml\preferences.xml
<Preference
    android:key="@string/preference_app_version"
    android:title="App Version"
    android:summary="@string/overwritten"
    />

// Within your PreferenceActivity class you might have an inner class
// that extends PreferenceFragment. Within that PreferenceFragment
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Preferences must ba an xml resource, not a layout resource.
    addPreferencesFromResource(PREFERENCE_XML_ID);

    setDateTime();
    setAppVersion();
}

private void setAppVersion() {
    Preference appVersionPreference =
findPreference(getResources().getString(R.string.preference_app_version));
    PackageInfo packageInfo = null;
    try {
        packageInfo = getActivity().getPackageManager().getPackageInfo
                (getActivity().getPackageName(),0);
    } catch (PackageManager.NameNotFoundException e) {
        e.printStackTrace();
    }
    appVersionPreference.setSummary(packageInfo.versionName);
}

// If you are trying to get the package manager from an activity do something like
packageInfo = getPackageManager().getPackageInfo(getPackageName(),0);
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MainActivity.java

Saving State and Handling Runtime Changes

[I'm still confused here and therefore the following sections on Saving State and Handling Runtime Changes is confused. Fix]

General

You need to handle runtime configuration changes (like screen rotations) so your loaded objects and data don't disappear from memory. See

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App basics, Handle Configuration Changes, https://developer.android.com/guide/topics/resources/runtime-changes.html

For Activities (only?) the chief technique to handling runtime configuration changes, like a screen rotation, is to use setRetainIntance(true) after a "create" method's super call.

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

```
    setRetainInstance(true);
```

For Fragments (only?) the easiest way to save state and handle runtime configuration changes is to keep an Activity level variable of the fragment.

```
// DialogDemoActivity.Java
public class DialogDemoActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dialog_demo);
        if (savedInstanceState == null) {
            getFragmentManager().beginTransaction().add(R.id.container,
                                                        new PlaceholderFragment())
                    .commit();
        }
    }
    ...

    // An activity level variable in order to keep the AlertDialogList alive between runtime
    // configuration changes (like rotating the screen), and also save state.
    static private MyAlertDialogListFragment mFireMissileDialogFragment;

    /**
     * A placeholder fragment containing a simple view.
     */
    public static class PlaceholderFragment extends Fragment implements View
            .OnClickListener {

        // "Every fragment must have an empty constructor, so it can be instantiated
        // when restoring its activity's state." http://developer.android
        // .com/reference/android/app/Fragment.html
        public PlaceholderFragment() {
        }

        public void createAndShowFireMissileDialogFragment() {
            // Only create if it doesn't already exist.
            if (mFireMissileDialogFragment == null) {
                mFireMissileDialogFragment = new MyAlertDialogListFragment();
            }

            // Android 3.0 (API 11) >= import android.app.Fragment
            // .getFragmentManager();
            // Android 3.0 (API 11) < import android.support.v4.app
            // .DialogFragment.getSupportFragmentManager()
            mFireMissileDialogFragment.show(getFragmentManager(), "FireMissilesTag");
        }
...

// MyAlertDialogListFragment.Java
public class MyAlertDialogListFragment extends DialogFragment {

    final String[] mItems = {"ICBM", "Hellfire", "Propaganda"};
    private boolean[] mItemsCheckedStatus;

    // "Every fragment must have an empty constructor, so it can be instantiated
    // when restoring its activity's state." http://developer.android
    // .com/reference/android/app/Fragment.html
    public MyAlertDialogListFragment() {
    }
...
```

John's testing has shown that the data state (e.g. selections of a DialogAlertList) is retained through orientation changes, quiting out of the app, reentering the app, and force closing the app, without the need for:

* #2 robert.r...@gmail.com's onDestroyView() workaround;
* setRetainInstance(true),
* Using default empty constructors for the fragment(s) (although I include them in my code because the official recommendation is to include them anyway).
* Using onSaveInstanceState() and a corresponding restore of a Bundle.
* Callbacks from the fragment to the activity, to explicitly save bespoke fragment data with the activity.

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MyAlertDialogListFragment.java
(Google, n.d., "Google Issue Tracker", https://issuetracker.google.com/), DialogFragment dismissed on orientation change when setRetainInstance(true) is set (compatibility library), https://code.google.com/p/android/issues/detail?id=17423#c33

Saving Activity State

By default, the system uses the Bundle instance state to save information about each View object in your activity layout (such as the text value entered into an EditText object). So, if your activity instance is destroyed and recreated, the state of the layout is restored to its previous state with no code required by you.

Save and restore state.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) Core Topics, Activities, The activity Lifecycle. https://developer.android.com/guide/components/activities/activity-lifecycle#saras

Save custom state. Use `onSaveInstaneState()` to implement custom saving the state of your activity. You do this by saving name-value pairs in the `Bundle` object. When you custom save state, call the super class implementation of `onSaveInstaneState()`.

```
static final String STATE_SCORE = "playerScore";
static final String STATE_LEVEL = "playerLevel";
...

@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    // Save the user's current game state
    savedInstanceState.putInt(STATE_SCORE, mCurrentScore);
    savedInstanceState.putInt(STATE_LEVEL, mCurrentLevel);

    // Always call the superclass so it can save the view hierarchy state
    super.onSaveInstanceState(savedInstanceState);
}
```

Before an activity is destroyed it calls onSaveInstaneState(), except in dire circumstances.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) Core Topics, Activities, The activity Lifecycle, Saving and restoring transietn UI state
https://developer.android.com/guide/components/activities/activity-lifecycle#saras

You can then retrieve custom state data that you saved in `onSaveInstanceState()`, in `onCreate()`. You could also use `onRestoreInstanceState()` but that is superfluous given that the `Bundle` object is passed to `onCreate()`.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) Core Topics, Activities, The activity Lifecycle, Saving and restoring transietn UI state
https://developer.android.com/guide/components/activities/activity-lifecycle#saras

Restore Custom State.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); // Always call the superclass first

    // Check whether we're recreating a previously destroyed instance
    if (savedInstanceState != null) {
        // Restore value of members from saved state
        mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
        mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
    } else {
        // Probably initialize members with default values for a new instance
    }
    ...
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) Core Topics, Activities, The activity Lifecycle, Restore activity UI state using saved instance state
https://developer.android.com/guide/components/activities/activity-lifecycle#restore-activity-ui-state-using-saved-instance-state

Always declare a View with an android:id in a layout.xml. By default the state of each View object is saved, so long as there is such an ID. E.g. The EditText widge saves any text entered by the user.

This is done by the system calling a default implementation of `onSaveInstaneState()`.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) Core Topics, Activities, The activity Lifecycle, Saving and restoring transietn UI state
https://developer.android.com/guide/components/activities/activity-lifecycle#saras

Prevent a View object from having its state saved with `android:saveEnabled="false"` or `setSaveEnabled()`.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) Core Topics, Activities, The activity Lifecycle, Saving and restoring transietn UI state
https://developer.android.com/guide/components/activities/activity-lifecycle#saras

Test saving state by rotating the device so the screen orientation changes.

When the screen orientation changes, the system destroys and recreates the activity.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) Core Topics, Activities, The activity Lifecycle, Saving and restoring transietn UI state
https://developer.android.com/guide/components/activities/activity-lifecycle#saras

Saving Fragment State

Like an activity a fragment can retain state with a Bundle object. Save state during onSaveInstanceState(). Restore state during onCreate(), onCreateView(), or onActivityCreated().

```
/**
 * An Alert List Dialog
 */
public class FireMissilesAlertListDialogFragment extends DialogFragment {
```

```
    public String[] mItems = {"ICBM", "Hellfire", "Propaganda"};
    private boolean[] mItemsCheckedStatus;

    // "Every fragment must have an empty constructor, so it can be instantiated
    // when restoring its activity's state." http://developer.android
    // .com/reference/android/app/Fragment.html
    public FireMissilesAlertListDialogFragment() {
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        // You must set the title.
        builder.setTitle("Fire Missiles Dialog");
        // Don't set a message.

        // Initialize
        if (mItemsCheckedStatus == null) {
            mItemsCheckedStatus = new boolean[mItems.length];
            for (int i = 0; i < mItems.length; i++) {
                mItemsCheckedStatus[0] = false;
            }
        }

        if (savedInstanceState != null) {
            mItemsCheckedStatus = savedInstanceState.getBooleanArray
                    ("mItemsCheckedStatus");
        }


...
    }

    /**
     * We need to save our class level variables to the outState bundle
     * so that they may be retrieved when the user:
     * 1. Selects some new values.
     * 2. Rotates the screen.
     * 3. And clicks the positive button without otherwise touching the dialog.
     */
    @Override
    public void onSaveInstanceState(Bundle outState) {
        outState.putBooleanArray("mItemsCheckedStatus", mItemsCheckedStatus);
        super.onSaveInstanceState(outState);
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\FireMissilesAlertListDialogFragment.java

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) Core Topics, Fragments, Fragment lifecycle https://developer.android.com/guide/fragments/lifecycle

"Every fragment must have an empty constructor, so it can be instantiated when restoring its **activity's** state" [Emphasis added]. Often this is all you need to do to save state, without having to mess with Bundle objects (saved with onSaveInstanceState() and restored via onCreate(), onCreateView(), ... etc.)

```
public static class PlaceholderFragment extends Fragment {


    public PlaceholderFragment() {
    }
```

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), Fragment, Default Constructor, https://developer.android.com/reference/android/app/Fragment.html

See AsyncTask Basic Procedure for a retained fragment example.

Resources

Providing resources

Providing default resources

Provide resources, which are external to your code, in order to best make your application adapt to different environments (different devices, languages, etc).

To provide a recource place it as a subdirectory of the /res/ folder

```
MyProject/
    src/
        MyActivity.java
    res/
        drawable/
            icon.png
        layout/
            main.xml
            info.xml
        values/
            strings.xml
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview https://developer.android.com/guide/topics/resources/providing-resources.html

Resource kinds and directories:

Table 1 Resource kinds and directories

Directory	Resource Type
anim/	XML files that define tween animations. (Property animations can also be saved in this directory, but the animator/ directory is preferred for property animations to distinguish between the two types.)
animator/	XML files that define property animations.
color/	XML files that define a state list of colors.
drawable/	Bitmap files (.png, .9.png, .jpg, .gif) or XML files that are compiled into the following drawable resource subtypes: • Bitmap files • Nine-Patches (re-sizable bitmaps) • State lists • Shapes • Animation drawables • Other drawables
layout/	XML files that define a user interface layout.
menu/	XML files that define application menus, such as an Options Menu, Context Menu, or Sub Menu.
raw/	Arbitrary files to save in their raw form. To open these resources with a raw InputStream, call Resources.openRawResource() with the resource ID, which is R.raw.filename. However, if you need access to original file names and file hierarchy, you might consider saving some resources in the assets/ directory (instead of res/raw/). Files in assets/ are not given a resource ID, so you can read them only using AssetManager.

values/	XML files that contain simple values, such as strings, integers, and colors. Because each resource is defined with its own XML element, you can name the file whatever you want and place different resource types in one file. However, for clarity, you might want to place unique resource types in different files. For example, here are some filename conventions for resources you can create in this directory: • arrays.xml for resource arrays (typed arrays). • colors.xml for color values • dimens.xml for dimension values. • strings.xml for string values. • styles.xml for styles.
xml/	Arbitrary XML files that can be read at runtime by calling Resources.getXML(). Various XML configuration files must be saved here, such as a searchable configuration.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview https://developer.android.com/guide/topics/resources/providing-resources.html

Providing alternate resources

The above res/ subfolders, drawable/, layout/ etc., specifiy default resources. You can provide alternative resources to handle different devices (e.g. tablets V mobiles) and different device configurations (e.g. different languages or screen in landscape).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Providing alternative resources https://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources

To provide an alternate resource create res/ subfolder and append one or more configuration qualifiers:

```
res/<recources_name>[{-<configuration_qualifier>...}]
```

```
res/
    drawable/
        icon.png
        background.png
    drawable-port-hdpi/
        icon.png
        background.png
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Providing alternative resources https://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources

The actual resources, in the example above "icon.png" and "background.png", must have identical names.

Configuration qualifier names and order of precedence.

Table 2 Configuration qualifier names and order of precedence

Config.	Qualifier Values	Description

MCC and MNC	Examples:
	`mcc310`
	`mcc310-mnc004`
		The mobile country code (MCC), optionally followed by mobile network code (MNC) from the SIM card in the device. For example,

	mcc208-mnc00 etc.	mcc310 is U.S. on any carrier, mcc310-mnc004 is U.S. on Verizon, and mcc208-mnc00 is France on Orange.
Language and region	Examples: en fr en-rUS fr-rFR fr-rCA etc.	The language is defined by a two-letter ISO 639-1 language code, optionally followed by a two letter ISO 3166-1-alpha-2 region code (preceded by lowercase "r"). The codes are **not** case-sensitive; the r prefix is used to distinguish the region portion. You cannot specify a region alone.
Layout Direction	ldrtl ldltr	The layout direction of your application. ldrtl means "layout-direction-right-to-left". ldltr means "layout-direction-left-to-right" and is the default implicit value. This can apply to any resource such as layouts, drawables, or values. For example, if you want to provide some specific layout for the Arabic language and some generic layout for any other "right-to-left" language (like Persian or Hebrew) then you would have: **Note:** To enable right-to-left layout features for your app, you must set supportsRtl to "true" and set targetSdkVersion to 17 or higher. **Added in API level 17.**
smallestWidth	sw<N>dp Examples: sw320dp sw600dp sw720dp etc.	The fundamental size of a screen, as indicated by the shortest dimension of the available screen area. Specifically, the device's smallestWidth is the shortest of the screen's available height and width (you may also think of it as the "smallest possible width" for the screen). You can use this qualifier to ensure that, regardless of the screen's current orientation, your application has at least <N> dps of width available for its UI. The smallestWidth is a fixed screen size characteristic of the device; **the device's smallestWidth does not change when the screen's orientation changes**. **Added in API level 13.**
Available width	w<N>dp Examples: w720dp w1024dp etc.	Specifies a minimum available screen width, in dp units at which the resource should be used—defined by the <N> value. This configuration value will change when the orientation changes between landscape and portrait to match the current actual width. **Added in API level 13.**
Available height	h<N>dp Examples: h720dp h1024dp etc.	Specifies a minimum available screen height, in "dp" units at which the resource should be used—defined by the <N> value. This configuration value will change when the orientation changes between landscape and portrait to match the current actual height. **Added in API level 13.**
Screen size	small normal	• small: Screens that are of similar size to a low-density QVGA screen. The minimum layout size for a small screen is

	large xlarge	approximately 320x426 dp units. Examples are QVGA low density and VGA high density. • normal: Screens that are of similar size to a medium-density HVGA screen. The minimum layout size for a normal screen is approximately 320x470 dp units. Examples of such screens a WQVGA low density, HVGA medium density, WVGA high density. • large: Screens that are of similar size to a medium-density VGA screen. The minimum layout size for a large screen is approximately 480x640 dp units. Examples are VGA and WVGA medium density screens. • xlarge: Screens that are considerably larger than the traditional medium-density HVGA screen. The minimum layout size for an xlarge screen is approximately 720x960 dp units. In most cases, devices with extra large screens would be too large to carry in a pocket and would most likely be tablet-style devices. **Added in API level 9.** **Added in API level 4.**
Screen aspect	long notlong	• long: Long screens, such as WQVGA, WVGA, FWVGA • notlong: Not long screens, such as QVGA, HVGA, and VGA **Added in API level 4.** This is based purely on the aspect ratio of the screen (a "long" screen is wider). This is not related to the screen orientation.
Screen orientation	port land	• port: Device is in portrait orientation (vertical) • land: Device is in landscape orientation (horizontal)
UI mode	car desk television appliance watch	• car: Device is displaying in a car dock • desk: Device is displaying in a desk dock • television: Device is displaying on a television, providing a "ten foot" experience where its UI is on a large screen that the user is far away from, primarily oriented around DPAD or other non-pointer interaction • appliance: Device is serving as an appliance, with no display • watch: Device has a display and is worn on the wrist **Added in API level 8, television added in API 13, watch added in API 20.**
Night mode	night notnight	• night: Night time • notnight: Day time **Added in API level 8.**
Screen pixel density (dpi)	ldpi mdpi hdpi xhdpi xxhdpi xxxhdpi nodpi tvdpi	• ldpi: Low-density screens; approximately 120dpi. • mdpi: Medium-density (on traditional HVGA) screens; approximately 160dpi. • hdpi: High-density screens; approximately 240dpi. • xhdpi: Extra high-density screens; approximately 320dpi. **Added in API Level 8** • xxhdpi: Extra-extra-high-density screens; approximately 480dpi. Added in API Level 16

		• xxxhdpi: Extra-extra-extra-high-density uses (launcher icon only, see the note in Supporting Multiple Screens); approximately 640dpi. Added in API Level 18 • `nodpi`: This can be used for bitmap resources that you do not want to be scaled to match the device density. • `tvdpi`: Screens somewhere between mdpi and hdpi; approximately 213dpi. This is not considered a "primary" density group. It is mostly intended for televisions and most apps shouldn't need it—providing mdpi and hdpi resources is sufficient for most apps and the system will scale them as appropriate. This qualifier was introduced with API level 13.
Touchscreen type	`notouch` `finger`	• `notouch`: Device does not have a touchscreen. • `finger`: Device has a touchscreen that is intended to be used through direction interaction of the user's finger.

Keyboard availability	`keysexposed`
	`keyshidden`
	`keyssoft`
		• `keysexposed`: Device has a keyboard available. If the device has a software keyboard enabled (which is likely), this may be used even when the hardware keyboard is **not** exposed to the user, even if the device has no hardware keyboard. If no software keyboard is provided or it's disabled, then this is only used when a hardware keyboard is exposed.
		• `keyshidden`: Device has a hardware keyboard available but it is hidden **and** the device does **not** have a software keyboard enabled.
		• `keyssoft`: Device has a software keyboard enabled, whether it's visible or not.

Primary text input method	`nokeys`
	`qwerty`
	`12key`
		• `nokeys`: Device has no hardware keys for text input.
		• `qwerty`: Device has a hardware qwerty keyboard, whether it's visible to the user or not.
		• `12key`: Device has a hardware 12-key keyboard, whether it's visible to the user or not.

Navigation key availability	`navexposed` `navhidden`	• `navexposed`: Navigation keys are available to the user. • `navhidden`: Navigation keys are not available (such as behind a closed lid).

Primary non-touch navigation method	`nonav`
	`dpad`
	`trackball`
	`wheel`
		1. `nonav`: Device has no navigation facility other than using the touchscreen.
		2. `dpad`: Device has a directional-pad (d-pad) for navigation.
		3. `trackball`: Device has a trackball for navigation.
		
		`wheel`: Device has a directional wheel(s) for navigation (uncommon).

Platform Version (API level)	Examples:
	`v3`
	`v4`
	`v7`
	etc.
		The API level supported by the device. For example, `v1` for API level 1 (devices with Android 1.0 or higher) and `v4` for API level 4 (devices with Android 1.6 or higher). See the Android API levels document for more information about these values.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Providing alternative resources https://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources

Alternate configuration naming rules:

The configuration qualifiers can only be added in order of precedence.

```
Wrong: drawable-hdpi-port/
Correct: drawable-port-hdpi/
```

Folder names are case insensitive. The compiler converts folder names to lowercase for matching. Any capitalisation is just used for readability.

Only one qualifier for each type is supported.

```
Wrong: drawable-rES-rFR/
Right: drawable-rES/
```

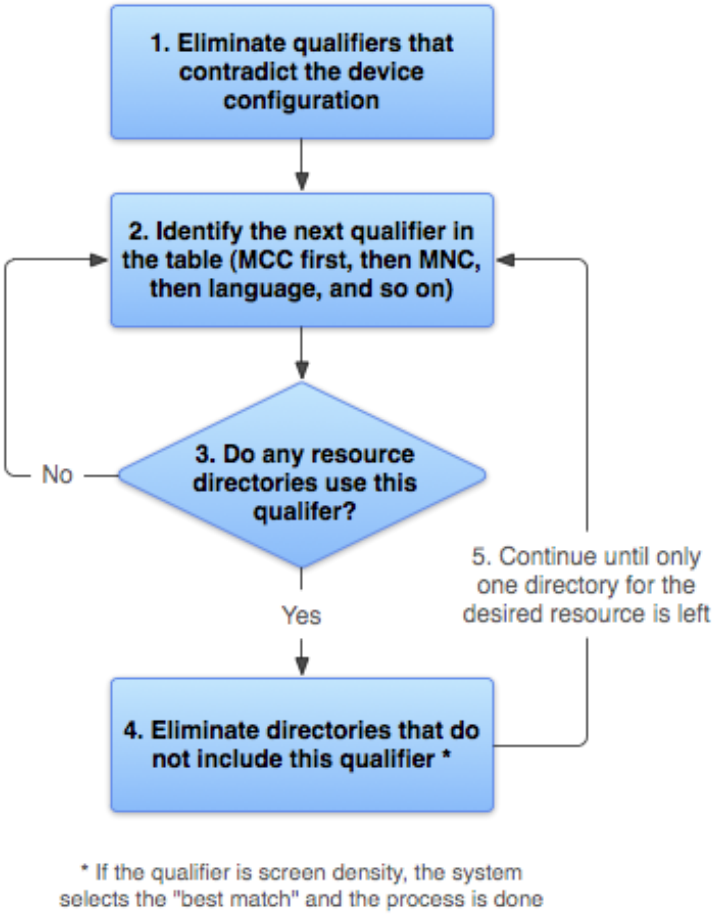
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Providing alternative resources, Qualifier name rules.
https://developer.android.com/guide/topics/resources/providing-resources.html#QualifierRules

If you provide an alternative resource you ought always provide a default resource.

Otherwise you application will fuck up when it meets a configuration that you didn't handle.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Providing the best device compatibility with resources
https://developer.android.com/guide/topics/resources/providing-resources.html#Compatibility

Alternative resource matching algorithm:



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, How Android finds the best-matching resource https://developer.android.com/guide/topics/resources/providing-resources.html#BestMatch

Resource references

Overview

The syntax of the resource reference depends upon whether you are referencing the resource from Java code or from xml.

Resource reference types

All resources resolve to a "resource ID" which is distinct from an "ID resource type". I'll speak of a "resource ID" as a "resource reference" to avoid the ambiguity. A resource reference is an integer (and generated automatically at compile time).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Accessing your app resources https://developer.android.com/guide/topics/resources/providing-resources#Accessing

A resource is one of three types:

1. A file (e.g. a drawable), which is referenced through the filename.

```
// Resource definition
res/drawable/myimage.png

// Resource reference, from XML
<ImageView
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/myimage" />

// Resource reference, from Java
Resources res = getResources();
Drawable drawable = res.getDrawable(R.drawable.myimage);
```

2. Value or value array (e.g. a string or string array), which is referenced through the name attribute (e.g. as for a string);

```
// Resource definition in \res\values\strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello_world">Hello world!</string>
</resources>

// Resource reference, from XML (using a resource alias in this case).
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello_world">Hello world!</string>
    <string name="hello_world_alias">@string/hello_world</string>
</resources>

// Resource references, from Java
// Get a string resource from your app's Resources
String hello = getResources().getString(R.string.hello_world);

// Or supply a string resource to a method that requires a string
TextView textView = new TextView(this);
textView.setText(R.string.hello_world);
```

3. Object (e.g. a button), which is referenced through the id attribute.

```
// Resource definition in \res\layout\my_activity.xml
...
    <Button
        android:id="@+id/button_writeFiles"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:text="Write Files" />

// Resource reference, from XML (in this case within the same file).
    <TextView
        android:id="@+id/textView_output"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="@id/button_writeFiles"
        android:layout_below="@id/button_writeFiles"
        android:text="@string/overwritten" />

// Resource references, from Java
TextView outputTextView = (TextView) getActivity().findViewById(R.id.textView_output);
outputTextView.setText(message);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Grouping resource types https://developer.android.com/guide/topics/resources/providing-resources.html#ResourceTypes

Resource referencing overview:

- Files in the `res/values/` directory describe multiple resources. For a file in this directory, each child of the <resources> element defines a single resource. It is referenced in java code with the "name" attribute.

```
<!-- res/values/strings.xml: -->
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello!</string>
</resources>


// In Java
// Reference by the attribute "name", not by the filename "strings".
String string = getString(R.string.hello);
```

- Whereas XML resource files in other `res/` subdirectories define a single resource based on the XML filename.

```
<!-- res/layout/activity_database.xml: -->
<ScrollView
    android:id="@+id/scrollView1"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    >

    <LinearLayout
...

// In Java
public class DatabaseActivity extends Activity {

    NorthwindDBHelper mDBHelper = new NorthwindDBHelper(this);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Reference by file name
        setContentView(R.layout.activity_database);
...
```

(Google 2013, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Grouping resource types https://developer.android.com/guide/topics/resources/providing-resources.html#ResourceTypes

References to resources are always scoped by the resource type (such as `id` or `string`), so using the same name does not cause collisions.

```
<!-- OK -->
<EditText
    android:id="@+id/edit_message"
...
    android:hint="@string/edit_message" />

<!-- The first defines an identifier recourse the second references a concrete string
resource defined in strings.xml -->
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, Build your first app, Build a simple user interface, https://developer.android.com/training/basics/firstapp/building-ui.html

Resource reference aliases

Create a recourse that is referenced by more than one alternate configuration:

For resources that are referenced via file name: create a text file that aliases another resource.

```
res/drawable/
    icon.png
    icon_alternate.png
res/drawable-en-rCA/
    icon.xml
res/drawable-fr-rCA/
    icon.xml

<!-- icon.xml in both directories -->
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/icon_alternate" />

// Reference in Java code as usual
R.drawable.icon
```

For resources that are referenced via a name attribute: just use the resource ID (usually via the name attribute) of the value.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="yellow">#f00</color>
    <color name="highlight">@color/yellow</color>
</resources>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Creating alias resources https://developer.android.com/guide/topics/resources/providing-resources.html#AliasResources

Aliasing is available for all resource types except animation, menu, raw, and other unspecified resources in the `xml/`.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Creating alias resources https://developer.android.com/guide/topics/resources/providing-resources.html#AliasResources

Drawable xml file alias example.

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/icon_ca" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Creating alias resources https://developer.android.com/guide/topics/resources/providing-resources.html#AliasResources

Layout xml file alias example.

```
<?xml version="1.0" encoding="utf-8"?>
<merge>
    <include layout="@layout/main_ltr"/>
</merge>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Creating alias resources https://developer.android.com/guide/topics/resources/providing-resources.html#AliasResources

String xml file alias example.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello</string>
    <string name="hi">@string/hello</string>
</resources>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Creating alias resources https://developer.android.com/guide/topics/resources/providing-resources.html#AliasResources

Resource references in code (Java)

Intro

Resource reference code syntax:

```
[android.|<package_name>.]R.<resource_type>.<resource_name>
```

- <package_name> is the name of the package in which the resource is located (not required when referencing resources from your own package).
- "android." is used if you are referencing a resource defined in the android framework, rather than a local resource.
- <resource_type> is the R subclass for the resource type. A resource_type includes the "id" type.
- <resource_name> is either the resource filename without the extension or the android:name attribute value in the XML element (for simple values).

```
// Reference a file resource
getMenuInflater().inflate(R.menu.main, menu);

// Reference a value or value array resource
textView.setText(R.string.hello_world);
...
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);

// Reference an object
EditText editText = (EditText) findViewById(R.id.edit_message);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Accessing resources in code https://developer.android.com/guide/topics/resources/providing-resources#ResourcesFromCode

When you reference an android resource, rather than a local resource, you must use the android package name.

```
if (savedInstanceState == null) {
    getFragmentManager().beginTransaction().replace(android.R.id.content,
            new SettingsFragment()).commit();
}

setListAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
myarray));
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\SettingsActivity.java

Common helper methods for referencing in Java

In code you can use a resource reference by passing it as a method parameter. Sometimes a method parameter can just take a resource parameter directly (as an int). At other times you need to retrieve and convert the resource reference into a java object. Two methods that are useful for this are Activity.**findViewById()** and Context.**getResources()**, to return a Resources object. The Resources object has many methods for access, like getString, getColor, getDrawable, getStringArray, getText, ...

```
// Direct resource reference
imageView.setImageResource(R.drawable.myimage);

// Retrieve and convert into Java Object
ImageView imageView = (ImageView) findViewById(R.id.myimageview);
...
String hello = getResources().getString(R.string.hello_world);
```

Reference a UI component with `findViewById()`

```
EditText editText = (EditText) findViewById(R.id.edit_message);
String message = editText.getText().toString();

TextView textView = (TextView) findViewById(R.id.output);
textView.setText(getResources().getString(R.string.cool_string));
```

Retrieve string resources in code

```
// Get a string resource from your app's Resources
String hello = getResources().getString(R.string.hello_world);

// Or supply a string resource to a method that requires a CharSequence
TextView textView = new TextView(this);
textView.setText(R.string.hello_world);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) Devices, Device compatibility, Support different languages and cultures, Use the resources in your app, ***Error! Hyperlink reference not valid.*** *https://developer.android.com/training/basics/supporting-devices/languages.html#UseAppResources*

Retrieve string array.

```
<!-- XML file saved at res/values/strings.xml -->
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
    </string-array>
</resources>

// In Code
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, String resources, String Array https://developer.android.com/guide/topics/resources/string-resource.html#StringArray

Access a file directly in Java

Sometimes you want to access a file internal to your app; and you want to do so directly, rather, that is, from an Android Context (and then with a `[android.|<package_name>.]R.<resource_type>.<resource_name>)`. You have two choices as to location: the `res/raw` folder or `assets/` folder (outside of the res parent). To choose between the two note from https://developer.android.com/guide/topics/resources/providing-resources.html

> Arbitrary files to save in their raw form. To open these resources with a raw InputStream, call Resources.openRawResource() with the resource ID, which is R.raw.filename.
>
> However, if you need access to original file names and file hierarchy, you might consider saving some resources in the assets/ directory (instead of res/raw/). Files in assets/ aren't given a resource ID, so you can read them only using AssetManager.

To access a file in res/raw/ directly rather, that is, from an Android Context (and then with a [android.|<package_name>.]R.<resource_type>.<resource_name>) you can do something like this:

```
public static void outputCountries() throws IOException {

    OutputStream outputStream = System.out;
    BufferedReader bufferedReader = null;
    File file = null;

    try {
        file = new File("app/src/main/res/raw/country_data_from_world_bank.xml");
        bufferedReader = new BufferedReader(new FileReader(file));

        String line;
        while ((line = bufferedReader.readLine()) != null) {
            System.out.println(line);
        }
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (bufferedReader != null) {
            bufferedReader.close();
        }
    }
}
```

(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/) "Getting file path for local Android project files", https://stackoverflow.com/a/49765227/872154
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\recyclerviewdemo\CountryXmlImporter.java

Resource references in XML

General

Resource reference in xml syntax:

```
@[android:|<package_name>:]<resource_type>/<resource_name>
```

- <package_name> is the name of the package in which the resource is located (not required when referencing resources from the same package)
- "android" is used if you are referencing a resource defined in the android framework, rather than a local resource.
- <resource_type> is the R subclass for the resource type
- <resource_name> is either the resource filename without the extension or the android:name attribute value in the XML element (for simple values).

```
// Reference a file resource
<?xml version="1.0" encoding="utf-8"?>
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/other_drawable" />

// Reference a value resource
<EditText
...
    android:text="@string/hello_world" />

// Reference an object
<Button
...
    android:layout_below="@id/imageView_picture"
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Accessing your app recourses, Accessing resources from XML, https://developer.android.com/guide/topics/resources/providing-resources#ResourcesFromXml

When you reference an android resource, rather than a local resource, you must use the android package name.

```
<EditText xmlns:android="http://schemas.android.com/apk/res/android"
    ...
    android:textColor="@android:color/secondary_text_dark"
    ... />
```

Style [Attribute] Resource

A style attribute references an attribute in the currently applied theme. Referencing a style attribute essentially says, "use the style that is defined by this attribute, in the current theme."

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Accessing your app recourses, Accessing resources from XML, Referencing style attributes, https://developer.android.com/guide/topics/resources/providing-resources#ReferencesToThemeAttributes

Style attribute resource reference in XML:

```
?([android|<package_name>]:)[attr/]<resource_name>
```

Note:

- Use a question mark "?" rather than an at symbol "@".
- "attr" can be regarded as a resource type. It is optional.

```
<EditText id="text"
...
    android:textColor="?android:textColorSecondary"
    android:text="@string/hello_world" />

<!-- Explicit "attr" -->
    android:textColor="?android:attr/textColorSecondary"
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Overview, Accessing your app recourses, Accessing resources from XML, Referencing style attributes, https://developer.android.com/guide/topics/resources/providing-resources#ReferencesToThemeAttributes

ID Resource Type

The ID resource is just another type of resource, which a special rule: Use a plus (+) symbol when defining.

To define an ID resource:

```
@+id/<resource_name>
```

The at-symbol "@" indicates that the XML parser should expand the rest of the ID string and identify it as an ID resource. The plus-symbol "+" indicates this is a new resource name (and so it must be created and added to the resources in R.java).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) Core Topics, User Interface, Layouts, Overview, Attributes, ID https://developer.android.com/guide/topics/ui/declaring-layout.html#id

The plus sign (+) is only needed when you define the ID resource for the first time.

```
<EditText
    android:id="@+id/edit_message"
...
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, Build your first app, Build a simple user interface, https://developer.android.com/training/basics/firstapp/building-ui.html

It is good programming practice to declare all View objects with an ID.

This ensures that state is maintained. Also view objects in a RelattiveLayout position each other by referencing their siblings (and so they need an ID to reference).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) Core Topics, User Interface, Layouts, Overview, Attributes, ID https://developer.android.com/guide/topics/ui/declaring-layout.html#id

Subsquent references to that identifier resource omit the plus sign.

```
// Resource definition in \res\layout\my_activity.xml
...
    <Button
        android:id="@+id/button_writeFiles"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:text="Write Files" />

// Resource reference, from XML (plus sign omitted).
    <TextView
        android:id="@+id/textView_output"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="@id/button_writeFiles"
        android:layout_below="@id/button_writeFiles"
        android:text="@string/overwritten" />
```

When you reference an android id resource, rather than a local resource, you must use the android package namespace.

```
android:id="@android:id/empty"
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) Core Topics, User Interface, Layouts, Overview, Attributes, ID https://developer.android.com/guide/topics/ui/declaring-layout.html#id

Localisation

Localisation is achieved just be leveraging Android's resources framework. Cheifly through separating all UI strings into res/values/strings.xml and providing an alternate resource for different regions and languages. E.g. res/values-fr/strings.xml.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Localization, Localize your app, https://developer.android.com/guide/topics/resources/localization.html

Localisation Checklist ...

(Google 2021, "Google Play > Best Practices (Strategies)", https://developer.android.com/distribute/best-practices) Launch, Localization checklist, https://developer.android.com/distribute/best-practices/launch/localization-checklist

You might also need to use Java localisation technqiues. See ...

(Bentley 2013, "JavaReference-Framework.Docx", C:\Users\John\Documents\Sda\Info\Java\KB\Reference\JavaReference-Framework.docx) Internationalisation.

Resource Types

File resources

Animation

The types of animation are:

1. Property animation (using `Animator` and extensions `ValueAnimator`, `ObjectAnimator`, or `AnimatorSet`). In `res/animator`/filename.xml. Modify an object's properties over time.
2. View animation, of which there are two types. In `res/anim`/filename.xml. View Animations operate on an image or images.
    - Tween Animation. Transform a single graphic with an `Animation`.
    - Frame Animation. Show a sequence of images with `AnimationDrawable`.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Animation, https://developer.android.com/guide/topics/resources/animation-resource.html

Property animation. Modifies an object's property over a time period with the `Animator`.

```
res/animator/property_animator.xml

<set android:ordering="sequentially">
    <set>
        <objectAnimator
            android:propertyName="x"
            android:duration="500"
            android:valueTo="400"
            android:valueType="intType"/>
        <objectAnimator
            android:propertyName="y"
```

```
            android:duration="500"
            android:valueTo="300"
            android:valueType="intType"/>
    </set>
    <objectAnimator
        android:propertyName="alpha"
        android:duration="500"
        android:valueTo="1f"/>
</set>

Animator animatorSet = AnimatorInflater.loadAnimator
        (getBaseContext(), R.animator.property_animator);
TextView textView = findViewById(R.id.textView_property_animation_target);
animatorSet.setTarget(textView);
animatorSet.start();
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Animation, Property animation, https://developer.android.com/guide/topics/resources/animation-resource.html#Property

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\resourcedemo\FileResourcesActivity.java

View animation, Tween. Transform a single graphic.

```
\res\anim\hyperspace_jump.xml

<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false"
    >
    <set android:interpolator="@android:anim/accelerate_decelerate_interpolator">

        <scale
            android:duration="700"
            android:fillAfter="false"
            android:fromXScale="1.0"
            android:fromYScale="1.0"
            android:pivotX="50%"
            android:pivotY="50%"
            android:toXScale="1.4"
            android:toYScale="0.6"
            />
        <translate
            android:duration="1000"
            android:fromXDelta="0"
            android:fromYDelta="0"
            android:toXDelta="20%"
            android:toYDelta="30%"
            />
    </set>
    <set
        android:interpolator="@android:anim/overshoot_interpolator"
        android:startOffset="1000"
        >
        <scale
            android:duration="400"
            android:fromXScale="1.4"
            android:fromYScale="0.6"
            android:pivotX="50%"
            android:pivotY="50%"
            android:toXScale="0.0"
            android:toYScale="0.0"
            />
        <rotate
            android:duration="400"
            android:fromDegrees="0"
            android:pivotX="50%"
            android:pivotY="50%"
            android:toDegrees="-45"
            android:toYScale="0.0"
            />
        <alpha
```

```
            android:duration="200"
            android:fromAlpha="1.0"
            android:toAlpha="0.2" />
    </set>
</set>

TextView textView = (TextView) findViewById(R.id.textView_property_animation_target);
Animation hyperspaceJump = AnimationUtils.loadAnimation(this,
                                                        R.anim.hyperspace_jump);
textView.startAnimation(hyperspaceJump);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Animation, View Animation, Tween animation
https://developer.android.com/guide/topics/resources/animation-resource.html#View

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\resourcedemo\FileResourcesActivity.java

There are several built in interpolators, which specify the rate of change of an animation. See:

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Animation, View Animation, Tween animation, Interpolartors
https://developer.android.com/guide/topics/resources/animation-resource#Interpolators

Custom interpolators modify the behaviour of built-in interpolators.

```
\res\anim\custom_overshoot_interpolator.xml

<?xml version="1.0" encoding="utf-8"?>
<overshootInterpolator xmlns:android="http://schemas.android.com/apk/res/android"
    android:tension="7.0"
    />

In \res\anim\hyperspace_jump.xml change to

    </set>
    <set
        android:interpolator="@anim/custom_overshoot_interpolator"
        android:startOffset="1000"
        >
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Animation, View Animation, Tween animation
https://developer.android.com/guide/topics/resources/animation-resource.html#View

View animation, Frame. A sequence of images.

```
<!-- This is in \drawable not \anim -->
\res\drawable\dramatic_chipmunk.xml

<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true"
    >
    <!-- Individual images -->
    <item android:drawable="@drawable/dramatic_chipmunk_0001" android:duration="200" />
    <item android:drawable="@drawable/dramatic_chipmunk_0002" android:duration="200" />
...
    <item android:drawable="@drawable/dramatic_chipmunk_0011" android:duration="200" />
    <item android:drawable="@drawable/dramatic_chipmunk_0012" android:duration="200" />
 </animation-list>


<!-- \res\layout\activity_file_resources.xml -->
<RelativeLayout xmlns:android=http://schemas.android.com/apk/res/android
  ...
    <Button
```

```
        android:id="@+id/button_animate_frames"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/textView_property_animation_target"
        android:onClick="animateFrames"
        android:text="Animate Frames" />

    <ImageView
        android:id="@+id/imageView_chipmunk"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/button_animate_frames" />

</RelativeLayout>

public void animateFrames(View view) {
    ImageView chipmunkImage = (ImageView) findViewById(R.id.imageView_chipmunk);

    // Reset animation if already run.
    if (Build.VERSION.SDK_INT < 16 ) {
        chipmunkImage.setBackgroundDrawable(null);
    } else {
        chipmunkImage.setBackground(null);
    }

    // dramatic_chipmunk is in \drawable not \anim
    chipmunkImage.setBackgroundResource(R.drawable.dramatic_chipmunk);

    AnimationDrawable chipmunkAnimation = (AnimationDrawable) chipmunkImage
            .getBackground();
    chipmunkAnimation.start();
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Animation, View Animation, Frame animation
https://developer.android.com/guide/topics/resources/animation-resource#Frame
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\resourcedemo\FileResourcesActivity.java

Color state list

Color state list. Works on android:textColor, not android:background.

```
res/color/my_button_color_state_list.xml

<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
          android:color="#ffff0000"/> <!-- pressed -->
    <item android:state_focused="true"
          android:color="#ff0000ff"/> <!-- focused -->
    <item android:color="#ff000000"/> <!-- default -->
</selector>

<Button
    android:id="@+id/button_animate_frames"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_below="@+id/textView_property_animation_target"
    android:textColor="@color/my_button_color_state_list"
    android:onClick="animateFrames"
    android:text="Animate Frames" />
```

Color state lists appear to work on android:textColor, not android:background. To work with android:background you'll have to use a State List Drawable. See http://stackoverflow.com/a/18131705/872154

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Color state list resource, https://developer.android.com/guide/topics/resources/color-list-resource.html

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\activity_file_resources.xml

Drawables

Bitmap drawables (Bitmap and XML Bitmap)

Bitmap drawable. A bitmap image in one of three formats: .png (preferred), .jpg (acceptable), .gif (discouraged).

```
res/drawable/myimage.png

<ImageView
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/myimage" />

Resources res = getResources();
Drawable drawable = res.getDrawable(R.drawable.myimage);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Bitmap, https://developer.android.com/guide/topics/resources/drawable-resource.html#Bitmap

To avoid any optimisations on a bitmap, as when reading an image as a bitstream, you might prefer to put the image in res/raw.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Bitmap, https://developer.android.com/guide/topics/resources/drawable-resource.html#Bitmap

XML Bitmap drawable. Points to a bitmap file. Serves several purposes:

1. An alias for a drawable. See also Resource aliases.

```
<!-- A bitmap -->
res\drawable\wistful_cat.png

<!-- An alias to the bitmap -->
\res\drawable\cat_alias.xml
<?xml version="1.0" encoding="utf-8"?>
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/wistful_cat"/>

<!-- Use the alias -->
<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:drawable/screen_background_dark_transparent"
    android:src="@drawable/cat_alias" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Bitmap, https://developer.android.com/guide/topics/resources/drawable-resource.html#Bitmap

2. Applying effects to a bitmap, such as dithering, tiling, and positioning within the container.

```
res\drawable\wistful_cat.png

res\drawable\cat_mod.xml
<?xml version="1.0" encoding="utf-8"?>
<!-- Hosting ImageView must have android:scaleType="fitXY" for gravity and tiling to
take effect (and possible ImageView height with a fixed value(e.g. 600 pd) -->
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/wistful_cat"
    android:gravity="center|clip_horizontal"
    android:tileMode="mirror"/>

<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="600dp"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:scaleType="fitXY"
    android:layout_below="@+id/textView_drawables_title"
    android:background="@android:drawable/screen_background_dark_transparent"
    android:src="@drawable/cat_mod"
    android:padding="@dimen/ui_padding" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Bitmap, https://developer.android.com/guide/topics/resources/drawable-resource.html#Bitmap
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\drawable\cat_mod.xml

3. Use as a child element in lists, like a State List or Layer List.

Nine-Patch

A nine-patch drawable is a bitmap image that is stretchable. It is a standard PNG image that includes an extra 1 pixel wide black border. The image must be saved with the extension .9.png

```
\res\drawable-mdpi\appwidget_dark_bg.9.png
\res\drawable-hdpi\appwidget_dark_bg.9.png
\res\drawable-xhdpi\appwidget_dark_bg.9.png

<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:background="@drawable/appwidget_dark_bg" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Nine-Patch, https://developer.android.com/guide/topics/resources/drawable-resource.html#NinePatch

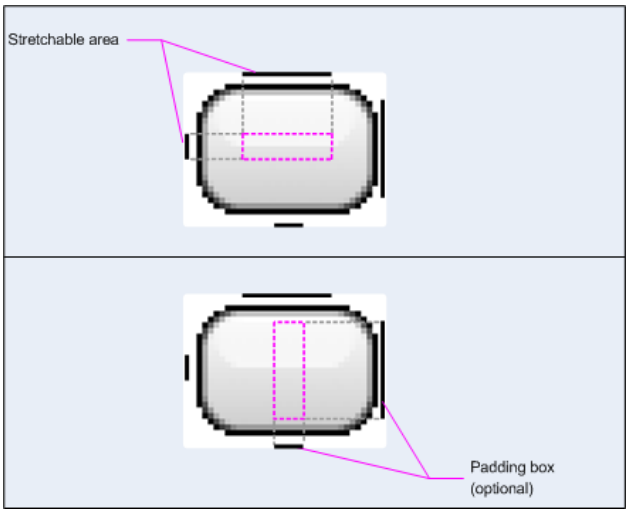
Xml Nine-Patch. A resource that points to a Nine-Patch, serving to specify dithering along the way.

```
\res\drawable-mdpi\myninepatch.9.png

res\drawable\myninepatch_mod.xml
<?xml version="1.0" encoding="utf-8"?>
<nine-patch xmlns:android="http://schemas.android.com/apk/res/android"
   android:src="@drawable/myninepatch"
   android:dither="false" />
```

```
<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:background="@drawable/myninepatch_mod" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Nine-Patch, https://developer.android.com/guide/topics/resources/drawable-resource.html#NinePatch

The lines of the extra border define the stretch and text containment behaviour. The top and left lines define the stretching, the bottom and right lines define the text containment.



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Nine-Patch, https://developer.android.com/guide/topics/resources/drawable-resource.html#NinePatch

Obtain some 9 patches from the App Widget Templates Pack.

(Google 2020, "App Widget Design Guidelines", https://developer.android.com/guide/practices/ui_guidelines/widget_design), Using the App Widget Templates Pack, https://developer.android.com/guide/practices/ui_guidelines/widget_design.html#templates

Adjust the extra border of your 9 patches by using the Draw 9-Patch tool. The tools is available at Android\sdk\tools\draw9patch.bat. But the easiest way to use it is to drag your PNG to /res/drawable/ and double click on it to let Android Studio open the Draw 9-Patch tool.

Layer List

Layer List. A layer list is a LayerDrawable that specifies an array of other drawables. Each drawable in the list is drawn in order, the last on top.

```
\res\drawable\android_red.png
\res\drawable\android_green.png
\res\drawable\android_red.png

\res\drawable\android_icon_layer_list.png
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
```

```
    <item>
        <bitmap android:gravity="center"
                android:src="@drawable/android_red" />
    </item>
    <item android:left="10dp" android:top="10dp">
        <bitmap android:gravity="center"
                android:src="@drawable/android_green" />
    </item>
    <item android:left="20dp" android:top="20dp">
        <bitmap android:gravity="center"
                android:src="@drawable/android_blue" />
    </item>
</layer-list>

<ImageView
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/android_icon_layer_list" />
```

(Google 2013, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Layer List, https://developer.android.com/guide/topics/resources/drawable-resource.html#LayerList

State List

State list. Changes the background drawable based on state. To change the foreground text use a Color state list.

```
// Physical Files
\res\drawable\button_hovered_blue.png
\res\drawable\button_normal.png
\res\drawable\button_pressed.png

// State List XML
\res\drawable\button_state_list.xml

<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
        android:drawable="@drawable/button_pressed" /> <!-- pressed -->
    <item android:state_focused="true"
        android:drawable="@drawable/button_hovered_blue" /> <!-- focused -->
    <item android:state_hovered="true"
        android:drawable="@drawable/button_hovered_blue" /> <!-- hovered -->
    <item android:drawable="@drawable/button_normal" /> <!-- default -->
</selector>

// State List Referenced from
\res\layout\activity_file_resources.xml

<Button
    android:id="@+id/button_state_list_target"
    android:layout width="wrap content"
    android:layout_height="wrap_content"
    android:background="@drawable/button_state_list" />
```

In the state list Android applies the first item that matches the current state. It does not do a "best match" search.

(Google API Guides, 2013. Develop > API Guides) http://developer.android.com/guide/topics/resources/drawable-resource.html#StateList

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, State list, https://developer.android.com/guide/topics/resources/drawable-resource.html#StateList

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\drawable\button_state_list.xml

Level List

A file drawable that displays different drawables according to an assigned level. You set the level in code with `setLevel()` or `setImageLevel()`. Each drawable item will display if it is the first one to be within range of the setLevel, as defined by that item's minLevel and maxLevel properties.

```
// res\layout\myLayout.xml
<ImageView
    android:id="@+id/imageView_level_list_target"
    android:layout_width="match_parent"
    android:layout_height="300dp"
    android:layout_below="@+id/textView_level_list"
    android:src="@drawable/dramatic_chipmunk_level_list" />

<NumberPicker
    android:id="@+id/numberPicker_level_list"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/imageView_level_list_target" />

// res\drawable\my_list_level.xml
<?xml version="1.0" encoding="utf-8"?>
<level-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:drawable="@drawable/dramatic_chipmunk_0001"
        android:minLevel="1"
        android:maxLevel="5" />
    <item
        android:drawable="@drawable/dog"
        android:minLevel="6"
        android:maxLevel="9" />
    <item
        android:drawable="@drawable/dramatic_chipmunk_0034"
        android:minLevel="10"
        android:maxLevel="15" />
</level-list>

// Java Activity
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_file_resources);

    NumberPicker levelListNumberPicker = (NumberPicker)
findViewById(R.id.numberPicker_level_list);

    NumberPicker.OnValueChangeListener onValueChanged = new NumberPicker
            .OnValueChangeListener() {
        @Override
        public void onValueChange(NumberPicker picker, int oldValue, int newValue) {
            ImageView levelListTargetImageView = (ImageView)
findViewById(R.id.imageView_level_list_target);
            levelListTargetImageView.setImageLevel(newValue);
        }
    };

    levelListNumberPicker.setWrapSelectorWheel(false);
    levelListNumberPicker.setOnValueChangedListener(onValueChanged);
    levelListNumberPicker.setMinValue(0);
    levelListNumberPicker.setMaxValue(15);
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Level list, https://developer.android.com/guide/topics/resources/drawable-

resource.html#LevelList
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\activity_file_resources.xml

Transition Drawable

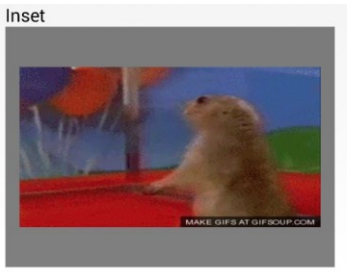
A transition drawable is a drawable that can cross-fade between two drawables.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Transition drawable, https://developer.android.com/guide/topics/resources/drawable-resource.html#Transition

Inset Drawable

An inset drawable in xml sort of puts padding within the UI component...

```
// res\layout\activity_file_resources.xml (Layout XML)
<ImageView
    android:id="@+id/imageView_inset_target"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:src="@drawable/inset_dramatic_chipmunk"
    android:background="@android:drawable/screen_background_dark_transparent" />

// \res\drawable\inset_dramatic_chipmunk.xml (The Inset Drawable)
<?xml version="1.0" encoding="utf-8"?>
<inset xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/dramatic_chipmunk_0001"
    android:insetTop="@dimen/ui_large"
    android:insetRight="@dimen/ui_margin"
    android:insetBottom="@dimen/ui_large"
    android:insetLeft="@dimen/ui_margin"
    />
```



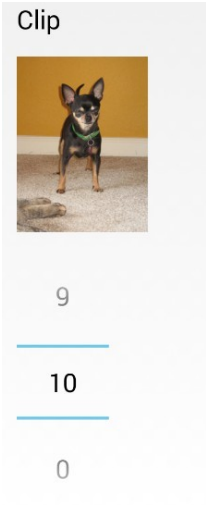
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Inset drawable, https://developer.android.com/guide/topics/resources/drawable-resource.html#Inset

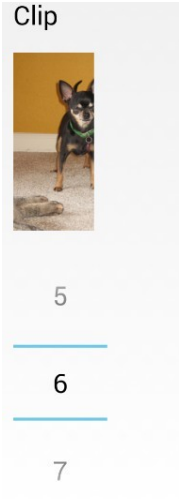
Clip Drawable

A clip drawable clips another drawable based on a level. Often used to implement progress bars. The level is defined to be between 0 (show nothing) and 10,000 (show everything).

```
<!-- \res\layout\activity_file_resources.xml (Layout File) -->

<ImageView
    android:id="@+id/imageView_clip_dog"
    android:layout_width="wrap_content"
    android:layout_height="150dp"
    android:layout_alignEnd="@+id/button_animate"
    android:layout_alignRight="@+id/button_animate"
    android:layout_below="@+id/textView_clip"
```

```
    android:src="@drawable/clip_dog" />

<NumberPicker
    android:id="@+id/numberPicker_clip_dog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/imageView_clip_dog" />

<!-- \res\drawable\clip_dog.xml (The Clip Drawable) -->
<?xml version="1.0" encoding="utf-8"?>
<clip xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/dog"
    android:clipOrientation="horizontal"
    android:gravity="left"
 />

// Code
public class FileResourcesActivity extends Activity {

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

        NumberPicker dogClipNumberPicker = (NumberPicker)
            findViewById(R.id.numberPicker clip dog);
        NumberPicker.OnValueChangeListener dogClipOnValueChanged = new NumberPicker
                .OnValueChangeListener() {
            @Override
            public void onValueChange(NumberPicker picker, int oldValue, int newValue) {
                ImageView dogClipImageView = (ImageView)
                     findViewById(R.id.imageView clip dog);
                ClipDrawable dogClipDrawable = (ClipDrawable) dogClipImageView
                        .getDrawable();
                dogClipDrawable.setLevel(newValue * 1000);
            }
        };
        dogClipNumberPicker.setWrapSelectorWheel(false);
        dogClipNumberPicker.setOnValueChangedListener(dogClipOnValueChanged);
        dogClipNumberPicker.setMinValue(0);
        dogClipNumberPicker.setMaxValue(10);
    }
```





(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Clip drawable, https://developer.android.com/guide/topics/resources/drawable-resource.html#Clip

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\resourcedemo\FileResourcesActivity.java

Scale Drawable

A drawable that changes the scale of another drawable.

```
<?xml version="1.0" encoding="utf-8"?>
<scale xmlns:android="http://schemas.android.com/apk/res/android"
  android:drawable="@drawable/logo"
  android:scaleGravity="center_vertical|center_horizontal"
  android:scaleHeight="80%"
  android:scaleWidth="80%" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Scale drawable, https://developer.android.com/guide/topics/resources/drawable-resource.html#Scale

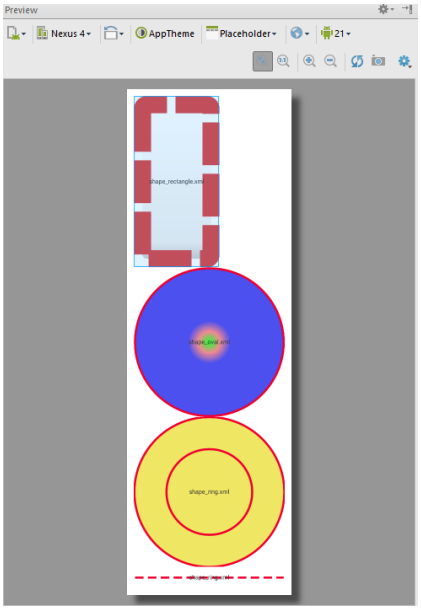
Shape Drawable

A shape drawable is rectangle, oval, line, or ring defined in XML with various properties. Create it by:

1. Define the shape drawable in \res\drawable. Specify the shape type ["rectangle" | "oval" | "line" | "ring"] at android:shape. Specify other properties are necessary.

```
\res\drawable\shape_rectangle.xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <gradient
        android:angle="270"
        android:centerColor="#f2efef"
        android:centerY="0.9"
        android:endColor="#cecccc"
        android:startColor="#FFFFFF"
        android:type="linear" />
    <padding
        android:bottom="7dp"
        android:left="7dp"
        android:right="7dp"
        android:top="7dp" />
    <corners android:radius="20dp" />
    <stroke
        android:width="40dp"
        android:color="#db4343"
        android:dashGap="20dp"
        android:dashWidth="100dp" />
</shape>

\res\drawable\shape_oval.xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <gradient
        android:type="radial"
        android:gradientRadius="50%"
        android:startColor="#ff1fff33"
        android:centerColor="#ffef8a91"
        android:endColor="#ff4a4fef"
         />
    <size
        android:height="100dp"
        android:width="100dp" />
    <stroke
        android:width="5dp"
        android:color="#ffef002f" />
</shape>

\res\drawable\shape_ring.xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="ring"
```

```
    android:thickness="75dp"
    android:innerRadius="100dp">
    <solid android:color="#ffefe763" />
    <stroke
        android:width="5dp"
        android:color="#ffef002f" />
</shape>

\res\drawable\shape_line.xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="line">
    <stroke
        android:width="5dp"
        android:color="#ffef002f"
        android:dashWidth="20dp"
        android:dashGap="10dp"/>
</shape>
```



Shape Ring Android 4.x Bug: Doesn't appear.
Shape Line Android 5.0 bug: Dashes don't take effect.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Shape drawable,
https://developer.android.com/guide/topics/resources/drawable-resource.html#Shape

2. Specifiying in \res\layout\my_fragment_or_activity_layout.xml a UI component with a background property that points to a shape drawable file.

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

tools:context="au.com.softmake.mysecondapp.resourcedemo.ShapeResourcesActivity$Placeho
lderFragment">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin">
```

```
        <TextView
            android:id="@+id/textView_hello"
            android:layout_width="200dp"
            android:layout_height="400dp"
            android:background="@drawable/shape_rectangle"
            android:gravity="center"
            android:text="shape_rectangle.xml" />

        <TextView
            android:id="@+id/textView_side"
            android:layout_width="match_parent"
            android:layout_height="350dp"
            android:background="@drawable/shape_oval"
            android:gravity="center"
            android:text="shape_oval.xml" />

    </LinearLayout>
</ScrollView>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\fragment_shape_resources.xml

3. Inflate this layout file from an Activity or Fragment.

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\resourcedemo\ShapeResourcesActivity.java

Alternative for associating a shape drawable with a UI component:

```
Resources res = getResources();
Drawable shape = res.getDrawable(R.drawable.shape_oval);

TextView tv = (TextView)findViewByID(R.id.textView_side);
tv.setBackground(shape);
```

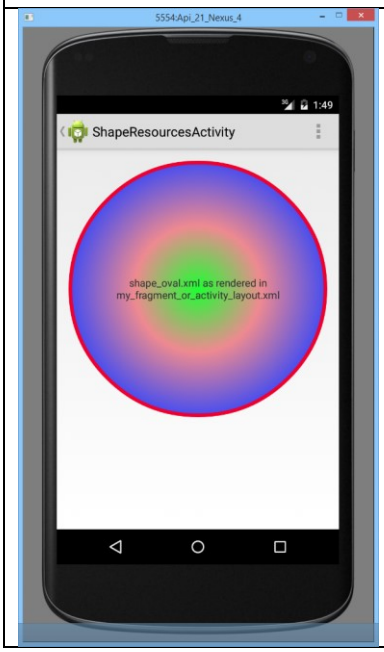
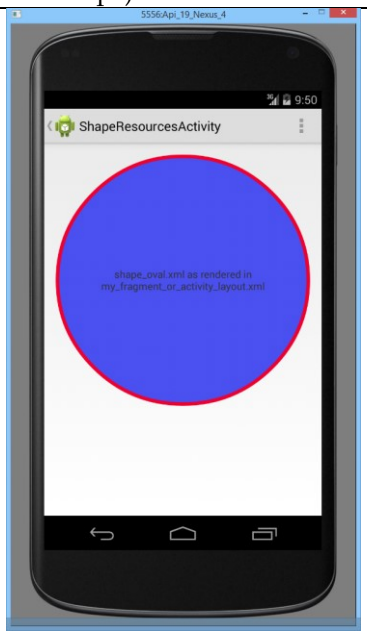
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Drawable, Shape drawable, https://developer.android.com/guide/topics/resources/drawable-resource.html#Shape

To use a gradientRadius in a Shape Drawable (for Android 5.0 and above) : An integer or floating point number appended with either % or %p, such as "14%", "14%p", "14.5%", or "14.5%p". Unappended numbers, such as "14" or "14.5", are not allowed. The "%" suffix (without "p") means the percentage of the shape object itself, in which case width and height must be specified in the <size> element (a child of <shape> in \res\drawable\shape_oval.xml). The "%p" suffix means that percentage of the size of the containing UI component (e.g. a <TextView ... android:background="@drawable/shape_oval" ... /> in res\layout\my_fragment_or_activity_layout.xml)

```
\res\drawable\shape_oval.xml to ...

<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <gradient
        android:type="radial"
        android:gradientRadius="50%"
        android:startColor="#ff1fff33"
        android:centerColor="#ffef8a91"
        android:endColor="#ff4a4fef"
        />
    <size
        android:height="100dp"
        android:width="100dp" />
    <stroke
        android:width="5dp"
        android:color="#ffef002f" />
```

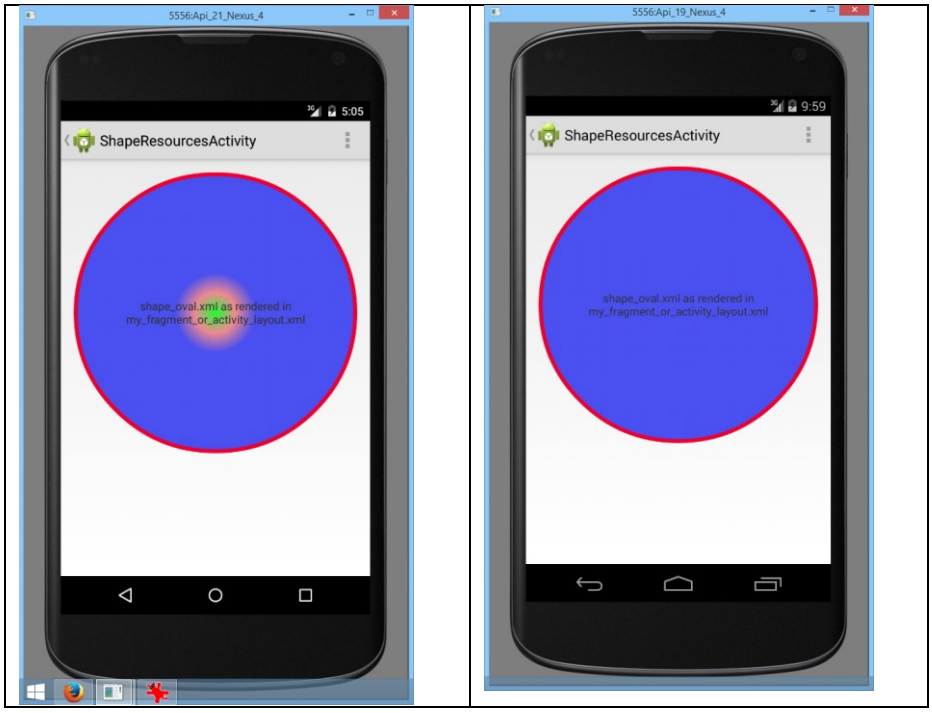
```
</shape>

[Project View] \MyApp\app\src\main\res\layout\my_fragment_or_activity_layout.xml

layout/fragment_shape_resources_02.xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"

tools:context="com.example.myapp.resourcedemo.ShapeResourcesActivity$PlaceholderFragment">

    <TextView
        android:id="@+id/textView_side"
        android:layout_width="match_parent"
        android:layout_height="350dp"
        android:background="@drawable/shape_oval"
        android:gravity="center"
        android:text="shape_oval.xml" />

</LinearLayout>
```

android:gradientRadius="50%p", with size in shape not set (shape size is not relevant when the parent determines the size).

Emulator: Lollipop, API 21, x86, Android 5.0: OK	Emulator: KitKat, API 19, x86, Android 4.4.2: Fail. (End color fills shape).
	

android:gradientRadius="50%", with size in shape set.

Emulator: Lollipop, API 21, x86, Android 5.0: OK.	Emulator: KitKat, API 19, x86, Android 4.4.2: Fail (End color fills shape)



"Issue 71065: GradientDrawable: Radial gradient gradientRadius not working as documented"
https://code.google.com/p/android/issues/detail?can=2&start=0&num=100&q=&colspec=ID%20Type%20Status%20Owner%20Summary%20Stars&groupby=&sort=&id=71065
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\drawable\shape_oval.xml
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\fragment_shape_resources_02.xml
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\resourcedemo\ShapeResourcesActivity.java

Layout

See Layouts and UI Components.

Menu

See Adding a Menu with Action Items (commands)

See ...

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Menu resource, https://developer.android.com/guide/topics/resources/menu-resource.html

Value resources

String

Intro

String resources provide text strings for your application with the power to provide:

- separation of strings and code;
- alternative translations;
- arrays of strings;
- plural handling; and
- string styling and formatting.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, https://developer.android.com/guide/topics/resources/string-resource

Single string

Retrieve string resources in code

```
// \res\values\strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello_world">Hello world!</string>
</resources>

// Get a string resource from your app's Resources
String hello = getResources().getString(R.string.hello_world);

// Or supply a string resource to a method that requires a string
TextView textView = new TextView(this);
textView.setText(R.string.hello_world);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) Devicess, Device compatibility, Support different languages and cultures, Use the rescources in your app https://developer.android.com/training/basics/supporting-devices/languages.html#UseAppResources

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, String https://developer.android.com/guide/topics/resources/string-resource#String

String array resources

String array

```
// res/values/strings.xml

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
    </string-array>
</resources>

Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, String array https://developer.android.com/guide/topics/resources/string-resource#StringArray

Quantity strings (plural handling)

It is often better to avoid the complication of quantity strings by using quantity neutral formulations, for example "Books: 1".

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, Quantity strings (plurals), https://developer.android.com/guide/topics/resources/string-resource#Plurals

Different languages handle plurals (e.g. "1 book" V "n books") differently. This is determined by the arbitrary rules of a language (you can't make algorimithic assumptions). Android supports: zero, one, two, few, many and other.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, Quantity strings (plurals), https://developer.android.com/guide/topics/resources/string-resource#Plurals

Only use quantity strings for language plurals. E.g. Don't use quantity strings for displaying "inbox" v "inbox (12)".

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, Quantity strings (plurals), https://developer.android.com/guide/topics/resources/string-resource#Plurals

Code a quantity string in xml.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals name="numberOfBooks">
        <item quantity="one">%d book</item>
        <item quantity="other">%d books</item>
    </plurals>
</resources>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, Quantity strings (plurals), https://developer.android.com/guide/topics/resources/string-resource#Plurals
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\values\strings_plural.xml

Reference the quantity string in Java (this example uses a fragment).

```
/**
 * A placeholder fragment containing a simple view.
 */
public static class PlaceholderFragment extends Fragment {

    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment value resources, container,
                                         false);

        int bookCount = 3;
        // If your plural string includes string formatting with a number you need
        // to pass a count twice. The first selects the right plural string,
        // the second the actual number to insert in the plural string. Generally
        // these two counts will be the same value.
        String bookCountDescription = getResources()
                .getQuantityString(R.plurals.numberOfBooks, bookCount, bookCount);

        TextView outputTextView = (TextView) rootView.findViewById(
                R.id.textView_output);
        outputTextView.setText(bookCountDescription);

        return rootView;
    }
```

```
}

// Output
3 books
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, Quantity strings (plurals), https://developer.android.com/guide/topics/resources/string-resource#Plurals
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\resourcedemo\ValueResourcesActivity.java

String formatting and styling

Quote escaping and String.format() use

When using quotes (or an "apostrophe") inside a string resource you must escape them either by:

- Surroundiing everything by a quote, using quotes of the other type; or
- Backslash the quote.

```
    <string name="quote_apostrophe_right00">"Surround everything by quotes"</string>
    <string name="quote_apostrophe_right01">"This'll work"</string>
    <string name="quote_apostrophe_right02">This\'ll also work</string>
    <string name="quote_apostrophe_right03">She said \"Very cool\"</string>
    <!--<string name="quote_apostrophe_wrong01">This doesn\'t work</string>
    <string name="quote_apostrophe_wrong02">XML encodings don&apos;t work</string>-->
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, Format and style, https://developer.android.com/guide/topics/resources/string-resource#FormattingAndStyling
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\resourcedemo\ValueResourcesActivity.java

You can use String.format() style formatting strings in your string xml resource. But you must use explicit positional indices (e.g. "1$", "2$", etc).

```
// res\values\strings.xml
<!-- OK -->
<string name="friendly_message">You are %1$s and %2$d years old.</string>

<!-- Fails -->
<string name="friendly_message">You are %s and %d years old.</string>

myString = String.format(getResources().getString(R.string.friendly_message),
                         "great", 25);
outputTextView = (TextView) rootView.findViewById(
        R.id.textView_output03);
outputTextView.setText(myString);

// Output
You are great and 25 years old.
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, Format and style, https://developer.android.com/guide/topics/resources/string-resource#FormattingAndStyling
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\resourcedemo\ValueResourcesActivity.java

Sytling with HTML Markup

You can style text, to underline, bold, or italics it with simple HTML markup. Only <u>, <b>, and <i> is supported. However you must use HTML entity escaping e.g. " <b>" for "<b>" and format the string with Html.fromHtml(text).

```
// \res\values\strings.xml
<string name="html_simple01">There is a &lt;u&gt;cat&lt;/u&gt; on the
&lt;b&gt;couch&lt;/b&gt;, looking &lt;i&gt;wistful&lt;/i&gt;.</string>

// Java
myString = getResources().getString(R.string.html_simple01);
CharSequence htmlCharSequence = Html.fromHtml(myString);
outputTextView = (TextView) rootView.findViewById(R.id.textView_output04);
outputTextView.setText(htmlCharSequence);

// Output
There is a cat on the couch, looking wistful.
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, Format and style, https://developer.android.com/guide/topics/resources/string-resource#FormattingAndStyling
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\resourcedemo\ValueResourcesActivity.java

You can also combine this simple HTML markup with String.fomart() type strings. Be sure to use explicit positional formatting ("1$", "2$", etc).

```
// \res\values\strings.xml
<string name="html_with_positional_formatting">There is a &lt;u&gt;%1$s&lt;/u&gt; on
    the &lt;b&gt;%2$s&lt;/b&gt;.</string>

// Java
myString = String.format(getResources().getString(R.string
                                        .html_with_positional_formatting), "dog",
"floor");
htmlCharSequence = Html.fromHtml(myString);
outputTextView = (TextView) rootView.findViewById(R.id.textView_output05);
outputTextView.setText(htmlCharSequence);

// Output
Three is a dog on the floor.
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, Format and style, https://developer.android.com/guide/topics/resources/string-resource#FormattingAndStyling
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\resourcedemo\ValueResourcesActivity.java

Styling with Spannables

Spannables allow you to format strings as bold, italic, or underline (??) and set color.

(Google 2013, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, Styling with spannables, https://developer.android.com/guide/topics/resources/string-resource#StylingWithSpannables

To implement Spannables use these helper functions in your android library (slightly modified from the official documentation).

```
public class Text {

    /**
     * Returns a CharSequence that concatenates the specified array of CharSequence
```

```
     * objects and then applies a list of zero or more tags to the entire range.
     *
     * @param content an array of character sequences to apply a style to
     * @param tags    the styled span objects to apply to the content
     *                such as android.text.style.StyleSpan
     */
    public static CharSequence apply(CharSequence content, Object... tags) {
        SpannableStringBuilder text = new SpannableStringBuilder();
        openTags(text, tags);
        // Different from the android documentation. An array of CharSequences no
        // longer required.
        text.append(content);
        closeTags(text, tags);
        return text;
    }

    /**
     * Iterates over an array of tags and applies them to the beginning of the specified
     * Spannable object so that future text appended to the text will have the styling
     * applied to it. Do not call this method directly.
     */
    public static void openTags(Spannable text, Object[] tags) {
        for (Object tag : tags) {
            text.setSpan(tag, 0, 0, Spannable.SPAN_MARK_MARK);
        }
    }

    /**
     * "Closes" the specified tags on a Spannable by updating the spans to be
     * endpoint-exclusive so that future text appended to the end will not take
     * on the same styling. Do not call this method directly.
     */
    public static void closeTags(Spannable text, Object[] tags) {
        int len = text.length();
        for (Object tag : tags) {
            if (len > 0) {
                text.setSpan(tag, 0, len, Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
            } else {
                text.removeSpan(tag);
            }
        }
    }

    /**
     * Returns a CharSequence that applies boldface to the concatenation
     * of the specified CharSequence objects.
     */
    public static CharSequence bold(CharSequence content) {
        return apply(content, new StyleSpan(Typeface.BOLD));
    }

    /**
     * Returns a CharSequence that applies italics to the concatenation
     * of the specified CharSequence objects.
     */
    public static CharSequence italic(CharSequence content) {
        return apply(content, new StyleSpan(Typeface.ITALIC));
    }

    /**
     * Returns a CharSequence that applies a foreground color to the
     * concatenation of the specified CharSequence objects.
     */
    public static CharSequence color(int color, CharSequence content) {
        return apply(content, new ForegroundColorSpan(color));
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, Styling with spannables, https://developer.android.com/guide/topics/resources/string-resource#StylingWithSpannables
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\standardapplibrary\src\main\java\au\com\softmake\standardapplibrary\android\Text.java

Client code that calls the helper functions ...

```
SpannableStringBuilder spannableStringBuilder = new SpannableStringBuilder();
spannableStringBuilder.append("Everything is ");

CharSequence myCharSequence = au.com.softmake.standardapplibrary.android
        .Text.bold(
        au.com.softmake.standardapplibrary.android.Text.italic("cool"));
spannableStringBuilder.append(myCharSequence);

spannableStringBuilder.append(" and ");

myCharSequence = au.com.softmake.standardapplibrary.android.Text.color(
        Color.RED, getResources().getString(R.string.groovy));
spannableStringBuilder.append(myCharSequence);

outputTextView = (TextView) rootView.findViewById(R.id.textView_output06);
outputTextView.setText(spannableStringBuilder);

// Output
Everything is cool and groovy
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, String resources, Styling with spannables, https://developer.android.com/guide/topics/resources/string-resource#StylingWithSpannables
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\resourcedemo\ValueResourcesActivity.java

Style

See Styles and Themes.

Bool

A boolean resource.

```
// res/values/bools.xml:
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <bool name="screen_small">true</bool>
    <bool name="adjust_view_bounds">true</bool>
</resources>


// Layout xml
<ImageView
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:src="@drawable/logo"
    android:adjustViewBounds="@bool/adjust_view_bounds" />

// Java
Resources res = getResources();
boolean screenIsSmall = res.getBoolean(R.bool.screen_small);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, More types, Bool https://developer.android.com/guide/topics/resources/more-resources#Bool

Color

Color

```
res/values/colors.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="opaque_red">#f00</color>
   <color name="translucent_red">#80ff0000</color>
```

```
</resources>

// Layout xml
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/translucent_red"
    android:text="Hello"/>

// Java
Resources res = getResources();
int color = res.getColor(R.color.translucent_red);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, More types, Color, https://developer.android.com/guide/topics/resources/more-resources#Color

Color values are always prefixed with a hash (#) followed by Alpha-Red-Green-Blue information in one of the following formats:

```
#RGB
#ARGB
#RRGGBB
#AARRGGBB
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, More types, Color, https://developer.android.com/guide/topics/resources/more-resources#Color

To make a color translucent use an 8 character hex code, where the first two places reference the alpha channel (the transparency specification). The alpha channel is a hex value from 0, totally transparent, to FF (hex)/255 (decimal), totally opaque.

```
In res/values/styles.xml

<color name="actionbarColorCustom">#40e0b8b8</color>

<style name="ActionBarCustom" parent="@style/Widget.AppCompat.ActionBar">
    <item name="android:background">@color/actionbarColorCustom</item>
</style>

Where 40 in "#40e0b8b8" is the hex value for the alpha channel, representing a 25% opaque
(75% transparent) value.

255 * 0.25 = 63.75 ≅ 64

64 Dec = 40 Hex
```

(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/) http://stackoverflow.com/questions/11285961/how-to-make-a-background-transparent-20-in-android

Dimension

A dimension resource.

```
// res/values/dimens.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="textview_height">25dp</dimen>
    <dimen name="textview_width">150dp</dimen>
    <dimen name="ball_radius">30dp</dimen>
    <dimen name="font_size">16sp</dimen>
</resources>

// Layout XML
<TextView
    android:layout_height="@dimen/textview_height"
    android:layout_width="@dimen/textview_width"
```

```
    android:textSize="@dimen/font_size"/>

// Java
Resources res = getResources();
float fontSize = res.getDimension(R.dimen.font_size);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, More types, Dimension, https://developer.android.com/guide/topics/resources/more-resources#Dimension

Dimension values. Only use "dp" (most dimensions) or "sp" (if you ever need to specify font sizes).

Name	Abbreviation	Usage
Denisity independent pixels	**dp**	All units, except for when you want to specify font size.
Scale independent pixels	**sp**	When specifying font sizes.
Points	pt	Never.
Pixels	px	Never.
Millimetres	mm	Never.
Inches	in	Never.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, More types, Dimension, https://developer.android.com/guide/topics/resources/more-resources#Dimension

ID

See also ID Resource Type.

Defining ID rescoure in specially dedicated XML file (rarely needed as you generally define an ID, in-line, at the time of use in a Layout XML).

```
// res/values/ids.xml:
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item type="id" name="button_ok" />
    <item type="id" name="dialog_exit" />
</resources>

// Layout XML
<Button android:id="@id/button ok"
    style="@style/button_style" />

// Java code.
showDialog(R.id.dialog_exit);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, More types, ID, https://developer.android.com/guide/topics/resources/more-resources#Id

Integer

Integer resource.

```
// res/values/integers.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer name="max_speed">75</integer>
    <integer name="min_speed">5</integer>
</resources>

// Layout XML
<TextView
    android:id="@+id/textView_output07"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@integer/max_speed" />

// Java
Resources res = getResources();
int maxSpeed = res.getInteger(R.integer.max_speed);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, More types, Integer, https://developer.android.com/guide/topics/resources/more-resources#Integer

Integer Array

Integer array resource.

```
// res/values/integers.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer-array name="bits">
        <item>4</item>
        <item>8</item>
        <item>16</item>
        <item>32</item>
    </integer-array>
</resources>

// Java
Resources res = getResources();
int[] bits = res.getIntArray(R.array.bits);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, More types, Integer Array, https://developer.android.com/guide/topics/resources/more-resources#IntegerArray

Typed (Resource) Array

A "Typed Array" is an array of other resources (not java types), like drawables, colors, etc. So strings and integers are not the only resources for which there can be arrays.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, More types, Typed array, https://developer.android.com/guide/topics/resources/more-resources#TypedArray

A typed (resource) array is not required to be homogeneous: it can have a mix of resource types in the single array. In that case you've got to be careful when you consume it in Java to use the right get...() method.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, More types, Typed array, https://developer.android.com/guide/topics/resources/more-resources#TypedArray

Type (Resource) Array example.

```
// res/values/arrays.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <array name="icons">
        <item>@drawable/home</item>
        <item>@drawable/settings</item>
        <item>@drawable/logout</item>
    </array>
    <array name="colors">
        <item>#FFFF0000</item>
```

```
        <item>#FF00FF00</item>
        <item>#FF0000FF</item>
    </array>
</resources>

// Java
// Obtain only the first entry in each array.
Resources res = getResources();
TypedArray icons = res.obtainTypedArray(R.array.icons);
Drawable drawable = icons.getDrawable(0);

TypedArray colors = res.obtainTypedArray(R.array.colors);
int color = colors.getColor(0,0);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, More types, Typed array, https://developer.android.com/guide/topics/resources/more-resources#TypedArray

User Interface

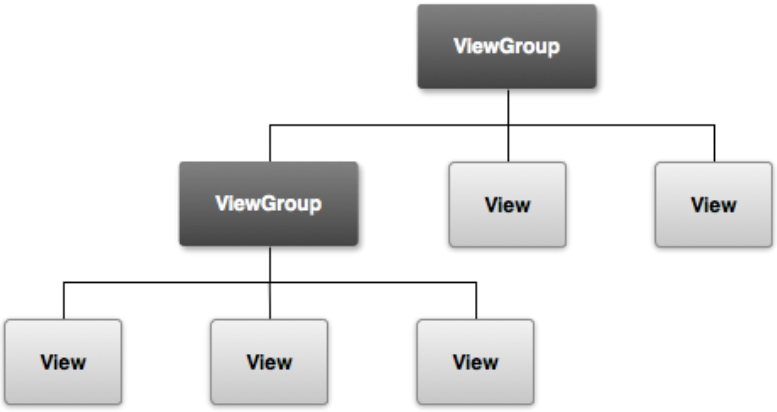
Layouts and UI Components

Common

Basics

A User Interface consists of a hierarchy of:

- ViewGroup objects or "Layouts"; and
- View objects (or "Controls") which are "UI components" but not exclusively so.



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, https://developer.android.com/guide/topics/ui/declaring-layout

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, Build your first app, Build a simple user interface, https://developer.android.com/training/basics/firstapp/building-ui.html

Some UI components are not implemented with a View object. For example, an Action Bar, Dialog, and Status Notifcation are independent objects.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, https://developer.android.com/guide/topics/ui/declaring-layout

A "Layout" references the visual structure of the UI.

Declaring and Instantiating Layouts and UI Components

A layout or UI component is declared either:

- With XML
- In code at runtime;
- A combination of the above.

It is good programming practice to declare layouts using XML where possible. Following web standards this separates presentation from behaviour.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, https://developer.android.com/guide/topics/ui/declaring-layout

Example layout declared with XML.

```
<!-- In res/layout/my_activity.xml -->
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
              xmlns:tools="http://schemas.android.com/tools"
tools:context="au.com.softmake.mysecondapp.resourcedemo.ShapeResourcesActivity$PlaceholderFr
agment"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical">
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout height="wrap content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, https://developer.android.com/guide/topics/ui/declaring-layout

Reference a layout xml and manipulate UI components in code ...

```
//
@Override
protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_display_message);
          // Show the Up button in the action bar.
          setupActionBar();

          // Get the message from the intent
          Intent intent = getIntent();
          String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

          // Reference UI component found in layout xml and manipulated it
          TextView textView = (TextView) findViewById(R.id.display_message_textview);
          textView.setText(message);
}
```

Build the layout (??) and UI components entirely in code ...

```
@Override
protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
//                    setContentView(R.layout.activity_display_message);
          // Show the Up button in the action bar.
          setupActionBar();

          // Get the message from the intent
          Intent intent = getIntent();
          String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

          // Create the UI component in code
          TextView textView = new TextView(this);
          textView.setTextSize(15);
          textView.setText(message);
          textView.setTextColor(Color.parseColor("#2DAD16"));

          // Use setContentView with the UI component.
          setContentView(textView);
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, Build your first app, Start another activity, Display the message, https://developer.android.com/training/basics/firstapp/starting-activity.html#DisplayMessage

You can use <merge><merge/> and <include /> to include one layout xml file in another.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Layout, https://developer.android.com/guide/topics/resources/layout-resource.html

<requestFocus> within a UI component gives it initial focus when the screen is loaded. You can have only one <requestFocus> per layout file.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, App Resources, Resource types, Layout, https://developer.android.com/guide/topics/resources/layout-resource.html

Layouts and UI Components Attributes

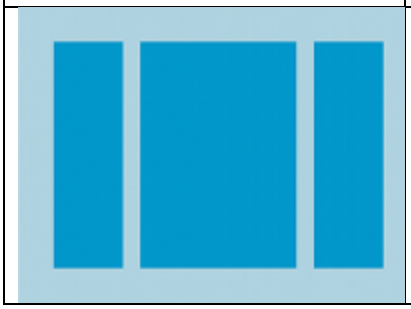
Layouts (ViewGroups) and UI components (Views) have many properties, referenced (if at all) as attributes in XML. Some properties are specific to an object, e.g. A TextView supports the textSize property. Other properties are shared by all Views or ViewGroups, e.g. the id property inherits from the root View class.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, Attributes, https://developer.android.com/guide/topics/ui/declaring-layout.html#attributes

For using the ID property (and attribute), something common to all View objects, see Resource references.

Layouts Overview

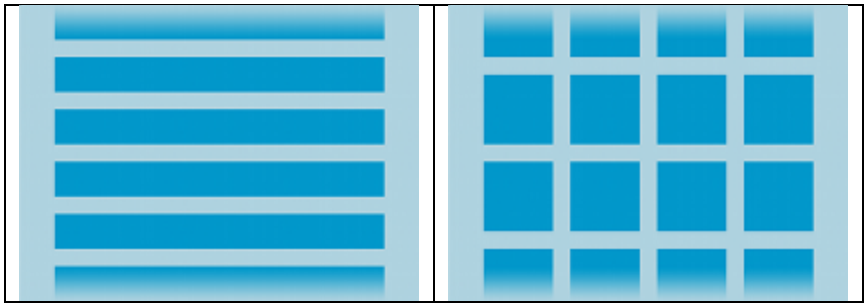
There were three main layouts. Each is implemented as a subclass of ViewGroup (though you most often define a layout in XML).

Linear Layout.	Relative Layout	Web Layout
Organises its children into a single horizontal row or vertical column.	Organises its children relative to each other or relative to the parent layout itself	Displays web pages
		

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, Common Layouts, https://developer.android.com/guide/topics/ui/declaring-layout.html#CommonLayouts

There is also two additional layouts that are built with an adapter. Each is implemented as a subclass of `AdapterView`.

List View	Grid View
Displays a scrolling single column list.	Displays a scrolling grid of columns and rows.



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, Building Layouts with an Adapter, https://developer.android.com/guide/topics/ui/declaring-layout.html#AdapterViews

Specific Layouts

Layout Attributes

General

There is a set of attributes common to all Layout and View objects are those to do with Layout. These are sometimes called "layout parameters". They define how the object (a nested layout or UI component) should size itself with respect to a parent. The layout parameters include `layout_width` and `layout_height`. The values for these are specified:

- With exact measurements (not generally recommended), in pixels or density-independent pixel units (dp).
- `wrap_content`. Meaning the object should size itself as small as possible while still showing its content.
- `match_parent` (formerly `fill_parent`, still supported). Meaning to become as big as the parent minus padding.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    tools:context=".MainActivity" >

    <EditText
        android:id="@+id/edit_message"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
...
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, Attributes, Layout Parameters, https://developer.android.com/guide/topics/ui/declaring-layout.html#layout-params

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, Build your first app, Build a simple user interface, https://developer.android.com/training/basics/firstapp/building-ui.html

Each child object must define layout parameters appropriate for its parent.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, Attributes, Layout Parameters, https://developer.android.com/guide/topics/ui/declaring-layout.html#layout-params

Retrieve the postion of a layout or UI component with `getLeft()`, `getTop()`, `getRight()`, and `getBottom()`.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, Layout Positions, https://developer.android.com/guide/topics/ui/declaring-layout.html#Position

Retrieve width and height measures with:

- `getMeasuredWidth()` and `getMeasuredHeight()`, defining how big a view wants to be within its parent; and
- `getWidth()` and `getHeight()`, defining the actual size on screen at drawing time, after layout.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, {Size, Padding and Magins}, https://developer.android.com/guide/topics/ui/declaring-layout.html#SizePaddingMargins

Padding: Views and ViewGroups support it. Padding offsets content of its object. For example, a left padding of 2 will puh a view's content by 2 pixels to the right of the left edge of the view. Set with `setPadding(int, int, int, int)`. Retrieve with `getPaddingLeft()`, `getPaddingTop()`, `getPaddingRight()` and `getPaddingBottom()`.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, {Size, Padding and Magins}, https://developer.android.com/guide/topics/ui/declaring-layout.html#SizePaddingMargins

LinearLayout Specific

`layout_weight` assigns a portion value among all the objects along a dimenions. This applies only to objects within a LinearLayout.

```
For example, if there are two objects along a horizontal dimension, a textbox and button,
and the textbox is assigned a 4, the button a 1, then the textbox will take up 4/5's of the
horizontal and the button a 1
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, Build your first app, Build a simple user interface, https://developer.android.com/training/basics/firstapp/building-ui.html

Fill the available remaining space with the current control, while allowing for other controls in the same space to take up as much space as they like, by specifying a layout_weight greater than 0 and specifying a layout_width of 0dp.

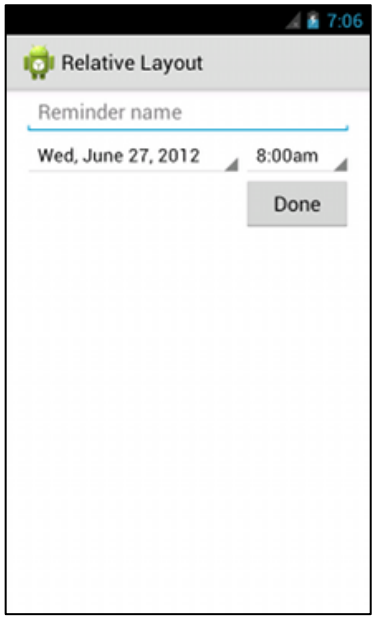
```
<EditText
          android:layout_weight="1"
          android:layout_width="0dp"
          ... />
```

The default weight for all views is 0, so if you specify any weight value greater than 0 to only one view, then that view fills whatever space remains after all views are given the space they require. In order to improve the layout efficiency when you specify the weight, you should change the width of the target object to be zero (0dp).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide) App Basics, Build your first app, Build a simple user interface, https://developer.android.com/training/basics/firstapp/building-ui.html

`Layout_gravity` is supported in LinearLayouts. Values include left, right, bottom, top (and more).

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/appGreen"
    android:orientation="vertical"
    tools:context="au.com.softmake.myfirstapp.LinearLayoutDemoActivity" >

    ...
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/temp_string" />

</LinearLayout>
```



RelativeLayout Specific

In RelativeLayouts:

- Child views can be specified relative to the parent (layout) or sibling child view;
- Are good substitutes where you might be tempted to nest several linear layouts;

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Relative layout, https://developer.android.com/guide/topics/ui/layout/relative.html

RelativeLayout.LayoutParams for Views either specify a child id (for positioning relative to a sibling View) or a boolean (for positioning relative the parent ViewGroup).

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
```

```
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Relative layout, https://developer.android.com/guide/topics/ui/layout/relative.html

ViewGroup Only

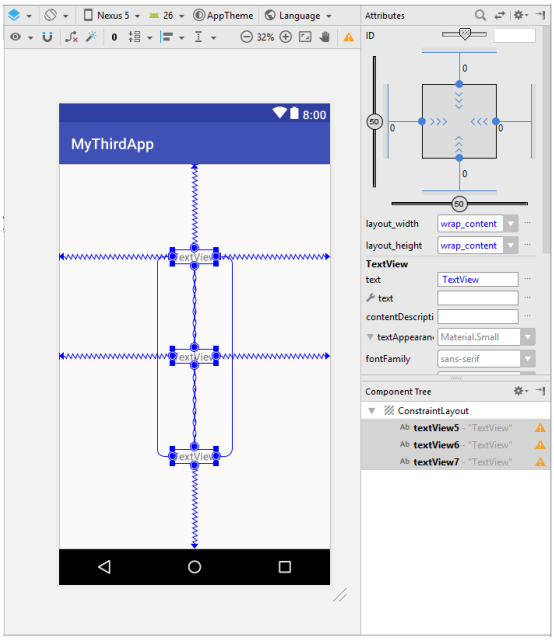
Margins: only ViewGroups support it.

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), ViewGroup.MarginLayoutParams, https://developer.android.com/reference/android/view/ViewGroup.MarginLayoutParams.html

ConstraintLayout

Overview

There is now the Constraint Layout. It is like, but superior to, the Relative Layout.



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, https://developer.android.com/training/constraint-layout/index.html

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_contraint_layout_demo.xml

The easiest way to work with a ConstraintLayout is through the Android Studio UI, rather than setting values in XML.

Start

Enable ConstraintLayouts in your project.

1. In `build.gradle (Project: MyProject)` ensure you have a reference to the google repo.

```
buildscript {

    repositories {
        google() // This is the google maven repository
        ..
    }
}

allprojects {
    repositories {
        google()
        ...
    }
}
```

2. In build.gradle (Module: app) insert the dependency.

```
dependencies {
    ...
    implementation 'com.android.support.constraint:constraint-layout:1.1.0-beta6'
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Add ConstraintLayout to your project, https://developer.android.com/training/constraint-layout/index.html#add-constraintlayout-to-your-project

(Google, n.d., "Android Studio User Guide", https://developer.android.com/studio/intro), Configure your build, Add build depenenencies, Google's Maven repository, https://developer.android.com/studio/build/dependencies.html#google-maven

For referencing the latest version of the constraint library see: Check the latest version.

To start a ContraintLayout you can: create a layout (shell); or convert an existing a layout. See:

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Add ConstraintLayout to your project, https://developer.android.com/training/constraint-layout/index.html#add-constraintlayout-to-your-project

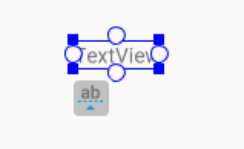
Once you have a ContraintLayout and some componenets on the page you can automatically "Infer Constraints" using the magic button ().

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Automatically create constraints, https://developer.android.com/training/constraint-layout/index.html#use-autoconnect-and-infer-constraints

Creating and deleting

To create contraints:

- Drag one or more views from the palette.
- Click a view to select. Observe square resizing handles (which you generally don't want to use) and circled constraint handles.



- Drag a circular constraint handle to one of three places:
    - The constraint handle of another view in the same plane i.e: a top/bottom handle to top/bottom handle; a side handle to side handle; or baseline to baseline;
    - The edge of a layout; or
    - The edge of a guideline.
- To drag a basline constraint handle, click on the view; click on the baseline icon below the view; click inside the baseline oval, making it green; then drag from that green oval to another green oval baseline in another view.

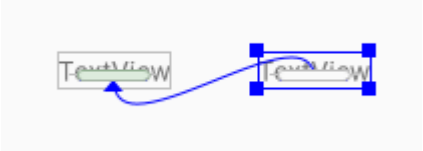
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Add ConstraintLayout to your project, Add or remove a constraint, https://developer.android.com/training/constraint-layout/index.html#add-a-constraint

Contraint creation restrictions:

- Every view must have at least two constraints: one vertical and one horizontal.
- From a view each constraint handle can be used only once. But to a view a constraint handle can accept multiple contraints (from multiple views).
- You can't create a "bi-directional position constraint", where a view links to a constraint handle of another view and from that handle another constraint is created linking back to the originating constraint handle - by dragging. Rather, such bi-directional position contraints are created by "chaining".

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Add ConstraintLayout to your project, Add or remove a constraint, https://developer.android.com/training/constraint-layout/index.html#add-a-constraint

To delete contraints:



- Delete individual contrainst: select the view > hover over a circular constaint handle > observe it is red > click it.
- Delete all contraints of a view: select the view > observe the delete contrain button nearby > click that button.
- Delete all contraints of the layout: click the "clear all contraints" button on the toolbar.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Add ConstraintLayout to your project, Add or remove a constraint, https://developer.android.com/training/constraint-layout/index.html#add-a-constraint

Constraint effects

A constraint effects:

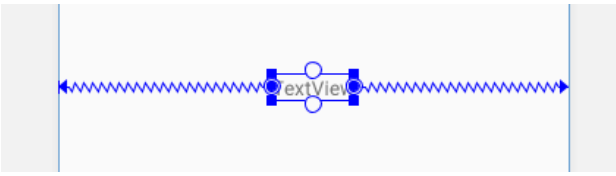
- The distance between the current view object and the contraint target (another view, edge of layout, or guideline) in the horizontal or vertical plain; and/or
- The size of the view.

When manipulating constraint distance you have three basic choices:

- With one contraint in a (horizontal or vertical) plane: set a fixed margin;
- With two contraints in a (horizontal or vertical) plane:
  - Center the view; or
  - Bias the view by setting a percentage value of mid point position in the plane.

Two contraints in same plane

Two contraints in a (horizontal or vertical) plane will turn the constraint line squiggly:



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Add ConstraintLayout to your project, Add or remove a constraint, https://developer.android.com/training/constraint-layout/index.html#add-a-constraint

With two constraints in the same plane you can adjust bias by either:

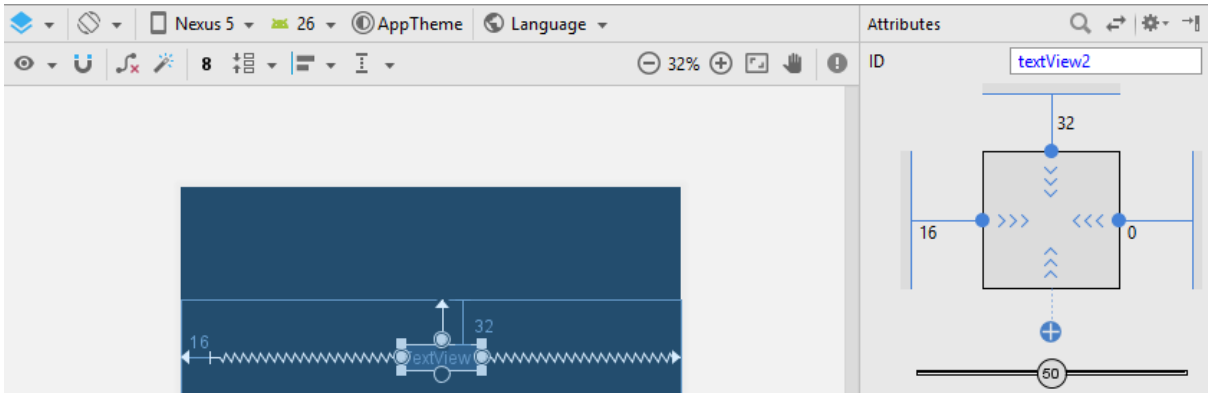
- In the designer window, dragging the view object; or
- In the attributes pane, draggin the constraint node.

(Google 2018, "Android Developer Guides", https://developer.android.com/guide/index.html), "Build a Responsive UI with ConstraintLayout", "Adjust the constraint bias", https://developer.android.com/training/constraint-layout/index.html#adjust-the-constraint-bias

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Adjust the constraint bias, https://developer.android.com/training/constraint-layout/index.html#adjust-the-constraint-bias

Contraint layout margins

Constraints can also be set at a fixed width, alone or in combination with an opposing (squiggly line) constraint. This fixed width is known as a "layout margin". Layout margins can be set to zero.



Relative position ("order")

You can align one view relative to another. E.g., "always to the right of (optionally by such-and-such a layout margin", or "always below (optionally by such-and-such layout margin". This relative positioning does not preserve alignment.

Alignment

To align you create a constraint to the same edge in each view. E.g. Left edge to left edge.

If you want to creaet a centred alignment:

- Create two constraints on each edge in the same plane, e.g. Left to left; and right to right; or
- (In the horizontal plane only) align baselines.

Barriers

A barrier is an invisible line which gets pushed in the relevant plane, the vertical or horizontal, according to the maxium distance of an edge of the enclosed views.

To create a barrier:

1. In build.gradle (Module: app) ensure you are using v1.1.0-beta6 or higher.

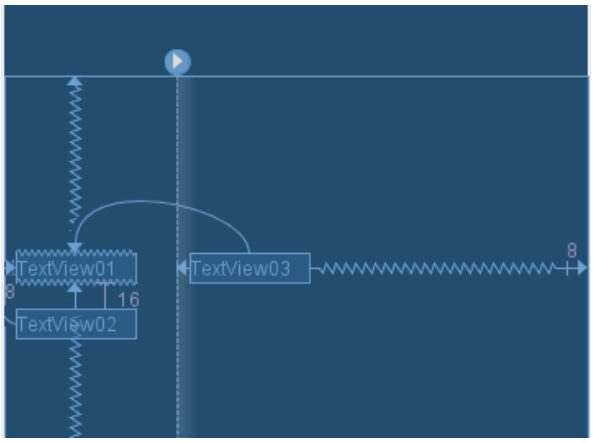
```
dependencies {
    ...
    implementation 'com.android.support.constraint:constraint-layout:1.1.0-beta6'
}
```

2. myapp/res/layout/mylayout.xml > |Design| view:
    a. Guidelines > [Add Vertical barrier] | [Add Horizontal barrier]
    b. Component Tree. Drag at least two views (otherwise a barrier is not relevant) into the barrier.

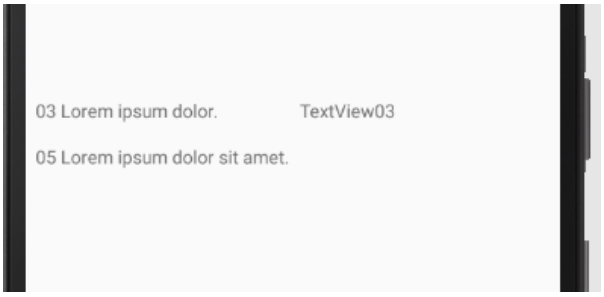
   c. Click on the barrier > Attributes panel > barrierDirection. Set it (e.g. "end", "start", etc).

3. For views *outside* the barrier, these are the views that will be pushed by the barrier, constrain them to the barrier
4. Optionally drag a guideline (of the same plane) into the barrier to set a minimum position for the barrier.

For example, the following ...



... could result in ...

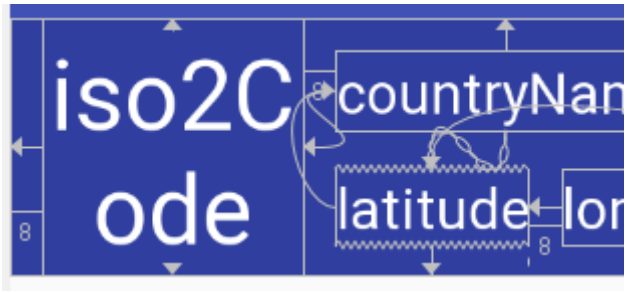


Adjusting view size

A view has the following `layout_[width|height]` sizing modes, in the horizontal and/or vertical plane:

- Fixed;
- Wrap content: the view shrinks to fit contents
- Match contraints. Either:
    - layout_constraintWidth_default = spread (default): expand the view until the constraint is met; or
    - layout_constraintWidth_default = wrap: the view shrinks to fit contents but will not violate contraints.
    - layout_constraintWidth_min. A view's minimum width.
    - layout_constraintWidth_max. A view's maximum width.
- Set size ratio.

If a view was their `layout_width` or `layout_height` set to `match_contraint` then this is represented in the design pane as a squiggley line. For example, the following latitude TextView has its width set to `match_contraint`.



C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\row_for_recycler_view_demo.xml

To set a View's size ratio:

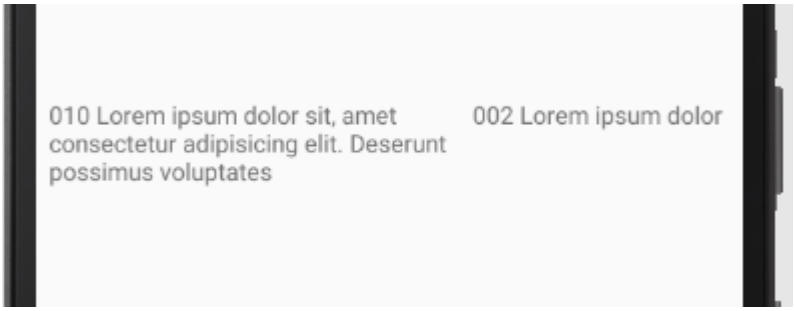
- One or both dimensions must be set to "match_constraint". Then,
- In the Attribute pane, click on the top left hand corner of the view representation to expose the ratio textbox.
- Enter a ratio in the textbox (e.g. "2:1", "16:9", etc).
- Optionally, if both sides are set to "match_contraint", to swap which side the ratio is based on, either:
    - Click the top left hand corner twice (currently this isn't working https://issuetracker.google.com/issues/76456591); or
    - Manually change the order in the text box. E.g. Change "2:1" to "1:2".

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Adjust the view size, Set size a a ration, https://developer.android.com/training/constraint-layout/index.html#ratio

For views that take a variable amount of text, i.e. when being set from code, you probably want to prevent the content from overflowing passed the screen edges through the following settings:

- layout_[width|height]: match_contraint; and
- layout_constraintWidth_default: wrap.

That is, you can achieve ...



... rather than put up with ...

Margins

You can change the default margin from the toolbar. By default, this is represented as an "8".

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Adjust the view margins, https://developer.android.com/training/constraint-layout/index.html#adjust-the-view-margins

Chaining

You can set up bi-directional constraints between views. This is done via chaining commands rather than dragging a second constraint from a view.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Control linear groups with a chain, https://developer.android.com/training/constraint-layout/index.html#constrain-chain

When a chain it setup in either, or both of, the horizontal or vertical plane then, by default, the views position themselves evenly with respect to each other. Then there are several ways in which they position themselves evenly:

- Spread (default): evenly distributed between themselves and edges (or guidelines).
- Spread inside: the first and last view push up against their respective screen edges. The remaining views within are spread evenly.
- Packed: all the views but up against each other in the middle.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Control linear groups with a chain, https://developer.android.com/training/constraint-layout/index.html#constrain-chain

Those positioning modes are selected either:

- In XML view goto the head view (the topmost or leftmost view) and change the `chainStyle` value.

```
app:layout_constraintHorizontal_chainStyle="spread"
```

- In Design views select any one of the Views in the chain and toggle the chain icon that appears below the view.

For the "spread" and "spread inside" positioning modes you can fill remaining space by:

- Setting one or more views to match_contraint in the relevant plane; and
- Assign a layout_contraint[Horizontal|Vertical]_weight to those views. The weight is an integer value that defines the ratio of how much space to take up. E.g. Assigning three views a weight of 4, 2, 2 respectively means the first view takes up half the space and the remaining two take up one quarter earch.

Chaining positioning mode examples:

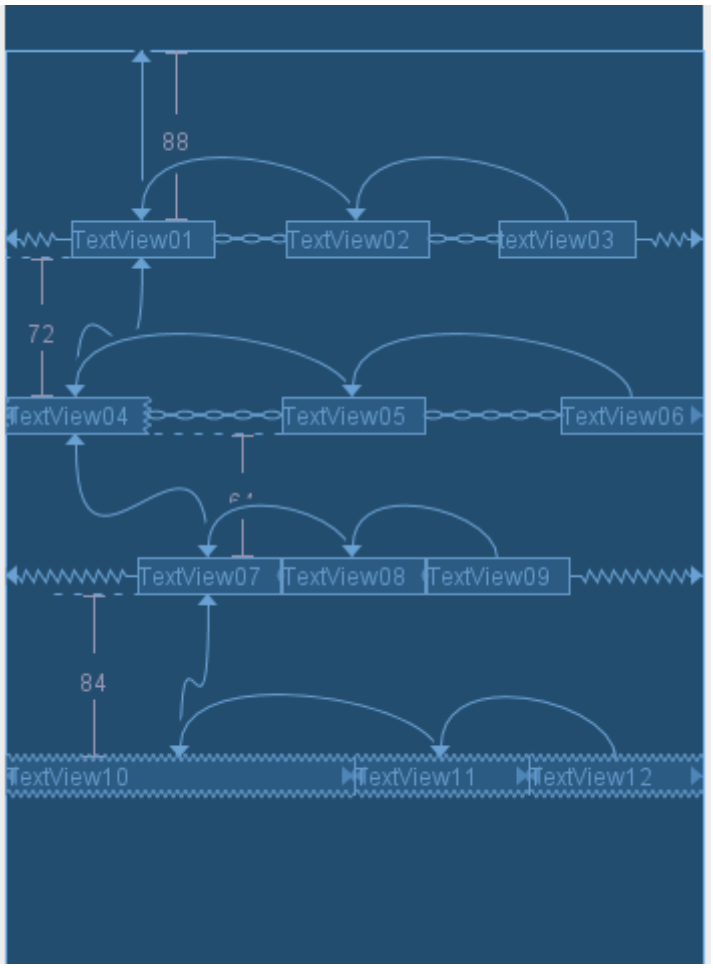


Figure 1 Chaining positioning modes. From the top: spread; spread inside; packed; weighted (with chainStyle="packed")

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_hello_world.xml
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Control linear groups with a chain, https://developer.android.com/training/constraint-layout/index.html#constrain-chain

> Although the orientation of a chain is either vertical or horizontal, using one does not align the views in that direction. So be sure you include other constraints to achieve the proper position for each view in the chain, such as alignment constraints.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Build a Responsive UI with ConstraintLayout, Control linear groups with a chain, https://developer.android.com/training/constraint-layout/index.html#constrain-chain

Adapter Views

Coding AdapterViews (ListView example)

When building an AdapterView you need these elements:

- An Activity that hosts an AdapterView, such as ListActivity or just Activity (for Grids).
- A datasource.
- An AdpaterView subclass, ListView or GridView. This is built in code;
- A layout for each item in the AdapterView. This can be custom or built-in.
- An Adpater subclass, such as ArrayAdapter or CursorAdaptor. This binds the: activity (fix?), layout item and datasource.
- Optionally an item click listener.

The steps to code an AdapterView:

1. Create an activity to host an AdapterView.

```
public class MyListActivity extends ListActivity {
          @Override
          public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
          }
}
```

2. Retrive some data.

```
public class MyListActivity extends ListActivity {
          @Override
          public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

    String[] countries = getResources().getStringArray(R.array.countries_array);
          }
}
```

See Resource references in Java to see how to define a String array in XML.

3. Define a custom layout file for each item in the datasource (skip this step if you want to use a simple inbluit layout file) .

```
<!-- res/layout/list_item_file.xml -->
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android=http://schemas.android.com/apk/res/android
    android:id="@+id/list_item_element"
    android:layout_width="match_parent"
    android:layout_height=" match_parent"
    android:padding="10dp"
    android:textSize="20sp" >
</TextView>
```

4. Instantiate an Adapter, binding the data source to a layout file containing a TextView that defines each item. Use a custom layout file or an inbuilt layout file.

```
public class MyListActivity extends ListActivity {
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		// Get Data Source.
		String[] countries = getResources().getStringArray(
					R.array.countries_array);

		// Bind an adapter, datasource, and a layout that defines each item.
		// The resource reference is to a list item layout file, not the id of
		// the list item element.

		// Referencing a custom layout file, res/layout/list_item_file.xml (as
above)
		ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
					R.layout.list_item_file, countries);

		// Referencing an inbuilt layout file.
//		ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
//					android.R.layout.simple_list_item_1,
countries);

	}
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\ListActivityDemo.java

5. Get the AdaptorView, either by referencing an XML AdapterView layout or by leveraging a ListActivity class.

```
// Reference AdapterView as defined in XML
ListView listView = (ListView) findViewById(R.id.listview);

// Get an AdapterView implicitly associated with the class at runtime
ListView listView = this.getListView();
```

6. Bind the Adapter to the AdapterView.

```
public class ListActivityDemo extends ListActivity {

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

		// Get Data Source.
		String[] countries = getResources().getStringArray(
					R.array.countries_array);

		// Bind an adapter, datasource, and a layout that defines each item.
		// The resource reference is to a list item layout file, not the id of
		// the list item element.

		// Referencing a custom layout file, res/layout/list_item_file.xml
		ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
					R.layout.list_item_file, countries);

		// Referencing an inbuilt layout file.
//		ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
//					android.R.layout.simple_list_item_1,
countries);

		// Get an AdapterView implicitly associated with this ListActivity
class
		// at run time.
		ListView listView = this.getListView();
```

```
                        listView.setAdapter(adapter);
            }
...
}
```

Optionally add an item click listener.

```
public class ListActivityDemo extends ListActivity {

  private OnItemClickListener mItemClickListener = new OnItemClickListener();

            public class OnItemClickListener implements
                                    android.widget.AdapterView.OnItemClickListener {
                        public void onItemClick(AdapterView<?> parent, View view, int
position,
                                                long id) {
                                    String text = String.format(Locale.getDefault(),
                                                            "Item position %d", position);
                                    Toast toast = Toast.makeText(getApplicationContext(),
text,
                                                            Toast.LENGTH_SHORT);
                                    toast.show();
                        }
            }

            @Override
            protected void onCreate(Bundle savedInstanceState) {
   ...
                        ListView listView = this.getListView();
                        listView.setAdapter(adapter);
                        listView.setOnItemClickListener(mItemClickListener);
   ...
```

Coding AdapterView options

You have several options when coding AdapterViews:

- AdpaterView: Choose a ListView or GridView AdpaterView.
- Implicit ListView: If using a ListView, base the Activity that hosts the list view: on ListActivity; or a regular Activity but reference your own ListView Layout (separate from a list item element) {fix: can a list item element be a child of a listview element?}
- Inbuilt or custom item view: Use a custom list or grid item; versus use an inbuilt list or grid item.
- Further item customization.
- Adpater: E.g. ArrayAdpater; SimpleCursorAdpater

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, Building Layouts with an Adapter, Filling an adpater with with data, https://developer.android.com/guide/topics/ui/declaring-layout.html#FillingTheLayout

Further item customization of an ArrayAdapter listview item.

- Override the toString() method for the objects in your array.
- Create an item view that is based on something other than TextView (e.g. ImageView): change the item element in your item.xml layout file; extend the ArrayAdapter class and override getView() to return the type of view you want.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, Building Layouts with an Adapter, Filling an adpater with with data, https://developer.android.com/guide/topics/ui/declaring-layout.html#FillingTheLayout

Adpater Types

Overview

There are serveral Adapter subclasses to use. Two common Adpater subclasses are ArrayAdpater and SimpleCursorAdatper.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Overview, Building Layouts with an Adapter, https://developer.android.com/guide/topics/ui/declaring-layout.html#AdapterViews

ArrayAdapters

ArrayAdpaters are suitable if you have a single value that you wish to output per list item. SimpleCusorAdapters are suitable if you have multiple values that you wish to return per list item.

ListView Coding

You can create a list layout:

- Exclusively in code; or
- Using XML layouts.

For an example of creating a list layout exclusively in code see Adapter Views.

For an example of creating a list layout using xml. You must use the specific id "@android:id/list" (in XML) and reference this as "android.R.id.list" (in Java).

```
res\layout\activity_file_selection.xml
<!-- You must use the specific inbuilt id "@android:id/list" -->
<?xml version="1.0" encoding="utf-8"?>
<ListView
    android:id="@android:id/list"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout height="match parent" />


\java\au\com\softmake\mysecondapp\sharefiles\FileSelectionActivity.java
public class FileSelectionActivity extends Activity {

    private File mAppInternalRegularStorageRootDir = null;
    private File mAppInternalRegularStorageSubDir = null;

    File[] mAppSubDirFiles = null;
    // Filenames corresponding to mAppSubDirFiles
    SortedMap<Integer, String> mAppSubDirFilesMap = new TreeMap<Integer, String>();

    Intent mResultIntent = null;
...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_file_selection);
        // Show the Up button in the action bar.
        setupActionBar();

        mResultIntent = new Intent("au.com.softmake.mysecondapp.ACTION_RETURN_FILE");

        // Get the custom file names.
        mAppInternalRegularStorageRootDir = getFilesDir();
```

```
        mAppInternalRegularStorageSubDir = new File(mAppInternalRegularStorageRootDir,
                                                    "coolFilesDir");
        mAppSubDirFiles = mAppInternalRegularStorageSubDir.listFiles();

        int i = 0;
        for (File file : mAppSubDirFiles) {
            mAppSubDirFilesMap.put(i++, file.getAbsolutePath());
        }
        String[] appSubDirFileNames = mAppSubDirFilesMap.values().toArray(new String[0]);

        // Set the activity's result to null to begin with.
        setResult(Activity.RESULT_CANCELED, null);

        // Create and Adapter from context, simple item layout, dataList
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                                                                android.R.layout
                                                                        .simple_list_item_1,
                                                                appSubDirFileNames);
        // Set the activity's list to the adapter.
        ListView listView = (ListView) findViewById(android.R.id.list);
        listView.setAdapter(adapter);
        listView.setOnItemClickListener(mItemClickListener);
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\sharefiles\FileSelectionActivity.java
(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/) *http://stackoverflow.com/questions/11050817/your-content-must-have-a-listview-whose-id-attribute-is-android-r-id-list*

SimpleCursorAdapters

For a SimpleCursorAdapter example see as follows.

1. Create an activity to host an AdapterView.

```
public class ListActivityCursorDemo extends ListActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                // Show the Up button in the action bar.
                setupActionBar();
 ...
```

2. Define a layout for each item in the datasource.

```
<!-- res/layout/list_item_cursor_file.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="horizontal" >

    <TextView
        android:id="@+id/_id"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="#23af41"
        android:paddingBottom="10dp"
        android:paddingLeft="30dp"
        android:paddingTop="10dp"
        android:text="123" />

    <TextView
        android:id="@+id/display_name"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="5"
        android:background="#f7f6c3"
```

```
        android:paddingBottom="10dp"
        android:paddingLeft="30dp"
        android:paddingTop="10dp"
        android:text="@string/temp_string" />

</LinearLayout>
```

3. Retrive some data.

```
// Get Data Source
Uri contactUri = ContactsContract.Contacts.CONTENT_URI;
String[] cursorColumnIDs = new String[] {
                ContactsContract.Data._ID ,
                ContactsContract.Data.DISPLAY_NAME };
int[] viewColumnIDs = new int[] { R.id._id, R.id.display_name };

Cursor cursor = null;

try {

    // Warning: run this code on a hardware phone. The emulator does
    // not have a list of contacts by default and so will return
    // nothing.
    cursor = getContentResolver().query(contactUri, cursorColumnIDs,
                            null, null, null);

    // If field not available in selected contact then an exception is
    // thrown but is not trappable (this is strange). We therefore
    // continue execution with a "finally" clause an test cursor for
    // null.
} finally {
```

4. Instantiate an Adapter, binding the data source to a layout file containing the TextViews that hold each value returned by the cursor. Map cursor columnIDS to View IDs. Ensure one of the view IDs (or cursor IDs??) is "_ID" otherwise it won't work.

```
} finally {
  if (cursor != null && cursor.getCount() > 0) {

  SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
      R.layout.list_item_cursor_file, cursor,
      cursorColumnIDs, viewColumnIDs, 0);
```

5. Optionally format the output returned by the cursor before insertion into the view item.

```
// Format column data
adapter.setViewBinder(new ViewBinder() {

  public boolean setViewValue(View view, Cursor cursor,
      int columnIndex) {
    // Cast the general view to a TextView
    TextView textView = (TextView) view;

    switch (columnIndex) {
    case 0:
      // Force string integer to 3 digits. E.g. "7" to
      // "007".
      textView.setText(String.format("%03d",Integer.valueOf(cursor.getString(0))));
      return true;
    default:
      // Let it be handled natively
      return false;
    }
  }
});
```

6. Bind the Adapter to the AdapterView.

```
ListView listView = getListView();
listView.setAdapter(adapt
```

7. Handle the cursor returning no records.

```
} else {
  Context context = getApplicationContext();
  int duration = Toast.LENGTH_SHORT;
  Toast toast = Toast.makeText(context,
      "No data returned into cursor", duration);
  toast.show();
}
```

Full example ...

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\Training01\MyFirstApp\src\au\com\softmake\myfirstapp\ListActivityCursorDemo.java

GridView Example

When building a GridView AdptaerView you need this:

- A set of images in /res/drawable
- A regular Acitivity to host the GridView.
- A GridView Layout file res/layout/my_grid_view.xml
- An ImageApdater which creates an returns an ImageView that inherits from BaseApdater.
- Instantiaton of a GridView in code and setting the ImageAdpater.
- Optionally add an item click listener.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Create dynamic lists with RecyclerView, https://developer.android.com/guide/topics/ui/layout/recyclerview [Formerly http://developer.android.com/guide/topics/ui/layout/gridview.html]

Code the GridView as follows.

A regular Acitivity to host the GridView.

```
public class GridViewDemo extends ActionBarActivity {

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

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\GridViewDemo.java

A GridView Layout file res/layout/my_grid_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnWidth="90dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="20dp"
    android:horizontalSpacing="20dp"
    android:stretchMode="columnWidth"
    android:gravity="center"
/>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\GridViewDemo.java

An ImageApdater which creates an returns an ImageView that inherits from BaseApdater.

```
package au.com.softmake.myfirstapp;

import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;

public class ImageAdpater extends BaseAdapter {

           private Context mContext;
           private Integer[] mThumbIDs = {
             R.drawable.sample_2, R.drawable.sample_3,
             R.drawable.sample_4, R.drawable.sample_5,
             R.drawable.sample_6, R.drawable.sample_7,
             R.drawable.sample_0, R.drawable.sample_1,
             R.drawable.sample_2, R.drawable.sample_3,
             R.drawable.sample_4, R.drawable.sample_5,
             R.drawable.sample_6, R.drawable.sample_7,
             R.drawable.sample_0, R.drawable.sample_1,
             R.drawable.sample_2, R.drawable.sample_3,
             R.drawable.sample_4, R.drawable.sample_5,
             R.drawable.sample_6, R.drawable.sample_7
    };


           public ImageAdpater(Context c) {
                      mContext = c;
           }

           @Override
           public int getCount() {
                      return mThumbIDs.length;
           }

           @Override
           public Object getItem(int position) {
                      return null;
           }

           @Override
           public long getItemId(int position) {
                      return 0;
           }

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

                      // If it is not recycled, initialize some attributes
                      if(convertView == null) {
                                 imageView = new ImageView(mContext);
                                 imageView.setLayoutParams(new GridView.LayoutParams(120,
120));
                                 imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
                                 imageView.setPadding(10, 10, 10, 10);
                      } else {
                                 imageView = (ImageView) convertView;
                      }

                      imageView.setImageResource(mThumbIDs[position]);

                      return imageView;
           }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\GridViewDemo.java

Instantiaton of a GridView in code and setting the ImageAdpater.

```
public class GridViewDemo extends ActionBarActivity {

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

		setContentView(R.layout.activity_grid_view_demo);
		GridView gridView = (GridView) findViewById(R.id.gridview);
		gridView.setAdapter(new ImageAdpater(this));
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\GridViewDemo.java

Optionally add an item click listener.

```
public class GridViewDemo extends ActionBarActivity {

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

		setContentView(R.layout.activity_grid_view_demo);
		GridView gridView = (GridView) findViewById(R.id.gridview);
		gridView.setAdapter(new ImageAdpater(this));

		gridView.setOnItemClickListener(new OnItemClickListener() {
		    public void onItemClick(AdapterView<?> parent, View v, int
position, long id) {
		Toast.makeText(GridViewDemo.this, "" + position,
Toast.LENGTH_SHORT).show();
		}
		});
	}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\GridViewDemo.java

Loading a SimpleCursorAdapter with a CursorLoader

See also Loaders.

However, using a CursorLoader is "the standard way to query a Cursor" given that this loads it asynchronously (thereby avoiding a blockage of your main thread). After the query returns you can then load your cursor into your Adapter.

CursorLoader is available from Android 3.0 (API Level 11) but made backwards compatible to Android 1.6 with the support library.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Create dynamic lists with RecyclerView, https://developer.android.com/guide/topics/ui/layout/recyclerview [Formerly http://developer.android.com/guide/topics/ui/layout/listview.html]

SimpleCursorAdapter with a CursorLoader (and Contacts) ....

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Create dynamic lists with RecyclerView, https://developer.android.com/guide/topics/ui/layout/recyclerview [Formerly http://developer.android.com/guide/topics/ui/layout/listview.html]

RecyclerView

Overview

The `RecyclerView` layout is "a more advanced and flexible version of `ListView`".

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Create dynamic lists with RecyclerView, https://developer.android.com/guide/topics/ui/layout/recyclerview [Formerly http://developer.android.com/guide/topics/ui/layout/listview.html]

RecyclerView parts:

- Overall container: `RecyclerView`.
- `RecyclerView` filled with views from a *layout manager* (e.g. `LinearLayoutManager` or `GridLayoutManager`).
- Individual views in the list are *view holders* (instances of your extension of `Recycler.ViewHolder`).
- View holders are managed by an *adapter*, here given by your extension of `RecylcerView.Adapter`. The adapter binds view holders to data by calling the adapter's method `onBindViewHolder()`.
- "When the displayed items change, you can notify the adapter by calling an appropriate `RecyclerView.Adapter.notify…()` method. The adapter's built-in code then rebinds just the affected items".

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Create dynamic lists with RecyclerView, https://developer.android.com/guide/topics/ui/layout/recyclerview [Formerly http://developer.android.com/guide/topics/ui/layout/listview.html]

RecyclerView Coding

Add the support library.

1. Open `build.grandle (Module: app).`
2. Add to dependencies ...

```
dependencies {
    implementation 'com.android.support:recyclerview-v7:27.1.1'
}
```

3. Update to the latest. See Libraries > Check the latest version above.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Create dynamic lists with RecyclerView, https://developer.android.com/guide/topics/ui/layout/recyclerview [Formerly http://developer.android.com/guide/topics/ui/layout/listview.html]

Add RecyclerView as root layout.

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

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Create dynamic lists with RecyclerView, https://developer.android.com/guide/topics/ui/layout/recyclerview [Formerly http://developer.android.com/guide/topics/ui/layout/listview.html]

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_recycler_view_demo.xml

If you are going to display multiple values per item you'll probably want to create an item data structure, as a class.

```
public class Country {
    public String name = "";
    public String iso2Code = "";
    public String incomeLevelIso2Code = "";
    // Using floats just for kicks. In production we'd use doubles given that
    // Mobile CPU architectures are moving toward 64 bits and in that context
    // there's no speed benefit in using 32bit datatypes.
    public float longitude = 0f;
    public float latitude = 0f;
    public boolean isRegion = false;

    public Country(String name, String iso2Code, String incomeLevelIso2Code, float
longitude, float latitude) {
        this.name = name;
        this.iso2Code = iso2Code;
        this.incomeLevelIso2Code = incomeLevelIso2Code;
        this.longitude = longitude;
        this.latitude = latitude;
    }

    public Country(String name, String iso2Code, String incomeLevelIso2Code, float
longitude, float latitude, boolean isRegion) {
        this(name, iso2Code, incomeLevelIso2Code, longitude, latitude);
        this.isRegion = isRegion;
    }

    @Override
    public String toString() {
        String result = "";
        String format = "%s: %s%n";
        result += String.format(format, "name", this.name);
        result += String.format(format, "iso2Code", this.iso2Code);
        result += String.format(format, "incomeLevelIso2Code", this.incomeLevelIso2Code);
        result += String.format(format, "longitude", this.longitude);
        result += String.format(format, "latitude", this.latitude);
        return result;
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\recyclerviewdemo\Country.java

Create a layout file that represents a row for an individual item.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
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="wrap_content"
    android:background="@drawable/borders"
    tools:layout_editor_absoluteY="81dp">

    <TextView
        android:id="@+id/iso2Code_textView"
        android:layout_width="65dp"
        android:layout_height="64dp"
        android:layout_marginLeft="8dp"
        android:layout_marginStart="8dp"
        android:gravity="center"
```

```
        android:text="iso2Code"
        android:textColor="@android:color/white"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        android:id="@+id/countryName_textView"
        style="@style/MtaTextView"
        android:layout_width="wrap_content"
        android:layout_height="20dp"
        android:layout_marginLeft="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="countryName"
        app:layout_constraintBottom_toTopOf="@+id/latitude_textView"
        app:layout_constraintStart_toEndOf="@+id/iso2Code_textView"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_chainStyle="spread_inside" />

    <TextView
        android:id="@+id/latitude_textView"
        style="@style/MtaTextView"
        android:layout width="100dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginTop="8dp"
        android:text="latitude"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout constraintStart toStartOf="@+id/countryName textView"
        app:layout_constraintTop_toBottomOf="@+id/countryName_textView" />

    <TextView
        android:id="@+id/longitude_textView"
        style="@style/MtaTextView"
        android:layout width="100dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="8dp"
        android:layout_marginStart="8dp"
        android:text="longitude"
        app:layout_constraintStart_toEndOf="@+id/latitude_textView"
        app:layout_constraintTop_toTopOf="@+id/latitude_textView" />

</android.support.constraint.ConstraintLayout>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\plain_row_for_recycler_view_demo.xml

Create a custom Adapter to associate your dataset item's values with fields in your row item layout. Do this by:

- Extending RecyclerView.Adapter.
- Defining a custom ViewHolder.
- Accepting as a dataset to the contructor your custom type (e.g. List<Country>).
- Implement the remaining RecyclerView.Adapter methods.

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.MyViewHolder> {
    private List<Country> mDataset;

    // Provide a reference to the views for each data item
    // Complex data items may need more than one view per item, and
    // you provide access to all the views for a data item in a view holder.
    public static class MyViewHolder extends RecyclerView.ViewHolder {
        // Each data item is just a string in this case.
        public TextView mTextView_countryName;
        public TextView mTextView_iso2Code;
        public TextView mTextView_latitude;
        public TextView mTextView_longitude;

        public MyViewHolder(View v) {
```

```
            super(v);
            mTextView_countryName = v.findViewById(R.id.countryName_textView);
            mTextView_iso2Code = v.findViewById(R.id.iso2Code_textView);
            mTextView_latitude = v.findViewById(R.id.latitude_textView);
            mTextView_longitude = v.findViewById(R.id.longitude_textView);
        }
    }

    // Provide a suitable constructor (depends on the kind of dataset)
    public MyAdapter(List<Country>  myDataset) {
        mDataset = myDataset;
    }

    // Create new views (invoked by the layout manager)
    @NonNull @Override
    public MyAdapter.MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {

        int layoutResourceID = 0;

        layoutResourceID = R.layout.plain_row_for_recycler_view_demo;
//        layoutResourceID = R.layout.card_row_for_recycler_view_demo;

        View v = (View) LayoutInflater.from(parent.getContext()).inflate(layoutResourceID,
parent, false);

        MyAdapter.MyViewHolder vh = new MyAdapter.MyViewHolder(v);
        return vh;
    }

    @Override
    public void onBindViewHolder(@NonNull MyViewHolder viewHolder, int position) {
        String countryName = mDataset.get(position).name;
        String iso2Code = mDataset.get(position).iso2Code;
        String latitude = "lat: " + Float.toString(mDataset.get(position).latitude);
        String longitude = "long: " + Float.toString(mDataset.get(position).longitude);

        viewHolder.mTextView countryName.setText(countryName);
        viewHolder.mTextView_iso2Code.setText(iso2Code);
        viewHolder.mTextView_latitude.setText(latitude);
        viewHolder.mTextView_longitude.setText(longitude);
    }

    @Override
    public int getItemCount() {
        return mDataset.size();
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\recyclerviewdemo\MyAdapter.java

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Create dynamic lists with RecyclerView, Implementing your adapter and view holder, https://developer.android.com/guide/topics/ui/layout/recyclerview#implement-adapter [Formerly https://developer.android.com/guide/topics/ui/layout/recyclerview#Adapter]

Bring it all together in your Display Activity:

- Set your overall recycler view layout.

```
setContentView(R.layout.activity_recycler_view_demo);
```

- Get a reference to your overall Recycler View layout.

```
mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);
```

- setHasFixedSize to true to improve performance if you know that changes in content do not change the layout size of the RecyclerView.

```
mRecyclerView.setHasFixedSize(true);
```

- Get and Set a layout manager.

```
// Get and Set a layout manager.
// Initialise Layout Manager in this case with a LinearLayoutManager
mLayoutManager = new LinearLayoutManager(this);
// Other layout kinds:
//  mLayoutManager = new GridLayoutManager(this, 2);
//  mLayoutManager = new StaggeredGridLayoutManager(2,
StaggeredGridLayoutManager.HORIZONTAL);

mRecyclerView.setLayoutManager(mLayoutManager);
```

- Get a datset

```
// Get a dataset.
// Get your dataset, if the dataset is a simple array  (or, say, a List) of values.
// String[] myDataset = getResources().getStringArray(R.array.countries_array);

// Get your dataset, if the dataset has multiple values per item.
// The type of your dataset, here List<Country>, must match the type defined by your
Adapter.
CountryXmlImporter countryXMLImporter = new CountryXmlImporter();
InputStream countriesXmlInputStream =
getResources().openRawResource(R.raw.country_data_from_world_bank);
List<Country> myDataset = null;
try {
    myDataset = countryXMLImporter.parseXml(countriesXmlInputStream);
} catch (IOException e) {
    e.printStackTrace();
}
```

- Initialise your Adapter with your Dataset.

```
mAdapter = new MyAdapter(myDataset);
```

- Set the RecyclerView to use your Adapter.

```
mRecyclerView.setAdapter(mAdapter);
```

- All together ....

```
public class RecyclerViewDemoActivity extends Activity {
    private RecyclerView mRecyclerView;
    private RecyclerView.LayoutManager mLayoutManager;
    private RecyclerView.Adapter mAdapter;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Set your overall Recycler View Layout.
        setContentView(R.layout.activity_recycler_view_demo);

        // Get a reference to your overall Recycler View layout.
        mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);

        // setHasFixedSize to true to improve performance if you know that changes
        // in content do not change the layout size of the RecyclerView.
        mRecyclerView.setHasFixedSize(true);

        // Get and Set a layout manager.
        // Initialise Layout Manager in this case with a LinearLayoutManager
        mLayoutManager = new LinearLayoutManager(this);
        // Other layout kinds:
```

```
        //  mLayoutManager = new GridLayoutManager(this, 2);
        //  mLayoutManager = new StaggeredGridLayoutManager(2,
StaggeredGridLayoutManager.HORIZONTAL);

        mRecyclerView.setLayoutManager(mLayoutManager);

        // Get a dataset.
        // Get you dataset, if the dataset is a simple array  (or, say, a List) of
values.
        // String[] myDataset =
getResources().getStringArray(R.array.countries_array);

        // Get your dataset, if the dataset has multiple values per item.
        // The type of your dataset, here List<Country>, must match the type defined
by your Adapter.
        CountryXmlImporter countryXMLImporter = new CountryXmlImporter();
        InputStream countriesXmlInputStream =
getResources().openRawResource(R.raw.country_data_from_world_bank);
        List<Country> myDataset = null;
        try {
            myDataset = countryXMLImporter.parseXml(countriesXmlInputStream);
        } catch (IOException e) {
            e.printStackTrace();
        }

        // Initialise your Adapter with your Dataset
        mAdapter = new MyAdapter(myDataset);

        // Set the RecyclerView to use your Adapter.
        mRecyclerView.setAdapter(mAdapter);
    }
}
```

- For details on how to parse an XML file, to fetch it into your dataset, see XML parsing.

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\recyclerviewdemo\RecyclerViewDemoActivity.java

The result:

Recycler View Further Customizations

Todo: Add item animations.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Advanced RecyclerView customization, https://developer.android.com/guide/topics/ui/layout/recyclerview-custom#animations

*(Szabo 2018, *Android Swipe Collapse Animation (Java)*, https://github.com/david-szabo97/Android-Swipe-Collapse-Animation) https://github.com/david-szabo97/Android-Swipe-Collapse-Animation & https://messedcode.com/dave/posts/android-java-swipe-collapse-animation*

Todo: Enable list-item selection.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Advanced RecyclerView customization, Enable list-item selection https://developer.android.com/guide/topics/ui/layout/recyclerview-custom#select

CardView

Instead of having a plain row in a RecycleView you can use a CardView as the row, to hold repeating items. CardView could also be used a single card, it's just a kind of layout to show card effects, but that would be an unlikely use.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Create a Card-Based Layout, https://developer.android.com/guide/topics/ui/layout/cardview

CardView Coding:

1. Add the library dependency to your project's main module. In build.gradle (Module: app).

```
dependencies {
    implementation 'com.android.support:cardview-v7:27.1.1'
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Create a Card-Based Layout, https://developer.android.com/guide/topics/ui/layout/cardview

2. Create a CardView Layout to serve as your RecyclerView row.

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

    <android.support.v7.widget.CardView
xmlns:card_view="http://schemas.android.com/apk/res-auto"
        android:id="@+id/country_card_view"
        android:layout_width="273dp"
        android:layout_height="144dp"
        android:layout_gravity="center"
        android:layout_marginBottom="8sp"
        android:layout_marginTop="8sp"
        android:theme="@style/AppTheme"
        card_view:cardBackgroundColor="@color/cardview_light_background"
        card_view:cardCornerRadius="16dp"
        card_view:cardElevation="8dp">

        <android.support.constraint.ConstraintLayout
            android:id="@+id/constraintLayout"
            android:layout_width="match_parent"
            android:layout_height="151dp">

            <TextView
                android:id="@+id/iso2Code_textView"
```

```
                android:layout_width="wrap_content"
                android:layout_height="0dp"
                android:layout_marginEnd="8dp"
                android:layout_marginLeft="8dp"
                android:layout_marginRight="8dp"
                android:layout_marginStart="8dp"
                android:layout_marginTop="8dp"
                android:text="XX"
                android:textSize="30sp"
                card_view:layout_constraintEnd_toEndOf="parent"
                card_view:layout_constraintHorizontal_bias="0.104"
                card_view:layout_constraintStart_toStartOf="parent"
                card_view:layout_constraintTop_toTopOf="parent" />

            <TextView
                android:id="@+id/countryName_textView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginEnd="8dp"
                android:layout_marginLeft="8dp"
                android:layout_marginRight="8dp"
                android:layout_marginStart="8dp"
                android:text="TextView"

card_view:layout_constraintBaseline_toBaselineOf="@+id/iso2Code_textView"
                card view:layout constraintEnd toEndOf="parent"
                card_view:layout_constraintHorizontal_bias="0.061"
                card_view:layout_constraintStart_toEndOf="@+id/iso2Code_textView" />

            <TextView
                android:id="@+id/longitude_textView"
                android:layout width="wrap content"
                android:layout_height="wrap_content"
                android:layout_marginBottom="8dp"
                android:text="longitude"
                card_view:layout_constraintBottom_toBottomOf="parent"
                card_view:layout_constraintStart_toStartOf="@+id/latitude_textView"
                card view:layout constraintTop toBottomOf="@+id/latitude textView" />

            <TextView
                android:id="@+id/latitude_textView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="latitude"
                card_view:layout_constraintBottom_toTopOf="@+id/longitude_textView"
                card_view:layout_constraintStart_toStartOf="@+id/countryName_textView"
                card_view:layout_constraintTop_toBottomOf="@+id/countryName_textView"
/>
        </android.support.constraint.ConstraintLayout>

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

</LinearLayout>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\card_row_for_recycler_view_demo.xml
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Create a Card-Based Layout, https://developer.android.com/guide/topics/ui/layout/cardview

3. Optionally, set some of the CardView properties.

```
//  In res\layout\card_row_for_recycler_view_demo.xml
<android.support.v7.widget.CardView
xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/country_card_view"
    android:layout_width="273dp"
    android:layout_height="144dp"
    android:layout_gravity="center"
    android:layout_marginBottom="8sp"
    android:layout_marginTop="8sp"
    android:theme="@style/AppTheme"
    card_view:cardBackgroundColor="@color/cardview_light_background"
    card_view:cardCornerRadius="16dp"
```

```
    card_view:cardElevation="8dp">

// In java code
CardView.setRadius
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\card_row_for_recycler_view_demo.xml
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User inteface, Layouts, Create a Card-Based Layout, https://developer.android.com/guide/topics/ui/layout/cardview

4. Adjust your RecyclerView's Custom Adapter to inflate from the card.

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.MyViewHolder> {
    ...

    // Create new views (invoked by the layout manager)
    @NonNull @Override
    public MyAdapter.MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {

        int layoutResourceID = 0;

//        layoutResourceID = R.layout.plain_row_for_recycler_view_demo;
        layoutResourceID = R.layout.card_row_for_recycler_view_demo;

        View v = (View)
LayoutInflater.from(parent.getContext()).inflate(layoutResourceID, parent, false);

        MyAdapter.MyViewHolder vh = new MyAdapter.MyViewHolder(v);
        return vh;
    }
...
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\recyclerviewdemo\MyAdapter.java

CoordinatorLayout

Intro

The CoordinatorLayout provides the following functionality:

- Works with a Snackbar and FloatingActionButton (fab) to move the fab out of the way when the Snackbar appears. See Pop-up messages (SnackBar)
- Works with a Toolbar and AppBarLayout to allow your toolbar and other views (e.g. Tabs provided by the TabLayout) to react to scroll events.

(Google, n.d., "Android Developers Blog", https://android-developers.googleblog.com/), 2015 May 29, Android Design Support Library, https://android-developers.googleblog.com/2015/05/android-design-support-library.html

The CoodinatorLayout is available from ...

```
implementation 'androidx.coordinatorlayout:coordinatorlayout:1.0.0'
```

(Google n.d., "Android API Reference > Andoid Support Library", Accessed 2021-02-19. https://developer.android.com/reference/android/support/classes), CoordinatorLayout, https://developer.android.com/reference/android/support/design/widget/CoordinatorLayout

... With Snackbar and Floating Action Button

Example CoordinatorLayout with Snackbar and Floating Action Button, to move the fab out of the way when the Snackbar appears.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.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"
    tools:context=".SnackbarDemoActivity"
    android:id="@+id/myCoordinatorLayout">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.ToolbarTheme">

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

    </com.google.android.material.appbar.AppBarLayout>

    <!--<include layout="@layout/content_snackbar_demo" />-->

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior">

        <Button
            android:id="@+id/button_fixed_go"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginLeft="8dp"
            android:layout marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginRight="8dp"
            android:layout_marginBottom="8dp"
            android:text="Fixed Go"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.303"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.266" />

    </androidx.constraintlayout.widget.ConstraintLayout>

    <com.google.android.material.floatingactionbutton.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"
        app:srcCompat="@android:drawable/ic_media_play" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_snackbar_demo.xml
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\SnackbarDemoActivity.java

... (too complex) with Collapsing toolbars (AppBarLayout For ScrollEvents)

The AppBarLayout allows a child Toolbar (and other views like a TabLayout) to react to scroll events in a AppBarLayout sibling view (e.g. a RecyclerView) marked with `app:layout_behavior="@string/appbar_scrolling_view_behavior"` (which maps to

com.google.android.material.appbar.AppBarLayout$ScrollingViewBehavior).

(Google, n.d., "Android Developers Blog", https://android-developers.googleblog.com/), 2015 May 29, Android Design Support Library, https://android-developers.googleblog.com/2015/05/android-design-support-library.html

The kind of Toolbar (or other view) reactions are set by androidx.appcompat.widget.Toolbar's app:layout_scrollFlags

```
<androidx.appcompat.widget.Toolbar
    android:id="@+id/my_toolbar"
    android:layout_width="match_parent"
    android:minHeight="?attr/actionBarSize"
    android:layout_height="@dimen/nav_header_height"
    app:theme="@style/AppTheme.ToolbarTheme"
    app:popupTheme="@style/AppTheme.PopupTheme"
    app:layout_scrollFlags="scroll|enterAlwaysCollapsed|exitUntilCollapsed" />
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_recycler_view_demo.xml

See: AndroidReferenceAdjunct\AppBarLayout-ScrollEffects.xhtml

Toolbar app:layout_scrollFlags settings:

- [none]
- scroll
- scroll|enterAlways
- scroll|enterAlways|enterAlwaysCollapsed
- scroll|enterAlways|enterAlwaysCollapsed|exitUntilCollapsed

app:layout_scrollFlags setting	Toolbar Appears on down drag when	Toolbar disappears on up drag when	Buggy
scroll	At top	At top (Anywhere not applicable)	
scroll|enterAlways	Anywhere	Anywhere	
(Not different from sroll) scroll|enterAlwaysCollapsed (with minHeight and layout_height set)	At top	At top (Anywhere not applicable)	
scroll| enterAlways| enterAlwaysCollapsed (with minHeight and layout_height set)	Anywhere: appears collapsed (its min height) At Top: Full height	Anywhere	
scroll| enterAlwaysCollapsed| exitUntilCollapsed (with minHeight and layout_height set)	Anywhere: appears collapsed (its min height) At Top: Full height	Anywhere: shrinks to collapsed only (its min height)	

(Bug?) scroll| enterAlways| enterAlwaysCollapsed| exitUntilCollapsed (with minHeight and layout_height set)	Anywhere: appears almost at full height. At Top: full height.	Anywhere: shrinks to collapsed only (its min height)	

Controls

Display control

[Make text in a control hyperlinkable]

To make text hyperlinkable set the android:autoLink property to "web".

```
<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/article"
        android:autoLink="web"
        android:layout_below="@id/article_subheading"
        android:lineSpacingExtra="@dimen/line_spacing"
        android:padding="@dimen/padding_regular"
        android:text="@string/article_text" />
```

(Google 2019, "Android Developers > Training > Android Developer Fundamentals > Codelabs for Android Developer Fundamentals", https://developer.android.com/courses/fundamentals-training/toc-v2), 4. Task 2: Add a ScrollView and an active web link, https://developer.android.com/codelabs/android-training-text-and-scrolling-views?index=..%2F..%2Fandroid-training#3

EditText

Add an Editable text field with <EditText>.

```
<EditText
    android:id="@+id/editText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Handling keyboard input, Specifying the input method type, https://developer.android.com/training/keyboard-input/style

Optionally use android:inputType, which allows bitwise combinations, to control keyboard layout and behaviour of the editable text field:

- For keyboard layout: text; textEmailAddress; textUri; number; phone.
- For behaviours of the editable text field include such values as: textCapSentences; textCapWords; textAutoCorrect; textPassword; textMultiline.

```
<EditText
    android:id="@+id/editText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textEmailAddress|textMultiLine|textCapWords" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Handling keyboard input, Specifying the input method type, https://developer.android.com/training/keyboard-input/style

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), https://developer.android.com/reference/android/widget/TextView.html#attr_android:inputType

After a user inputs text into an Editable text field they specify they have finished by pressing an "action" button on the keyboard. By default this action button is "actionDone" if there is no subsequent android:focusable fields; and "actioinNext" if there is such a field. However, you can override the kind of action button by setting the android:imeOptions attribute. For example, with values like "actionSend", "actionSearch", or "actionNone". android:imeOptions can be set in addition to any android:input type setting.

```
<EditText
    android:id="@+id/editText02"
    android:layout_width="match_parent"
    android:layout_height="wrap_content
    android:inputType="textEmailAddress
    android:imeOptions="actionSend"/>
```

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), https://developer.android.com/reference/android/widget/TextView.html#attr_android:imeOptions

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Handling keyboard input, Specifying the input method type, https://developer.android.com/training/keyboard-input/style#Action

android:imeOptions can also be used to set keyboard behaviours other than the kind of action button to display. android:imeOptions allows bitwise combinations.

```
<EditText
   android:id="@+id/editText02"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:inputType="textEmailAddress"
   android:imeOptions="actionSend|flagForceAscii" />
```

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), https://developer.android.com/reference/android/widget/TextView.html#attr_android:imeOptions

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Handling keyboard input, Specifying the input method type, https://developer.android.com/training/keyboard-input/style#Action

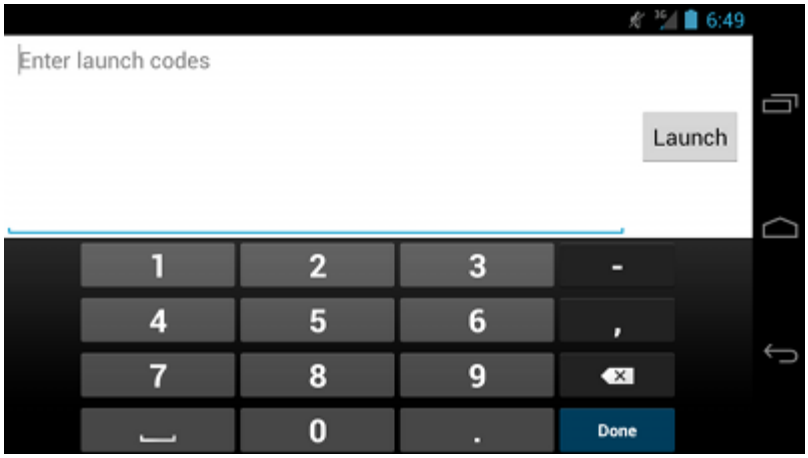
You can respond to the Action button specified by android:imeOptions by implementing a TextView.OnEditorActionListener Interface, following the general procedure specified in Event Listener Best Practice. Therefore we have something like ...

```
// Java. Inner fragment class of an activity.
// Exemplifying listening to OnClick as well as OnEditorAction events.
public static class PlaceholderFragment extends Fragment implements View
        .OnClickListener, TextView.OnEditorActionListener {

    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_user_interface, container,
                                         false);
```

```
        ....

        EditText editText02 = (EditText) rootView.findViewById(R.id.editText02);
        editText02.setOnEditorActionListener(this);

        return rootView;
    }
...

    @Override
    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
        boolean handled = false;
        switch (actionId) {
            case EditorInfo.IME_ACTION_SEND:
                Toast.makeText(getActivity(), "Sent: " + v.getText() ,
                               Toast.LENGTH_SHORT).show();
                handled = true;
            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }
        return handled;
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Handling keyboard input, Specifying the input method type, https://developer.android.com/training/keyboard-input/style#Action

Fullscreen "exract mode" is triggered when the keyboard is too big to share space with the underlying app. In this mode the keyboard and Editable text field fill to evenly share the screen; and the action button gets a label. This mode is generally triggered when you orient a handset in landscape.



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Handling keyboard input, Specifying the input method type, https://developer.android.com/training/keyboard-input/style#Action

You can specify the text of the Action button label when in fullscreen extract mode with android:imeActionLabel (although it might make using the OnEditorActionListener a bit more difficult).

```
<EditText
   android:id="@+id/editText02"
   android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:inputType="textEmailAddress"
    android:imeOptions="actionSend"
    android:imeActionLabel="Fonzi"/>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Handling keyboard input, Specifying the input method type, https://developer.android.com/training/keyboard-input/style#Action

ImageView

Bug. If ImageView does not display in lower API versions, try resizing it to get fixed dimensions rather than using wrap_content.

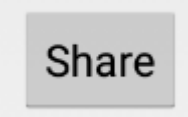
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\VectorDrawableDemoCompatActivity.java

Input controls

Buttons

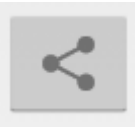
Buttons can be coded to show text, an icon, or both as follows:

Text.



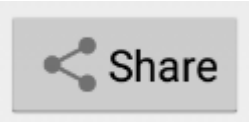

```
<Button
    android:id="@+id/button_share_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Share" />
```

Icon



```
<ImageButton
    android:id="@+id/button_share_icon"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_action_share" />
```

Text and Icon. The following technique could also be used to show only an icon by omitting the android:text property.




```
<Button
    android:id="@+id/button_share_icon_and_text"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawableLeft="@drawable/ic_action_share"
    android:text="Share" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Buttons, https://developer.android.com/guide/topics/ui/controls/button.html

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\fragment_user_interface.xml

For listening to a button's onclick event see Listening to Events.

One styling option particular to buttons is a borderless button.

```
<Button
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage"
    style="?android:attr/borderlessButtonStyle" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Buttons, https://developer.android.com/guide/topics/ui/controls/button.html

CompoundButtons

Compound Buttons are those that have a boolean state. These include: Checkboxes, ToggleButtons, Switches (from Android 4.0 API 14), and RadioButtons.

Checkboxes

Checkboxes allow users to set one or more boolean options from a set. So they work independently of each other.

```
<CheckBox
    android:id="@+id/checkBox01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="CheckBox 01" />
```

(Google API Guides, 2013. Develop > API Guides) http://developer.android.com/guide/topics/ui/controls/checkbox.html

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Checkboxes, https://developer.android.com/guide/topics/ui/controls/checkbox

You can listen to events by implementing a CheckBox.OnCheckedChangeListener Interface (or ComboundButton. OnCheckedChangeListener), following the general procedure specified in Event Listener Best Practice. Therefore we have something like ...

```
// Java. Inner fragment class of an activity.
public static class PlaceholderFragment extends Fragment implements View
        .OnClickListener, TextView.OnEditorActionListener,
        CheckBox.OnCheckedChangeListener {

    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
```

```
        View rootView = inflater.inflate(R.layout.fragment_user_interface, container,
                                         false);

   ....

        // Checkboxes
        CheckBox checkBox01 = (CheckBox) rootView.findViewById(R.id.checkBox01);
        checkBox01.setOnCheckedChangeListener(this);
        CheckBox checkBox02 = (CheckBox) rootView.findViewById(R.id.checkBox02);
        checkBox02.setOnCheckedChangeListener(this);

        return rootView;
    }
...

    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        String message = "";

        switch (buttonView.getId()){
            case R.id.checkBox01:
                if (isChecked) {
                    message = "Checkbox01 checked";
                } else {
                    message = "Checkbox01 unchecked";
                }
                break;
            case R.id.checkBox02:
                if (isChecked) {
                    message = "Checkbox02 checked";
                } else {
                    message = "Checkbox02 unchecked";
                }
                break;
            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }

        Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
    }
}
```

In the documentation it is suggested you use an onCheckClicked event (hooked up from the XML layout). This might have been deprecated??? In any case implementing onCheckedChanged seem to be the only possible way to do it in an XML independent manner (in keeping with best practice).

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\InputControlsActivity.java

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Checkboxes, Responding to Click Events,
https://developer.android.com/guide/topics/ui/controls/checkbox#HandlingEvents

Toggle Buttons

Toggle buttons allows the user to change a setting between two states, on and off by default.

```
<ToggleButton
    android:id="@+id/toggleButton01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOff="Lethargic"
    android:textOn="Energised" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Toggle Buttons, https://developer.android.com/guide/topics/ui/controls/togglebutton

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\fragment_user_interface.xml

You can listen to events by implementing a CombuondButton.OnCheckedChangeListener Interface, following the general procedure specified in Event Listener Best Practice. You therefore share the same event function with CheckBoxes and ToggleButtons. Note RadioButtons use a separate event function, leveraging the RadioGroup.OnCheckedChangeListener interface.

```
// Java. Inner fragment class of an activity.
public static class PlaceholderFragment extends Fragment implements View
        .OnClickListener, TextView.OnEditorActionListener,
        CompoundButton.OnCheckedChangeListener, RadioGroup.OnCheckedChangeListener {

    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_user_interface, container,
                                         false);

...

        // Checkboxes
        CheckBox checkBox01 = (CheckBox) rootView.findViewById(R.id.checkBox01);
        checkBox01.setOnCheckedChangeListener(this);
        CheckBox checkBox02 = (CheckBox) rootView.findViewById(R.id.checkBox02);
        checkBox02.setOnCheckedChangeListener(this);

        // RadioGroup
        RadioGroup radioGroup = (RadioGroup) rootView.findViewById(R.id.radioGroup01);
        radioGroup.setOnCheckedChangeListener(this);


        // ToggleButton
        ToggleButton toggleButton = (ToggleButton) rootView.findViewById(R.id

.toggleButton01);
        toggleButton.setOnCheckedChangeListener(this);
        return rootView;
    }

    // For Checkboxes and ToggleButtons
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        String message = "";

        switch (buttonView.getId()) {
            case R.id.checkBox01:
                if (isChecked) {
                    message = "Checkbox01 checked";
                } else {
                    message = "Checkbox01 unchecked";
                }
                break;
            case R.id.checkBox02:
                if (isChecked) {
                    message = "Checkbox02 checked";
                } else {
                    message = "Checkbox02 unchecked";
                }
                break;
            case R.id.toggleButton01:
                if (isChecked) {
                    message = "ToggleButton01 On (Energised)";
                } else {
                    message = "ToggleButton01 Off (Lethargic)";
                }
```

```
                break;
            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }

        Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        String message = "";
        switch (checkedId) {
            case R.id.radioButton01:
                message = "RadioButton 01";
                break;
            case R.id.radioButton02:
                message = "RadioButton 02";
                break;
            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }
        Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\InputControlsActivity.java

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Toggle Buttons, Responding to Button Presses, https://developer.android.com/guide/topics/ui/controls/togglebutton#ClickListener

Switches

Switches are essentially modern ToggleButtons and behave in the same manner. Available from Android 4.0 API 14.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Toggle Buttons, Responding to Button Presses, https://developer.android.com/guide/topics/ui/controls/togglebutton#ClickListener

Radio Buttons (and Radio Group)

Radio buttons allow the user to chose one, and only one, option from a set. To setup this mutually exclusive function you must place RadioButtons inside a RadioGroup. A RadioGroup is a subclass of LinearLayout and has a vertical orientation by default.

```
<RadioGroup
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <RadioButton
        android:id="@+id/radioButton01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Radio Button 01" />
    <RadioButton
        android:id="@+id/radioButton02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:text="Radio Button 02" />
</RadioGroup>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Radio Buttons, https://developer.android.com/guide/topics/ui/controls/radiobutton

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\fragment_user_interface.xml

You can listen to events by implementing a RadioGroup.OnCheckedChangeListener Interface, following the general procedure specified in Event Listener Best Practice. Therefore we have something like ...

```
// Java. Inner fragment class of an activity.
public static class PlaceholderFragment extends Fragment implements
        CheckBox.OnCheckedChangeListener, RadioGroup.OnCheckedChangeListener {

    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_user_interface, container,
                                         false);
...

        // Checkboxes
        CheckBox checkBox01 = (CheckBox) rootView.findViewById(R.id.checkBox01);
        checkBox01.setOnCheckedChangeListener(this);
        CheckBox checkBox02 = (CheckBox) rootView.findViewById(R.id.checkBox02);
        checkBox02.setOnCheckedChangeListener(this);

        // RadioGroup
        RadioGroup radioGroup = (RadioGroup) rootView.findViewById(R.id.radioGroup01);
        radioGroup.setOnCheckedChangeListener(this);

        return rootView;
    }

   ...

    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        String message = "";

        switch (buttonView.getId()) {
            case R.id.checkBox01:
                if (isChecked) {
                    message = "Checkbox01 checked";
                } else {
                    message = "Checkbox01 unchecked";
                }
                break;
            case R.id.checkBox02:
                if (isChecked) {
                    message = "Checkbox02 checked";
                } else {
                    message = "Checkbox02 unchecked";
                }
                break;
            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }

        Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
    }
```

```
    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        String message = "";
        switch (checkedId) {
            case R.id.radioButton01:
                message = "RadioButton 01";
                break;
            case R.id.radioButton02:
                message = "RadioButton 02";
                break;
            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }
        Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
    }
}
```

In the documentation you it is suggested to use onRadioButtonClicked (at the RadioButton level) rather than onCheckedChanged (at the RadioGroup level). The former method might be deprecated???

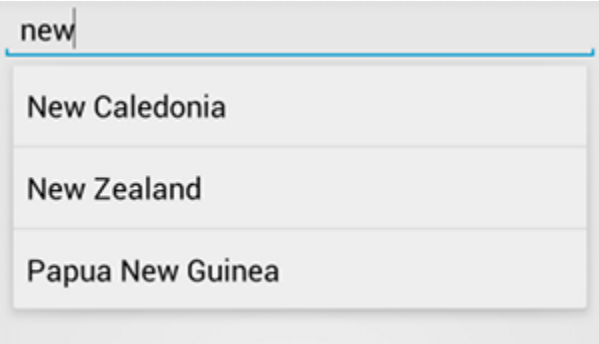
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\InputControlsActivity.java

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Radio Buttons, Responding to Click Events
https://developer.android.com/guide/topics/ui/controls/radiobutton#HandlingEvents

ComboBox-like inputs

Auto Complete Text View

AutoCompleteTextView provides suggestions to users as they type, like an auto-dropdown combo box, for very large sets of values. AutoCompleteTextView is a subclass of EditText.



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Handling keyboard input, Specifying the input method type, Provide auto-complete suggestions,
https://developer.android.com/training/keyboard-input/style#AutoComplete

Coding AutoCompleteTextView entails the following steps:

1. Specify an AutoCompleteTextView in your XML layout.

```
<AutoCompleteTextView
    android:id="@+id/autoCompleteTextView_country
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:hint="Country" />
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\fragment_user_interface.xml

2. Define a data source. In this example we'll use an XML resource file that defines a string array.

```
// \res\values\countries.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string-array name="countries_array">
    <item>Afghanistan</item>
    <item>Åland Islands</item>
    <item>Albania</item>
    <item>Algeria</item>
    <item>American Samoa</item>
    <item>Andorra</item>
    <item>Angola</item>
    <item>Anguilla</item>
...
    <item>Zambia</item>
    <item>Zimbabwe</item>
  </string-array>
</resources>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\values\countries.xml

3. In Java, in your Activity or Fragment, reference the AutoCompleteTextView and Datasource and use a suitable Array adapter to link the two. In this example we use an ArrayAdapter, but you could use an adapter suitable for a Database, for example.

```
// Java. Inner fragment class of an activity.
public static class PlaceholderFragment extends Fragment implements View
        .OnClickListener, TextView.OnEditorActionListener,
        CheckBox.OnCheckedChangeListener {

    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_user_interface, container,
                                         false);

...

        // AutoCompleteTextView Coding
        AutoCompleteTextView autoCompleteTextView = (AutoCompleteTextView) rootView
                .findViewById(
                R.id.autoCompleteTextView_country);

        String[] countries = getResources().getStringArray(R.array.countries_array);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(getActivity(),
                                                                android.R.layout

.simple_expandable_list_item_1,
                                                                countries);
        autoCompleteTextView.setAdapter(adapter);


        return rootView;
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Handling keyboard input, Specifying the input method type, Provide auto-complete suggestions, https://developer.android.com/training/keyboard-input/style#AutoComplete

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\InputControlsActivity.java

4. Limit an autocomplete entry to a single line with android:inputType="text".

```
<AutoCompleteTextView
    android:id="@+id/autoCompleteTextView_country"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Country"
    android:inputType="text"/>
```

(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/), http://stackoverflow.com/questions/12368558/android-how-to-make-an-autocompletetextview-singleline

Spinners

A spinner is essentially a combo-box, suitable for a small number of values in the set. Code a spinner as follows:

1. Specify the Spinner control in XML layout.

```
<Spinner
    android:id="@+id/spinner01"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Spinners, https://developer.android.com/guide/topics/ui/controls/spinner

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\fragment_user_interface.xml

2. Create a datasource. The data could come from an array or a database.

```
// \res\values\planets.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
        <item>Jupiter</item>
        <item>Saturn</item>
        <item>Uranus</item>
        <item>Neptune</item>
    </string-array>
</resources>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\values\planets.xml

3. Create an adapter from a datasource and specifying both the spinner default layout (when there is a single item selected) and the drop down layout. If the datasource is an array use an ArrayAdapter. If the datasource is a database use a CursorAdapter.

```
// Create Adapter and default spinner layout
```

```
ArrayAdapter<CharSequence> adapter1 = ArrayAdapter.createFromResource(
        getActivity(), R.array.planets, android.R.layout.simple_spinner_item);
// Specify layout when list of choices appears.
adapter1.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\InputControlsActivity.java

4. Hook up the spinner to the adapter.

```
// In a fragment we use rootView.
Spinner spinner = (Spinner) rootView.findViewById(R.id.spinner01);
spinner.setAdapter(adapter1);
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\InputControlsActivity.java

5. You can listen to events by implementing a AdapterView.OnItemSelectedListener Interface, following the general procedure specified in Event Listener Best Practice. Two methods will need to be implemented: onItemSelected and onNothingSelected.

```
// Java. Inner fragment class of an activity.
public static class PlaceholderFragment extends Fragment implements View
        .OnClickListener, TextView.OnEditorActionListener,
        CompoundButton.OnCheckedChangeListener, RadioGroup.OnCheckedChangeListener,
        AdapterView.OnItemSelectedListener {

    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_user_interface, container,
                                         false);

        ...

        // Spinner
        // Create Adapter and default spinner layout
        ArrayAdapter<CharSequence> adapter1 = ArrayAdapter.createFromResource(
                getActivity(), R.array.planets, android.R.layout.simple spinner item);
        // Specify layout when list of choices appears.
        adapter1.setDropDownViewResource(
                android.R.layout.simple_spinner_dropdown_item);
        // Hook the spinner up to the adapter.
        Spinner spinner = (Spinner) rootView.findViewById(R.id.spinner01);
        spinner.setAdapter(adapter1);
        // Hooke the spinner up to the event listening interface.
        spinner.setOnItemSelectedListener(this);

        return rootView;
    }



    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position,
                               long id) {
        String planet = (String) parent.getItemAtPosition(position);
        Toast.makeText(getActivity(), "Selected " + planet,
                       Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
        Toast.makeText(getActivity(), "Nothing selected", Toast.LENGTH_SHORT).show();
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Spinners, Responding to User Selections, https://developer.android.com/guide/topics/ui/controls/spinner#SelectListener

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\InputControlsActivity.java

Events

Event Types

Event types include:

Method	Interface
onClick()	View.OnClickListener
onLongClick()	View.OnLongClickListener
OnFocusChange()	View.OnFocusChangeListener
OnKey()	View.OnKeyListener
OnTouch()	View.OnTouchListener
OnCreateContextMenu()	View.OnCreateContextMenuListener

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Event listeners, https://developer.android.com/guide/topics/ui/ui-events#EventListeners

Some event methods return a boolean. True is used to indicate that the event has been "consumed" and should not be processed further.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Event listeners, https://developer.android.com/guide/topics/ui/ui-events#EventListeners

For validating user input do not rely on key events when the user presses return. Instead actions like IME_ACTION_DONE (??? fix : more detail).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Event listeners, https://developer.android.com/guide/topics/ui/ui-events#EventListeners

In general you do not rely on key events. In this respect Android is unlike a Windows application. Note that the click events occur for both touch and not touch events. E.g. onClick is called either when:

- The user touches the item; or
- Focuses upon the item with the navigation keys, or trackball; and presses the Enter key, or presses down on the trackball.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Event listeners, https://developer.android.com/guide/topics/ui/ui-events#EventListeners

Creating Events

When thinking about how all the code needs to deal with events we need to think about three kinds of code:

- In server code: provide a means to for the client code to define a response to the event.
- In client code: respond to the event.
- In server code: raise an event.

When writing custom components you'll want to handle 1 & 3 with some standard events. See...

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Event handlers, https://developer.android.com/guide/topics/ui/ui-events#EventHandlers

Listening to Events

Overview

There are several ways to listen to events (such as a click event from a button). There are three choices you need to make:

- How to "hook up" the event: from XML (to Java); or purely in Java.
- If hooking up the event purely in Java, how to pass a View.On...Listener() Interface to a control's set...Listener() method.
- Whether to have, for the controls in an activity or fragment: separate event handlers; or a central event handler.

Hook up events

Hook up the event from XML (to Java):

- In your layout xml specify the event name and the methodName that is to respond to the event.
- In Java write the procedure matching the methodName with: public scope; void return value; and a view parameter.

```
// my_layout.xml
<Button
    android:id="@+id/button_share_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Share"
    android:onClick="shareOnClick"/>

// Java
public void shareOnClick(View view) {
// Do something
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App basics, Build your first app, Startanotehr activity, Respond to the Send button, https://developer.android.com/training/basics/firstapp/starting-activity#RespondToButton

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Buttions, Responding to Click Events, https://developer.android.com/guide/topics/ui/controls/button#HandlingEvents

Hook up the event purely in Java when: you create the control entirely in code; or you are listening to the event in a fragment.

(Google API Guides, 2013. Develop > API Guides)
http://developer.android.com/guide/topics/ui/controls/button.html#ClickListener

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Buttions, Responding to Click Events, Using an OnClickListener, https://developer.android.com/guide/topics/ui/controls/button#ClickListener

Hook up the event purely in Java by passing an implementation of an anonymous View.On....Listener Interface to a control's set...Listener() method.

```
public static class PlaceholderFragment extends Fragment {

    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_user_interface, container,
                                         false);

        // The following is for use in a fragment. In an activity drop "rootView"
        Button button = (Button) rootView.findViewById(R.id.button_share_text);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Do Something
            }
        });

        return rootView;
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Buttions, Responding to Click Events, https://developer.android.com/guide/topics/ui/controls/button#HandlingEvents

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Event listeners, https://developer.android.com/guide/topics/ui/ui-events#EventListeners

There are three ways to pass a View.On...Listener() Interface to a control's set...Listener() method:

1. Inline, as above.
2. As a class level variable.

```
public static class PlaceholderFragment extends Fragment {

    private View.OnClickListener mOnClickListener = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Do Semething
        }
    };

    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
```

```
        View rootView = inflater.inflate(R.layout.fragment_user_interface, container,
                                         false);

        // The following is for use in a fragment. In an activity drop "rootView"
        Button button = (Button) rootView.findViewById(R.id.button_share_text);
        button.setOnClickListener(mOnClickListener);

        return rootView;
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Event listeners, https://developer.android.com/guide/topics/ui/ui-events#EventListeners

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\InputControlsActivity.java

By passing the containing the class (fragment or activity), where the containing class implements View.OnClickListener.

```
public static class PlaceholderFragment extends Fragment implements View
        .OnClickListener {

    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_user_interface, container,
                                         false);

        // The following is for use in a fragment. In an activity drop "rootView"
        Button button = (Button) rootView.findViewById(R.id.button_share_text);
        button.setOnClickListener(this);

        return rootView;
    }

    @Override
    public void onClick(View v) {
        // Do Something
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Event listeners, https://developer.android.com/guide/topics/ui/ui-events#EventListeners

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\InputControlsActivity.java

Separate Event Listeners V Central Event Listener

When you have two or more controls for which you want to listen to the same event then you can choose between having, for every control: a central event listener; or separate event listeners.

A separate event listeners example.

```
// res\layout\my_layout.xml
// Note no event listener is specified.
<Button
    android:id="@+id/button_share_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Share"
  />
```

```
<Button
    android:id="@+id/button_share_icon_and_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawableLeft="@drawable/ic_action_share"
    android:text="Share" />

// Java. Inner fragment class of an activity.
public static class PlaceholderFragment extends Fragment {

    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_user_interface, container,
                                         false);

        // The following is for use in a fragment. In an activity drop "rootView"
        Button button = (Button) rootView.findViewById(R.id.button_share_text);
        button.setOnClickListener(new View.OnClickListener() {
                                      @Override
                                      public void onClick(View v) {
                                          Toast.makeText(getActivity(),
                                                         "button_share_text",
                                                         Toast.LENGTH_SHORT).show();
                                      }
                                  });

        // The following is for use in a fragment. In an activity drop "rootView"
        button = (Button) rootView.findViewById(R.id.button_share_icon_and_text);
        button.setOnClickListener(new View.OnClickListener() {
                                      @Override
                                      public void onClick(View v) {
                                          Toast.makeText(getActivity(),
                                                         "button share icon and text",
                                                         Toast.LENGTH_SHORT).show();
                                      }
                                  });

        return rootView;
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Event listeners, https://developer.android.com/guide/topics/ui/ui-events#EventListeners

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\InputControlsActivity.javas

A central event listener example (this is also an example of best practice).

```
// res\layout\my_layout.xml
// Note no event listener is specified.
<Button
    android:id="@+id/button_share_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Share"
  />

<Button
    android:id="@+id/button_share_icon_and_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawableLeft="@drawable/ic_action_share"
    android:text="Share" />

// Java. Inner fragment class of an activity.
public static class PlaceholderFragment extends Fragment implements View
        .OnClickListener {
```

```
    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_user_interface, container,
                                         false);

        // The following is for use in a fragment. In an activity drop "rootView"
        Button button = (Button) rootView.findViewById(R.id.button_share_text);
        button.setOnClickListener(this);

        // The following is for use in a fragment. In an activity drop "rootView"
        button = (Button) rootView.findViewById(R.id.button_share_icon_and_text);
        button.setOnClickListener(this);

        return rootView;
    }

    /**
     * Central Event Listener
     */
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button_share_text:
                Toast.makeText(getActivity(), "button_share_text",
                               Toast.LENGTH_SHORT).show();
                break;
            case R.id.button_share_icon_and_text:
                Toast.makeText(getActivity(), "button_share_icon_and_text",
                               Toast.LENGTH_SHORT).show();
                break;
            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }

    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Event listeners, https://developer.android.com/guide/topics/ui/ui-events#EventListeners

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\InputControlsActivity.javas

Event Listener Best Practice

Unless there is an overriding reason to do so the best practice for creating an event listener is to:

Hook it up purely in Java:

- Pass a View.On...Listener() Interface to a control's set...Listener() method: By passing the containing the class (fragment or activity), where the containing class implements View.OnClickListener.
- Use a central event listener.

Because:

- Fragment event listeners can only be created purely in Java. Even if you are creating an event listener(s) for an Activity, you'll want to make your code portable such that it could be moved to a fragment.
- Having the containing class implement View.OnClickListener avoids extra class load and object allocation. It also reduced code complexity.
- Central event listeners create tidier code.

These are John Bentley assertions. But one of the reasons can be found at:
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Event listeners, https://developer.android.com/guide/topics/ui/ui-events#EventListeners

See also (Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/), How to handle button clicks using the xml onClick within Fragments, Adorjan Princz, 2013-01-28 http://stackoverflow.com/a/14571018/872154

Best practice example: see "A central event listener example" above.

For listening to multiple event types just implement more event interfaces.

```
public static class PlaceholderFragment extends Fragment implements View
        .OnClickListener, TextView.OnEditorActionListener {
....
```

Focus

Some UI components can receive focus. Whether they receive focus or not depends, in party, on whether the system is in "Touch mode" or not. Touch mode begins when a user touches a screen. Touch mode exits when a user hits a directional key or moves a trackball.

Determine whether the system is in touch mode with `isInTouchMode()`.

Determine whether a UI component can receive focuse with `isFocusable()` and `isFocusableInTouchMode()`.

In touch mode (when a user is navigating with a finger) it doesn't make sense for some UI components to receive focus. For example, when the button is touched it will just fire their onClick listener, it won't receive focus.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Touch Mode, https://developer.android.com/guide/topics/ui/ui-events#TouchMode

In general you don't need to do anything to properly handle the focus of UI components. The framework sets sensible defaults.

However, you can change focus behaviour of a UI component (a view) in these ways:

In java use `setFocusable()` or `setFocusableInTouchMode()` for a view. In XML use `android:focusable` and `android:focusableInTouchMode` attributes.

```
<CheckBox
    android:id="@+id/checkBox02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="CheckBox 02"
    android:focusableInTouchMode="true" />
```

Initial testing shows that in XML at least using `android:focusable` and `android:focusableInTouchMode` is tempremental. Even when only using the keyboard direction keys

`android:focusableInTouchMode` should be used in order to give a UI component the focus.

Therefore, if you want a UI component to get focus when it normally wouldn't: use (at least) android:focusableInTouchMode.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Handling focus, https://developer.android.com/guide/topics/ui/ui-events#HandlingFocus

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\fragment_input_controls.xml

Change focus order in XML with the attributes: nextFocusDown, nextFocusLeft, nextFocusRight, nextFocusUP.

```
LinearLayout
    android:orientation="vertical"
    ... >
  <Button android:id="@+id/top"
          android:nextFocusUp="@+id/bottom"
          ... />
  <Button android:id="@+id/bottom"
          android:nextFocusDown="@+id/top"
          ... />
</LinearLayout>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Handling focus, https://developer.android.com/guide/topics/ui/ui-events#HandlingFocus

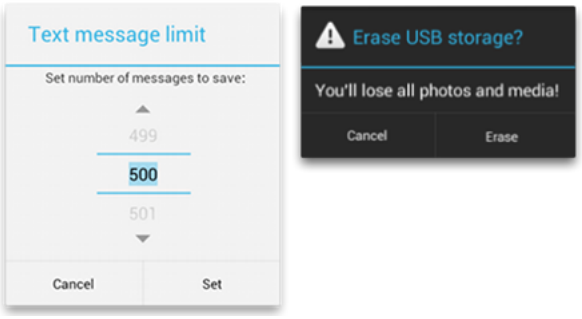
In java give a few focus with `requestFocus()` on a view.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Touch & Input, Input events, Handling focus, https://developer.android.com/guide/topics/ui/ui-events#HandlingFocus

Dialogs

Overview

A dialog (box) is a small window that prompts the user for a response. It is usually modal.



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, https://developer.android.com/guide/topics/ui/dialogs.html

Dialogs inherit from the base class "Dialog". You don't use the base class directly. There are three Dialog subclasses:

- AlertDialog. Shows any of: title, a message, a list of selectable items, or up to three buttons; or a custom layout.
- TimePickerDialog.
- DatePickerDialog.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, https://developer.android.com/guide/topics/ui/dialogs.html

Basic Coding Procedure

To code a Dialog follow this general procedure:

Create a DialogFragment, within which you will create your Dialog. The DialogFragment can be created:

- Within a PlaceholderFragment (which is in turn inside an Activity).
- Outside the PlaceholderFragment, within the enclosing Activity.
- Independently, in it's own Java Class file.

```
// In this example we choose 1.c., Independently, in it's own Java Class file.
public class FireMissilesDialogFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        return super.onCreateDialog(savedInstanceState);
    }
}
```

The different optons, a, b, or c, will change whether you have to declare the class statically or not.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, https://developer.android.com/guide/topics/ui/dialogs.html

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\FireMissilesDialogFragment.java

If you are supporting Apps:

- Lower than Android 3.0 (API 11) then use: import android.support.v4.app.DialogFragment;
- Only equal to and above Android 3.0 (API 11) then use: import android.app.DialogFragment;

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, https://developer.android.com/guide/topics/ui/dialogs.html

Within the DialogFragment.onCreateDialog(), create your dialog.

```
public class FireMissilesDialogFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        builder.setTitle("Fire Missiles Dialog");
        builder.setMessage("Fire missiles?");
        builder.setPositiveButton("Fire", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getActivity(), "Fire!", Toast.LENGTH_SHORT).show();
            }
        });
        builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
```

```
            public void onClick(DialogInterface dialog,
                                int which) {
                FireMissilesDialogFragment.this.getDialog().cancel();
                Toast.makeText(getActivity(),
                               "Missiles Cancelled!",
                               Toast.LENGTH_SHORT).show();
            }
        });
        return builder.create();
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Creating a Dialog Fragment, https://developer.android.com/guide/topics/ui/dialogs.html#DialogFragment

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\FireMissilesDialogFragment.java

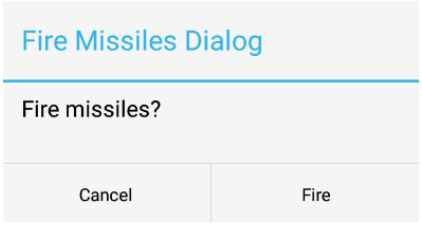
Create an activity level variable for the target DialogFragment (this helps with preserving state and handling runtime configuration changes, like rotating the screen). Show the dialog fragment (which contains the dialog) from your interface. Use getFragmentManager() >= Api 11 and getSupportFragmentManager() < Api 11.

```
// \res\layout\fragment_dialog_demo.xml
<Button
    android:id="@+id/button_fire_missiles"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Fire Missiles" />

// \java\au\com\softmake\mysecondapp\DialogDemoActivity.java

    // An activity level variable in order to keep the AlertDialogList alive between
    // runtime
    // configuration changes (like rotating the screen), and also save state.
    static private FireMissilesDialogFragment mFireMissileDialogFragment;

// Fragment is inside a the class DialogDemoActivity
public static class PlaceholderFragment extends Fragment implements View
        .OnClickListener {
....

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button_fire_missiles:
                // Only create if it doesn't already exist.
                if (mFireMissileDialogFragment == null) {
                    mFireMissileDialogFragment = new FireMissilesDialogFragment();
                }

                // Android 3.0 (API 11) >=  android.app.Fragment.getFragmentManager();
                // Android 3.0 (API 11) < import android.support.v4.app.FragmentActivity
                // .getSupportFragmentManager()
                mFireMissileDialogFragment.show(getFragmentManager(),
                                                "FireMissileDialogFragmentTag");
                break;
            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Showing a Dialog, https://developer.android.com/guide/topics/ui/dialogs.html#ShowingADialog

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\DialogDemoActivity.java

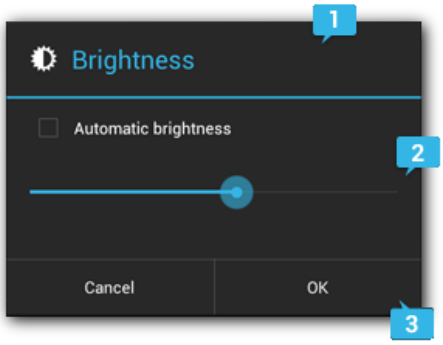
The result ...



Alert Dialog Anatomy and Design

An AlertDialog has three regions:

- Title. Optional. Should only be used if the content area has a detailed message, a list, or a custom layout.
- Content area. For a message, list, or custom layout.
- Action buttons. Maximum of three. Postive (E.g. OK, Yes, or Accept); Negative (Cancel); Neutral, when you don't want to proceed but you don't want to cancel (e.g "Remind me later").



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Building an Alert Dialog, https://developer.android.com/guide/topics/ui/dialogs.html#AlertDialog

Alert List Dialog

Coding Options

You can set the AlertDialog content area as a list. The list can be comprised of three types:

- Single choice items. Plain. Not persistent.
- Single choice items. Radio buttons. Persistent.
- Multiple choice items. Checkboxes. Persistent.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Building an Alert Dialog, Adding a list, https://developer.android.com/guide/topics/ui/dialogs.html#AddingAList

Coding Procedure

Basics

Procedure for an Alert List Dialog. Starting with: Single choice items; Plain; Not persistent.

When building your Alert List Dialog: don't set a message, you can't share the content area between a list and a message; and always set a title.

```
/**
 * An Alert List Dialog
 */
public class FireMissilesAlertListDialogFragment extends DialogFragment {
...

    // "Every fragment must have an empty constructor, so it can be instantiated
    // when restoring its activity's state." http://developer.android
    // .com/reference/android/app/Fragment.html
    public FireMissilesAlertListDialogFragment() {
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        // You must set the title.
        builder.setTitle("Fire Missiles Dialog");
        // Don't set a message.
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Building an Alert Dialog, Adding a list, https://developer.android.com/guide/topics/ui/dialogs.html#AddingAList

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\FireMissilesAlertListDialogFragment.java

Create a datasource for the list items. This can be an array or a database. In our example we just have a simple array.

```
public class FireMissilesAlertListDialogFragment extends DialogFragment {
    public String[] mItems = {"ICBM", "Hellfire", "Propaganda"};
```

Pass the datasource to the AlertDialog with setItems; and handle click events on each item. If you use a database then you'll want to: specify the list with setAdpater(); back the list with a ListAdapter; and use a Loader to load the list Asyncrhonously. In the following example, however, we use an array. We don't set any action buttons as we listen to events on list item press.

```
public class FireMissilesAlertListDialogFragment extends DialogFragment {
    public String[] mItems = {"ICBM", "Hellfire", "Propaganda"};

    // "Every fragment must have an empty constructor, so it can be instantiated
```

```
    // when restoring its activity's state." http://developer.android
    // .com/reference/android/app/Fragment.html
    public FireMissilesAlertListDialogFragment() {
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        // You must set the title.
        builder.setTitle("Fire Missiles Dialog");
        // Don't set a message.

        builder.setItems(mItems, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getActivity(), "Clicked: " + mItems[which],
                               Toast.LENGTH_SHORT).show();
            }
        });


        return builder.create();
    }
}
```

In your calling activity set an activity level variable for the DialogFragment, in order to handle runtime configuration changes.

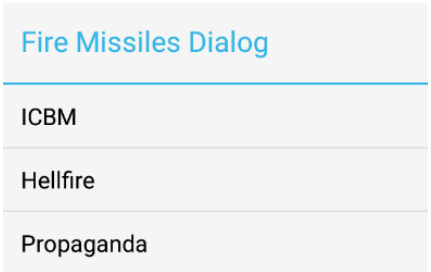
```
public class DialogDemoActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dialog_demo);
        if (savedInstanceState == null) {
            getFragmentManager().beginTransaction().add(R.id.container,
                                                        new PlaceholderFragment())
                    .commit();
        }
    }


    // An activity level variable in order to keep the AlertDialogList alive between
    // runtime
    // configuration changes (like rotating the screen), and also save state.
    static private FireMissilesAlertListDialogFragment mFireMissileDialogFragment;
```

Call the DialogFragment from a PlaceholderFragment (or Activity) as follows.

```
public class DialogDemoActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dialog_demo);
        if (savedInstanceState == null) {
            getFragmentManager().beginTransaction().add(R.id.container,
                                                        new PlaceholderFragment())
                    .commit();
        }
    }


    // An activity level variable in order to keep the AlertDialogList alive between
    // runtime
    // configuration changes (like rotating the screen), and also save state.
    static private FireMissilesAlertListDialogFragment mFireMissileDialogFragment;

    /**
     * A placeholder fragment containing a simple view.
     */
```

```
    public static class PlaceholderFragment extends Fragment implements View
            .OnClickListener {

        // "Every fragment must have an empty constructor, so it can be instantiated
        // when restoring its activity's state." http://developer.android
        // .com/reference/android/app/Fragment.html
        public PlaceholderFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            View rootView = inflater.inflate(R.layout.fragment_dialog_demo, container,
                                             false);
            Button button = (Button) rootView.findViewById(R.id.button_fire_missiles);
            button.setOnClickListener(this);
            return rootView;
        }

        public void createAndShowFireMissileDialogFragment() {

                // Only create if it doesn't already exist.
                if (mFireMissileDialogFragment == null) {
                    mFireMissileDialogFragment = new FireMissilesAlertListDialogFragment();
                }

                // Android 3.0 (API 11) >=  android.app.Fragment.getFragmentManager();
                // Android 3.0 (API 11) < import android.support.v4.app.FragmentActivity
                // .getSupportFragmentManager()
                mFireMissileDialogFragment.show(getFragmentManager(),
                                                "FireMissileDialogFragmentTag");
        }

        @Override
        public void onClick(View v) {
            switch (v.getId()) {
                case R.id.button_fire_missiles:
                    createAndShowFireMissileDialogFragment();
                    break;
                default:
                    try {
                        throw new Exception("Unhandled case in switch");
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
            }
        }
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\DialogDemoActivity.java



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Building an Alert Dialog, Adding a list, https://developer.android.com/guide/topics/ui/dialogs.html#AddingAList

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\FireMissilesDialogFragment.java

Multihcoice, Save State

To create a persistent AlertDialog box, as either a Single choice list (radio buttons) or a Multi choice list (checkboxes), refactor the above as follows (we'll use a multi choice list as our example):

In your DialogFragment create a fragment level boolean array mItemsCheckedStatus and initialize all values to false.

```
public class FireMissilesAlertListDialogFragment extends DialogFragment {
    public String[] mItems = {"ICBM", "Hellfire", "Propaganda"};
    private boolean[] mItemsCheckedStatus;

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        // You must set the title.
        builder.setTitle("Fire Missiles Dialog");
        // Don't set a message.

        // Initialize
        if (mItemsCheckedStatus == null) {
            mItemsCheckedStatus = new boolean[mItems.length];
            for (int i = 0; i < mItems.length; i++) {
                mItemsCheckedStatus[0] = false;
            }
        }
```

To save state of class level variagles in your AlertList DialogFragment use Bundle objects with onSaveStabeInstanceState() and initialize your class level variables with the the bundle. This is uncessary if your AlertDialog is a custom dialog and your layout components have ids, since the state of those components is preserved, for free, by the system.

```
/**
 * An Alert List Dialog
 */
public class FireMissilesAlertListDialogFragment extends DialogFragment {
    public String[] mItems = {"ICBM", "Hellfire", "Propaganda"};
    private boolean[] mItemsCheckedStatus;

    // "Every fragment must have an empty constructor, so it can be instantiated
    // when restoring its activity's state." http://developer.android
    // .com/reference/android/app/Fragment.html
    public FireMissilesAlertListDialogFragment() {
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        // You must set the title.
        builder.setTitle("Fire Missiles Dialog");
        // Don't set a message.

        // Initialize
        if (mItemsCheckedStatus == null) {
            mItemsCheckedStatus = new boolean[mItems.length];
            for (int i = 0; i < mItems.length; i++) {
                mItemsCheckedStatus[0] = false;
            }
        }

        if (savedInstanceState != null) {
            mItemsCheckedStatus = savedInstanceState.getBooleanArray
                    ("mItemsCheckedStatus");
        }
```

```
...
    }

    /**
     * We need to save our class level variables to the outState bundle
     * so that they may be retrieved when the user:
     * 1. Selects some new values.
     * 2. Rotates the screen.
     * 3. And clicks the positive button without otherwise touching the dialog.
     */
    @Override
    public void onSaveInstanceState(Bundle outState) {
        outState.putBooleanArray("mItemsCheckedStatus", mItemsCheckedStatus);
        super.onSaveInstanceState(outState);
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\FireMissilesAlertListDialogFragment.java

Set the list items with builder.setMultiChoiceItems (or setSingleChoiceItems): passing the array datasource, defining a listener, and storing the selected values (in a module level variable).

```
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        ...

        builder.setMultiChoiceItems(mItems, mItemsCheckedStatus,
                                    new DialogInterface.OnMultiChoiceClickListener() {
                                        @Override
                                        public void onClick(DialogInterface dialog,
                                                            int which,
                                                            boolean isChecked) {
                                            mItemsCheckedStatus[which] = isChecked;
                                        }
                                    });
```

Set click listeners for the positive and negative (and possibly neutral) buttons.

```
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        ...
        // Setup Action buttons and respond to click.
        builder.setPositiveButton("Fire", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                ArrayList<String> checkedItems = new ArrayList<String>();
                for (int i = 0; i < mItems.length; i++) {
                    if (mItemsCheckedStatus[i]) {
                        checkedItems.add(mItems[i]);
                    }
                }
                String message = "Fire: " + checkedItems;
                Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();

            }
        });
        builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getActivity(), "Missiles Cancelled!",
                               Toast.LENGTH_SHORT).show();
            }
        });
```

Return the builder from onCreateDialog.

```
    @Override
```

```
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        return builder.create();
    }
```

Full Example

```
// DialogDemoActivity.java
public class DialogDemoActivity extends Activity implements
        SignInAlertCustomDialogFragment.SignInAlertCustomDialogFragmentListener {

    @Override
    public void onDialogPositiveClick(DialogFragment dialogFragment, String someData) {
        Toast.makeText(getApplicationContext(), "Activity Event CallBack: " + someData,
                       Toast.LENGTH_SHORT).show();
    }

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

        setContentView(R.layout.activity_dialog_demo);
        if (savedInstanceState == null) {
            getFragmentManager().beginTransaction().add(R.id.container,
                                                        new PlaceholderFragment())
                    .commit();
        }
    }

    @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_dialog_demo, 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);
    }


    // An activity level variable in order to keep the AlertDialogList alive between
    // runtime
    // configuration changes (like rotating the screen), and also save state.
    static private FireMissilesAlertListDialogFragment mFireMissileDialogFragment;
    static private SignInAlertCustomDialogFragment mSignInAlertCustomDialogFragment;

    /**
     * A placeholder fragment containing a simple view.
     */
    public static class PlaceholderFragment extends Fragment implements View
            .OnClickListener {

        // "Every fragment must have an empty constructor, so it can be instantiated
        // when restoring its activity's state." http://developer.android
        // .com/reference/android/app/Fragment.html
        public PlaceholderFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            View rootView = inflater.inflate(R.layout.fragment_dialog_demo, container,
                                             false);
            Button button = (Button) rootView.findViewById(R.id.button_fire_missiles);
```

```
            button.setOnClickListener(this);

            button = (Button) rootView.findViewById(R.id.button_sign_in);
            button.setOnClickListener(this);

            return rootView;
        }

        private void createAndShowFireMissileDialogFragment() {

            // Only create if it doesn't already exist.
            if (mFireMissileDialogFragment == null) {
                mFireMissileDialogFragment = new FireMissilesAlertListDialogFragment();
            }

            // Android 3.0 (API 11) >=  android.app.Fragment.getFragmentManager();
            // Android 3.0 (API 11) < import android.support.v4.app.FragmentActivity
            // .getSupportFragmentManager()
            mFireMissileDialogFragment.show(getFragmentManager(),
                                            "FireMissileDialogFragmentTag");
        }

        private void createAndShowSignInAlertCustomDialogFragment() {
            Bundle bundle = new Bundle();

            // Only create if it doesn't already exist.
            if (mSignInAlertCustomDialogFragment == null) {
                mSignInAlertCustomDialogFragment = new SignInAlertCustomDialogFragment();
                bundle.putString("TITLE", "Beautiful Sign In");
                mSignInAlertCustomDialogFragment.setArguments(bundle);
            }

            // Android 3.0 (API 11) >=  android.app.Fragment.getFragmentManager();
            // Android 3.0 (API 11) < import android.support.v4.app.FragmentActivity
            // .getSupportFragmentManager()
            mSignInAlertCustomDialogFragment.show(getFragmentManager(),
                                                  "SignInAlertCustomDialogFragmentTag");
        }


        @Override
        public void onClick(View v) {
            switch (v.getId()) {
                case R.id.button_fire_missiles:
                    createAndShowFireMissileDialogFragment();
                    break;
                case R.id.button_sign_in:
                    createAndShowSignInAlertCustomDialogFragment();
                    break;
                default:
                    try {
                        throw new Exception("Unhandled case in switch");
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
            }
        }
    }
}

// FireMissilesAlertListDialogFragment
/**
 * An Alert List Dialog
 */
public class FireMissilesAlertListDialogFragment extends DialogFragment {
    public String[] mItems = {"ICBM", "Hellfire", "Propaganda"};
    private boolean[] mItemsCheckedStatus;

    // "Every fragment must have an empty constructor, so it can be instantiated
    // when restoring its activity's state." http://developer.android
    // .com/reference/android/app/Fragment.html
    public FireMissilesAlertListDialogFragment() {
    }



    @Override
```

```
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        // You must set the title.
        builder.setTitle("Fire Missiles Dialog");
        // Don't set a message.

        // Initialize
        if (mItemsCheckedStatus == null) {
            mItemsCheckedStatus = new boolean[mItems.length];
            for (int i = 0; i < mItems.length; i++) {
                mItemsCheckedStatus[0] = false;
            }
        }

        if (savedInstanceState != null) {
            mItemsCheckedStatus = savedInstanceState.getBooleanArray
                    ("mItemsCheckedStatus");
        }


//        builder.setItems(mItems, new DialogInterface.OnClickListener() {
//            @Override
//            public void onClick(DialogInterface dialog, int which) {
//                Toast.makeText(getActivity(), "Clicked: " + mItems[which],
//                               Toast.LENGTH_SHORT).show();
//            }
//        });


        builder.setMultiChoiceItems(mItems, mItemsCheckedStatus,
                                    new DialogInterface.OnMultiChoiceClickListener() {
                                        @Override
                                        public void onClick(DialogInterface dialog,
                                                            int which,
                                                            boolean isChecked) {
                                            mItemsCheckedStatus[which] = isChecked;
                                            Toast.makeText(getActivity(),
                                                           "Clicked: " + mItems[which],
                                                           Toast.LENGTH_SHORT).show();
                                        }
                                    });

        // Setup Action buttons and respond to click.
        builder.setPositiveButton("Fire", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                ArrayList<String> checkedItems = new ArrayList<String>();
                for (int i = 0; i < mItems.length; i++) {
                    if (mItemsCheckedStatus[i]) {
                        checkedItems.add(mItems[i]);
                    }
                }
                String message = "Fire: " + checkedItems;
                Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();

            }
        });
        builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getActivity(), "Missiles Cancelled!",
                               Toast.LENGTH_SHORT).show();
            }
        });

        return builder.create();
    }

    /**
     * We need to save our class level variables to the outState bundle
     * so that they may be retrieved when the user:
     * 1. Selects some new values.
     * 2. Rotates the screen.
     * 3. And clicks the positive button without otherwise touching the dialog.
     */
```

```
    @Override
    public void onSaveInstanceState(Bundle outState) {
        outState.putBooleanArray("mItemsCheckedStatus", mItemsCheckedStatus);
        super.onSaveInstanceState(outState);
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\DialogDemoActivity.java
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\FireMissilesAlertListDialogFragment.java

Custom Alert Dialog

There are two ways to creat a custom alert dialog:

1. Create a custom Activity and in AndroidManifest.xml set its theme to Theme.Holo.Dialog.

   ```
   <activity android:theme="@android:style/Theme.Holo.Dialog" >
   ```

   (Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Building an Alert Dialog, Creating a Custom Layout, https://developer.android.com/guide/topics/ui/dialogs.html#CustomLayout

2. Use the Dialog API

   (Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Building an Alert Dialog, Creating a Custom Layout, https://developer.android.com/guide/topics/ui/dialogs.html#CustomLayout

To use the Dialog API to create a custom alert dialog you ...

Create: a custom UI in xml.

```
// layout/fragment_dialog_demo.xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity horizontal margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="au.com.softmake.mysecondapp.DialogDemoActivity$PlaceholderFragment">

    <TextView
        android:id="@+id/textView heading alertDialog"
        style="@style/TextHeadingStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Alert Dialog" />

    <Button
        android:id="@+id/button_fire_missiles"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Fire Missiles - Alert List Dialog" />

    <Button
        android:id="@+id/button_sign_in"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sign In - Custom Dialog " />
```

```
</LinearLayout>
```

Inflate the UI in your AlertCustomCustomDialogFragment and then set that inflation using the AlertDialog.Builder.

```
public class SignInAlertCustomDialogFragment extends DialogFragment {

    // "Every fragment must have an empty constructor, so it can be instantiated
    // when restoring its activity's state." http://developer.android
    // .com/reference/android/app/Fragment.html
    public SignInAlertCustomDialogFragment() {
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        builder.setTitle("Sign In Dialog");
        // Don't set a message.

        LayoutInflater inflater = getActivity().getLayoutInflater();
        builder.setView(inflater.inflate(R.layout.dialog_sign_in, null));
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\zaradata\SignInAlertCustomDialogFragment.java

Set the positive action button, casting DialogInterface to Dialog to access UI widgets (and their values).

```
        // Setup Action buttons and respond to click.
        builder.setPositiveButton("Sign In", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

                // Cast DialogInterface to Dialog class to access UI widgets
                Dialog dialogObject = (Dialog) dialog;
                TextView userNameTextView = (TextView) dialogObject.findViewById(
                        R.id.editText_userName);
                TextView passwordTextView = (TextView) dialogObject.findViewById(
                        R.id.editText_password);
                String message = String.format("Signed in with %s, %s",
                                               userNameTextView.getText(),
                                               passwordTextView.getText());
                Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();

            }
        });
```

Set the negative action button, call dialog.cancel() if you are cancelling.

```
        builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
                Toast.makeText(getActivity(), "Cancelled!", Toast.LENGTH_SHORT).show();
            }
        });

        return builder.create();
    }
}
```

Return the builder.

```
       return builder.create();
    }
}
```

From the calling Activity and Fragment: Set an instance variable at the Activity level of the AlertDialogFragment; and show the fragment.

```
public class DialogDemoActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dialog_demo);
        if (savedInstanceState == null) {
            getFragmentManager().beginTransaction().add(R.id.container,
                                                    new PlaceholderFragment())
                    .commit();
        }
    }

...


    // An activity level variable in order to keep the AlertDialogList alive between
    // runtime
    // configuration changes (like rotating the screen), and also save state.
    static private SignInAlertCustomDialogFragment mSignInAlertCustomDialogFragment;

    /**
     * A placeholder fragment containing a simple view.
     */
    public static class PlaceholderFragment extends Fragment implements View
            .OnClickListener {

        // "Every fragment must have an empty constructor, so it can be instantiated
        // when restoring its activity's state." http://developer.android
        // .com/reference/android/app/Fragment.html
        public PlaceholderFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            View rootView = inflater.inflate(R.layout.fragment_dialog_demo, container,
                                             false);
            Button button = (Button) rootView.findViewById(R.id.button_fire_missiles);
            button.setOnClickListener(this);

            button = (Button) rootView.findViewById(R.id.button_sign_in);
            button.setOnClickListener(this);

            return rootView;
        }


        private void createAndShowSignInAlertCustomDialogFragment() {
            if (mSignInAlertCustomDialogFragment == null) {
                mSignInAlertCustomDialogFragment = new
                        SignInAlertCustomDialogFragment();
            }
            mSignInAlertCustomDialogFragment.show(getFragmentManager(),
                                                  "SignInAlertCustomDialogFragmentTag");
        }

        @Override
        public void onClick(View v) {
            switch (v.getId()) {
                case R.id.button_fire_missiles:
                    createAndShowFireMissileDialogFragment();
                    break;
                case R.id.button_sign_in:
                    createAndShowSignInAlertCustomDialogFragment();
                    break;
                default:
                    try {
                        throw new Exception("Unhandled case in switch");
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
```

```
            }
        }
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Building an Alert Dialog, Creating a Custom Layout, https://developer.android.com/guide/topics/ui/dialogs.html#CustomLayout

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\DialogDemoActivity.java

Communitcate with the host (Activity or Fragment)

From Dialog Fragment to host

You generally only need to handle user actions within the Dialog Fragment itself. But if you need to raise an event back to the Dialog's host (the calling Activity or Fragment), you can do so with a Listener Interface.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Passing Events Back to the Dialog's Host, https://developer.android.com/guide/topics/ui/dialogs.html#PassingEvents

The procedure to pass an event back to the Dialog's host ...

DialogFrament. Create a Listener Interface.

```
public class SignInAlertCustomDialogFragment extends DialogFragment {

...

    public interface SignInAlertCustomDialogFragmentListener{
        public void onDialogPositiveClick(DialogFragment dialogFragment, String someData);
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\zaradata\SignInAlertCustomDialogFragment.java

DialogFragment. Create a class level Listener variable and instantiate this using the host activity. This will ensure the host activity implements the event listener.

```
public class SignInAlertCustomDialogFragment extends DialogFragment {
....

    // Use an instance of this Listener to raise events
    SignInAlertCustomDialogFragmentListener mListener;

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        // Force the host activity to implement the Listener Interface
        try {
            // Instantiate the listener
            mListener = (SignInAlertCustomDialogFragmentListener) activity;
        } catch (ClassCastException e) {
            // In this case the Activity doesn't implement the interface
            throw new ClassCastException(
                    activity.toString() + "must implement
SignInAlertCustomDialogFragmentListener");
        }
    }
```

DialogFragment. Raise the event in the DialogFragment, using the Listener's methods.

```
public class SignInAlertCustomDialogFragment extends DialogFragment {
...

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
...

        // Setup Action buttons and respond to click.
        builder.setPositiveButton("Sign In", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

                // Cast DialogInterface to Dialog class to access UI widgets
                Dialog dialogObject = (Dialog) dialog;
                TextView userNameTextView = (TextView) dialogObject.findViewById(
                        R.id.editText_userName);
                TextView passwordTextView = (TextView) dialogObject.findViewById(
                        R.id.editText_password);
                String message = String.format("DialogFragment Signed in with %s, %s",
                                               userNameTextView.getText(),
                                               passwordTextView.getText());
                Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
                mListener.onDialogPositiveClick(SignInAlertCustomDialogFragment.this,
                                                "Oh Hi: " + userNameTextView.getText()
                                                        + passwordTextView.getText());

            }
        });
```

Host Activity. Implement the Listening Interface, and respond to the event in the implementation.

```
public class DialogDemoActivity extends Activity implements
        SignInAlertCustomDialogFragment.SignInAlertCustomDialogFragmentListener {

    @Override
    public void onDialogPositiveClick(DialogFragment dialogFragment, String someData) {
        Toast.makeText(getApplicationContext(), "Activity Event CallBack: " + someData,
                       Toast.LENGTH_SHORT).show();
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\DialogDemoActivity.java
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Passing Events Back to the Dialog's Host, https://developer.android.com/guide/topics/ui/dialogs.html#PassingEvents

From host to dialog fragment

To communicate from activity to fragment use setArguments(Bundle) and getArguments(). See also From activity to fragment

```
// Acivity
static private SignInAlertCustomDialogFragment mSignInAlertCustomDialogFragment;

// Hosting Fragment
private void createAndShowSignInAlertCustomDialogFragment() {
    Bundle bundle = new Bundle();

    // Only create if it doesn't already exist.
    if (mSignInAlertCustomDialogFragment == null) {
        mSignInAlertCustomDialogFragment = new SignInAlertCustomDialogFragment();
        bundle.putString("TITLE", "Beautiful Sign In");
        mSignInAlertCustomDialogFragment.setArguments(bundle);
    }

    // Android 3.0 (API 11) >=  android.app.Fragment.getFragmentManager();
    // Android 3.0 (API 11) < import android.support.v4.app.FragmentActivity
    // .getSupportFragmentManager()
    mSignInAlertCustomDialogFragment.show(getFragmentManager(),
```

```
                                        "SignInAlertCustomDialogFragmentTag");
}

// DialogFragment
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

    Bundle argumentBundle = getArguments();
    String title = argumentBundle.getString("TITLE");

    // Set a default title, if none supplied.
    // Checking for both zero length and null as a defensive programming practice.
    if (title == "" || title == null ) {
        title = "Sign In Dialog";
    }
```

"Default constructor. Every fragment must have an empty constructor, so it can be instantiated when restoring its activity's state. It is strongly recommended that subclasses do not have other constructors with parameters, since these constructors will not be called when the fragment is re-instantiated; instead, arguments can be supplied by the caller with setArguments(Bundle) and later retrieved by the Fragment with getArguments()."

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), Fragment, https://developer.android.com/reference/android/app/Fragment.html

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\DialogDemoActivity.java
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\zaradata\SignInAlertCustomDialogFragment.java

Showing a dialog: FullScreen V Embedded Fragment.

You can show the dialog as an embedded fragment (as usual) or as a full screen depending on the circumstances, like screen size.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Showing a Dialog Fullscreen or as an Embedded Fragment, https://developer.android.com/guide/topics/ui/dialogs.html#FullscreenDialog

This flexibility is also afforded to Activities that are conditionally dialogs.

```
<activity android:theme="@android:style/Theme.Holo.DialogWhenLarge" >
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Showing a Dialog Fullscreen or as an Embedded Fragment, https://developer.android.com/guide/topics/ui/dialogs.html#FullscreenDialog

Dismissing and Cancelling

Dismiss events and cancel events behave as follows:

	Dismiss event	**Cancel event**
Triggered when	Any action button pressed; Item on plain list (not checkboxes or radiobuttons) pressed;	Back button; Press outside dialog; cancel() called.
Intercepting methods you can implement	onDismiss()	onCancel()
Initiate the event explicitly	dismiss()	cancel()

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Dismissing a Dialog, https://developer.android.com/guide/topics/ui/dialogs.html#DismissingADialog

You should generally call dismiss when the user presses the positive button. (Why when this should occur automatically anyway???).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Dialogs, Dismissing a Dialog, https://developer.android.com/guide/topics/ui/dialogs.html#DismissingADialog

Date and Time Pickers

Time Picker

The basic coding steps for creating a Date or Time Picker (using Time as an example):

Create a custom TimePicker class that extends DialogFragment

```
public class MyTimePickerDialogFragment extends DialogFragment {

    // "Every fragment must have an empty constructor, so it can be instantiated
    // when restoring its activity's state." http://developer.android
    // .com/reference/android/app/Fragment.html
    public MyTimePickerDialogFragment() {
    }
...
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Pickers, https://developer.android.com/guide/topics/ui/controls/pickers.html

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MyTimePickerDialogFragment.java

Override the onCreateDialog() method, returning a TimePickerDialog.

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    //        super.onCreateDialog(savedInstanceState);
    final Calendar calendar = Calendar.getInstance();
    int hour = calendar.get(Calendar.HOUR_OF_DAY);
    int minute = calendar.get(Calendar.MINUTE);

    return new TimePickerDialog(getActivity(), this, hour, minute,
                                android.text.format.DateFormat.is24HourFormat(
                                        getActivity()));
}
```

Implement TimePickerDialog.OnTimeSetListener -> onTimeSet. That is, do something with the time the user picks.

```
public class MyTimePickerDialogFragment extends DialogFragment implements
        TimePickerDialog.OnTimeSetListener {

...

    @Override
    public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
        Calendar calendar = GregorianCalendar.getInstance();
        calendar.set(Calendar.HOUR_OF_DAY, hourOfDay);
        calendar.set(Calendar.MINUTE, minute);

        String message = String.format("Time chosen - %1$tH:%1$tM", calendar);
        Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
    }
}
```

In the initiating Activity show the custom time picker. No need for activity level variables of the custom time picker: runtime configuration changes (rotating screen) seems to be handled automatically (which is inconsistent with list dialogs).

```
public class DateAndTimePickersActivity extends ActionBarActivity {

....


    /**
     * A placeholder fragment containing a simple view.
     */
    public static class PlaceholderFragment extends Fragment implements View
            .OnClickListener {

        public PlaceholderFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            View rootView = inflater.inflate(R.layout.fragment_date_and_time_pickers,
                                             container, false);
            Button timePickerButton = (Button) rootView.findViewById(
                    R.id.button_TimePicker);
            timePickerButton.setOnClickListener(this);

            return rootView;
        }

        @Override
        public void onClick(View v) {
            switch (v.getId()) {
                case R.id.button_TimePicker:

                    MyTimePickerDialogFragment myTimePickerDialogFragment = new
                            MyTimePickerDialogFragment();
                    myTimePickerDialogFragment.show(getFragmentManager(),
                                                    "MyTimePickerDialogFragment");
                    break;

                default:
                    try {
                        throw new Exception("Unhandled case in switch");
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
            }

        }
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Pickers, https://developer.android.com/guide/topics/ui/controls/pickers.html

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\DateAndTimePickersActivity.java

Date Picker

Date Picker example code:

Create a custom DatePicker class that extends DialogFragment.

```
public class MyDatePickerDialogFragment extends DialogFragment implements
        DatePickerDialog.OnDateSetListener {

    // "Every fragment must have an empty constructor, so it can be instantiated
    // when restoring its activity's state." http://developer.android
```

```
    // .com/reference/android/app/Fragment.html
    public MyDatePickerDialogFragment() {
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        final Calendar calendar = Calendar.getInstance();
        int year = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH);
        int dayOfMonth = calendar.get(Calendar.DAY_OF_MONTH);

        return new DatePickerDialog(getActivity(), this, year, month, dayOfMonth);
    }

    @Override
    public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {
        Calendar calendar = GregorianCalendar.getInstance();
        calendar.set(Calendar.YEAR, year);
        calendar.set(Calendar.MONTH, monthOfYear);
        calendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);

        String message = String.format("Time chosen = %1$tY-%1$tm-%1$td", calendar);
        Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Pickers, Creating a Date Picker, https://developer.android.com/guide/topics/ui/controls/pickers.html#DatePicker

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MyDatePickerDialogFragment.java

Calll the DatePicker from your host Activity/Fragment.

```
public class DateAndTimePickersActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_date_and_time_pickers);
        if (savedInstanceState == null) {
            getFragmentManager().beginTransaction().add(R.id.container,
                                                        new PlaceholderFragment())
                    .commit();
        }
    }
...

    /**
     * A placeholder fragment containing a simple view.
     */
    public static class PlaceholderFragment extends Fragment implements View
            .OnClickListener {

        public PlaceholderFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            View rootView = inflater.inflate(R.layout.fragment_date_and_time_pickers,
                                             container, false);
            Button timePickerButton = (Button) rootView.findViewById(
                    R.id.button_TimePicker);
            timePickerButton.setOnClickListener(this);

            Button datePickerButton = (Button) rootView.findViewById(
                    R.id.button_TimePicker);
            datePickerButton.setOnClickListener(this);

            return rootView;
        }
```

```
        @Override
        public void onClick(View v) {
            switch (v.getId()) {
                case R.id.button_TimePicker:
                    MyTimePickerDialogFragment myTimePickerDialogFragment = new
                            MyTimePickerDialogFragment();
                    myTimePickerDialogFragment.show(getFragmentManager(),
                                                    "MyTimePickerDialogFragment");
                    break;

                case R.id.button_DatePicker:
                    MyDatePickerDialogFragment myDatePickerDialogFragment = new
                            MyDatePickerDialogFragment();
                    myDatePickerDialogFragment.show(getFragmentManager(),
                                                    "MyDatePickerDialogFragment");
                    break;

                default:
                    try {
                        throw new Exception("Unhandled case in switch");
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
            }
        }
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Pickers, Creating a Date Picker, https://developer.android.com/guide/topics/ui/controls/pickers.html#DatePicker

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MyDatePickerDialogFragment.java

Styles and Themes

Intro

A (generic) style is a collection of properties that specify the look and feel of a View, ViewGroup (layout), Activity, or Application.

"Style": A (generic) style applied to a View.

"Theme": A (generic) style applied to an Application, Activity or View Hierarchy (?). Themes are also applicable to non-view elements, like the status bar or window background.

A style is never applied (directly) to a ViewGroup ("layout"). Styles therefore operate like Cascading Style Sheets in web design.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Styles and Themes, https://developer.android.com/guide/topics/ui/look-and-feel/themes

A list of style attributes: R.attr

A list of theme attributes: R.styleable

Define

Style

Define a Style in res/values/styles.xml. The name of the file is arbitrary but is "styles.xml" (or perhaps "themes.xml") by convention.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="CodeFont" parent="@android:style/TextAppearance.Medium">
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>

   <style name="GreenText" parent="TextAppearance.AppCompat">
        <item name="android:textColor">#00FF00</item>
    </style>
</resources>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Styles and Themes, Create and apply a style, https://developer.android.com/guide/topics/ui/look-and-feel/themes#Styles

Theme

Define a Theme in res/values/styles.xml. The name of the file is arbitrary but is "styles.xml" (or perhaps "themes.xml") by convention. There is no "<theme>" element, you use "<style>" for Themes.

```
values/styles.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>

values/colors.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3f51b5</color>
    <color name="colorPrimaryDark">#303f9f</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Styles and Themes, Apply a style as a theme, https://developer.android.com/guide/topics/ui/look-and-feel/themes#Theme

Parent Property

When defining a (generic) style, a Style or Theme, override built in (generic) styles with an "android" prefix.

```
<style name="LightThemeSelector" parent="android:Theme.Holo.Light">
    ...
</style>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, User interface, Look and feel, Styles and Themes, https://developer.android.com/guide/topics/ui/look-and-feel/themes [Formerly http://developer.android.com/guide/topics/ui/themes.html#SelectATheme]

When defining a (generic) style, a Style or Theme, override custom (generic) styles in libraries (like android-support-v7-appcompat), without without "@android"

```
<style name="LightThemeSelector" parent="Theme.AppCompat.Light">
    ...
</style>

<!-- Old school technique -->
<style name="LightThemeSelector" parent="@style/Theme.AppCompat.Light">
    ...
</style>
```

Apply in layout.xml or AndroidManifest.mxl

Styles

Once defined apply a Style for a view in your layout xml like this ...

```
<TextView
    style="@style/CodeFont"
    android:text="@string/hello" />
```

Note the attribute is "style", not "android:style", for styles you define.

Themes

Apply a Theme for an Activity or Application in the AndroidManifest.xml like this ...

```
<!-- Application -->
<application android:theme="@style/AppTheme"> <!-- Custom, your own, having been defined -->
<!-- or -->
<application android:theme="@android:style/Theme.Holo.Light"> <!-- Built in -->

<!-- Activity -->
<activity android:theme="@style/AppTheme"> <!-- Custom, your own, having been defined -->
<!-- or -->
<activity android:theme="@android:style/Theme.Holo.Light"> <!-- Built in -->
```

Apply custom themes and themes in libraries (like android-support-v7-appcompat), with "@style" prefix.

```
<application android:theme="@style/AppTheme">
<application android:theme="@style/Theme.AppCompat.Light">
```

Apply built in themes with "@android:style" prefix.

```
<application android:theme="@android:style/Theme.Holo.Light">
```

Theme Customisation

Customise a theme by referencing, from styles.xml (or themes.xml), a theme as a parent and overriding various properties.

- Example 1:

```
values/styles.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
```

```
</resources>

values/colors.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3f51b5</color>
    <color name="colorPrimaryDark">#303f9f</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

- Example 2:

```
<resources>
    <color name="custom_theme_color">#fcf3c4</color>
    <style name="AppBaseTheme" parent="Theme.AppCompat.Light">
                <item name="android:windowBackground">@color/custom_theme_color</item>
                <item name="android:colorBackground">@color/custom_theme_color</item>
    </style>
</resources>
```

When customizing a theme you can override the theme styling for various views (UI components) by using a theme attribute "android:xxxStyle".

```
        <color name="custom_theme_color">#fcf3c4</color>
        <style name="ButtonStyleCustom" parent="@android:style/Widget.Button">
            <item name="android:textColor">#13A348</item>
    </style>
        <style name="AppBaseTheme" parent="Theme.AppCompat.Light">
                <!-- API 14 theme customizations can go here. -->
                <item name="android:windowBackground">@color/custom_theme_color</item>
                <item name="android:colorBackground">@color/custom_theme_color</item>
                <item name="android:buttonStyle">@style/ButtonStyleCustom</item>
        </style>
```

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages) R.styleable, Theme, https://developer.android.com/reference/android/R.styleable#Theme

Common "android:xxxStyle" theme attributes

```
<!-- Example theme attributes allowing style overrides -->
android:actionBarStyle
android:actionButtonStyle
android:buttonStyle
android:datePickerStyle
```

For a list of theme attributes that allow style overriding see

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages) R.styleable, Theme, https://developer.android.com/reference/android/R.styleable#Theme

Basic Themes

The basic themes, if you want to support the action bar over several Android versions:

- Theme.AppCompat (Dark Theme)
- Theme.AppCompat.Light
- Theme.AppCompat.Light.DarkActionBar

See /android-support-v7-appcompat/res/values/themes.xml

The basic themes, without requiring support for action bar over several Android versions:

- Theme.Holo (Dark Theme)
- Theme.Holo.Light
- Theme.Holo.Light.DarkActionBar

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), R.style, http://developer.android.com/reference/android/R.style.html

Holo is available from API 11, Android 3.0.x HONEYCOMB.

Theme and style setting standard practice

Define styles that apply to all API levels in AppTheme /res/values/styles.xml. Define Styles that are overrding in a later API level in AppBaseTheme in /res/values-vXX/styles.xml.

The theme and style setting standard practice allows for Support Library compatibility (e.g for AppCompat) and the targeting of different android versions. The procedure:

1. In AndroidManifest.xml set a custom theme for the application.

```
<application ... android:theme="@style/AppTheme" ... >
```

That pickes up the custom theme in /res/values/styles.xml and platform specific themes (in /res/values-v14/styles etc.)

2. In your base styles.xml (e.g. /MyFirstApp/res/values/styles.xml) set your custom theme to reference a native or support library theme.
    a. For AppCompat theme, in each style.xml, set as follows.

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
<!-- or whatever AppCompat theme you want -->
```

    b. For the Holo theme, in each style.xml, set as follows, noting the "android:style" prefix as we need to refer to the framework, not a custom library.

```
<style name="AppTheme" parent="android:style/Theme.Holo.Light">
<!-- or whatever AppCompat theme you want -->
```

3. Define theme colors (following Material Design) in values/colors.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3f51b5</color>
    <color name="colorPrimaryDark">#303f9f</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

4. Customize your AppTheme to reference previously defined colors, in values/styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
```

```
    </style>
</resources>
```

5. Define Toolbar and Popup menu styles ("themes"), in values/styles.xml

```
<style name="AppTheme.ToolbarTheme" parent="ThemeOverlay.AppCompat.Dark.ActionBar" />
<style name="AppTheme.PopupTheme" parent="ThemeOverlay.AppCompat.Light" />

<style name="AppTheme.ToolbarThemeAnother"
parent="ThemeOverlay.AppCompat.Dark.ActionBar">
    <!--<item name="android:textColorPrimary">@android:color/holo_red_light</item>-->
    <item name="android:textColorPrimary">@color/colorAccent</item>
</style>

<style name="AppTheme.PopupThemeAnother" parent="ThemeOverlay.AppCompat.Light">
    <!--<item name="android:background">@android:color/white</item>-->
    <item name="android:textColor">@android:color/black</item>
</style>
```

6. Optoinally, in your base styles.xml (e.g. res/values/styles.xml), define native overriding styles (styles that override a native style as defined in android or the support library) using the parent attribute.

```
<!-- This overrides a style defined in the support library -->
<style name="ActionBarCustom" parent="@style/Widget.AppCompat.ActionBar.Solid">
     <item name="android:background">@color/custom_actionbar_color</item>
     <!-- Support library compatibility -->
     <item name="background">@color/custom_actionbar_color</item>
 </style>

<!-- This overrides a style defined in android natively -->
<style name="ButtonStyleCustom" parent="@android:style/Widget.Button">
     <item name="android:textColor">#13a348</item>
     <item name="android:typeface">monospace</item>
</style>
```

7. Optionally, in your base styles.xml (e.g. res/values/styles.xml), define independent custom styles.

```
<style name="WarningStyleCustom">
    <item name="android:textColor">#ea1c53</item>
    <item name="android:typeface">serif</item>
</style>
```

8. For version specific themeing:

    a. In the unqalified styles.xml define a theme named "BaseAppTheme" with any desired customizations; and point to this indirectly with an "AppTheme" name.

```
<!-- /res/values/styles.xml -->
<!-- base set of styles that apply to all versions -->
<style name="BaseAppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="colorPrimary">@color/primaryColor</item>
    <item name="colorPrimaryDark">@color/primaryTextColor</item>
    <item name="colorAccent">@color/secondaryColor</item>
</style>

<style name="AppTheme" parent="BaseAppTheme">
    <!-- All customizations that are NOT specific to a
    particular API-level can go here. -->
    <item name="android:actionBarStyle">@style/ActionBarCustom</item>
    <item name="actionBarStyle">@style/ActionBarCustom</item>
</style>
```

    b. Don't define an AppTheme in higher levels for which you want to automatically get the lower level definition.

```
<!-- /res/values-V11/styles.xml -->
  <!-- No <style name="AppTheme"> defined, so previous AppTheme gets applied --
>
<!-- This will apply to all API versions above 11 unless you define an App
Theme -->
```

c. For version specific theme, extending the BaseAppTheme

```
<!-- /res/values-V14/styles.xml -->

<!--  AppTheme for API 14+. This theme extends
      BaseAppTheme from BOTH res/values/styles.xml
      on API 14+ devices. -->
<style name="AppTheme" parent="BaseAppTheme">
  <!-- API 14 theme customizations can go here. -->
  <!-- custom_theme_color defined in res/values/styles.xml  -->
  <item name="android:windowBackground">@color/custom_theme_color</item>
  <item name="android:colorBackground">@color/custom_theme_color</item>
  <!-- ButtonStyleCustom defined in res/values/styles.xml -->
  <item name="android:buttonStyle">@style/ButtonStyleCustom</item>
</style>
```

9. If <item name="android:xxxx"> prevents compiling at lower api levels, clean your project (Project > Clean ...).

Background Image

Use "android:windowBackground". If your emulator doesn't launch try moving this value higher in the order of items. Supported since API 1.

```
<!-- In /res/values/styles.xml -->

<style name="AppTheme" parent="AppBaseTheme">
    <!-- All customizations that are NOT specific to a
    particular API-level can go here. -->
    <item name="android:windowBackground">@drawable/monuriki_island_fiji</item>
    <item name="android:actionBarStyle">@style/ActionBarCustom</item>
    <item name="actionBarStyle">@style/ActionBarCustom</item>
</style>
```

Scrolling

Scrolling is something you want to do if you have more information than can fit on a screen. When enabled a user can scroll up and down; or left and right.

(Google 2019, "Android Developers > Training > Android Developer Fundamentals > Codelabs for Android Developer Fundamentals", https://developer.android.com/courses/fundamentals-training/toc-v2), "Android fundamentals 01.3: Text and scrolling views", https://codelabs.developers.google.com/codelabs/android-training-text-and-scrolling-views/index.html?index=..%2F..%2Fandroid-training#0

Scrolling can be achieved via:

- ScrollView. ScrollView is a subclass of FrameLayout. Within ScrollView you must only have one child. That child contains all the contents you want to scroll. That child may be a ViewGroup (e.g. LinearLayout).

```
<ScrollView
        android:layout width="wrap content"
        android:layout_height="wrap_content"
        android:layout_below="@id/article_heading">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout height="wrap content"
            android:orientation="vertical">
```

```
            <TextView
                android:id="@+id/article_subheading"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:padding="@dimen/padding_regular"
                android:text="@string/article_subtitle"
                android:textAppearance=
                     "@android:style/TextAppearance.DeviceDefault" />

            <TextView
                android:id="@+id/article"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:autoLink="web"
                android:lineSpacingExtra="@dimen/line_spacing"
                android:padding="@dimen/padding_regular"
                android:text="@string/article_text" />
        </LinearLayout>
</ScrollView>
```

- NestedScrollView. For vertical scrolling use this instead of ScrollView.

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), NestedScrollView, https://developer.android.com/reference/android/support/v4/widget/NestedScrollView

- RecyclerView using `android:scrollbars` xml attribute, inherited from View. Because ScrollView loads all items in memory it may choke for a large numbers of items. In this case use RecyclerView.

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

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), https://developer.android.com/reference/android/support/v7/widget/RecyclerView
(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), https://developer.android.com/reference/android/view/View#xml-attributes

- For allowing a toolbar to react to scroll events:
  - Use a CoordinatorLayout;
  - Use an AppBarLayout as a direct child of CooridinatorLayout;
  - A scrolling sibling (such as NestedScrollView or RecyclerView) with layout_behaviour of that sibling set to be an instance of AppBarLayout.ScrollingViewBehavior.
  - Have a toolbar (and/or other views) as a child of AppBarLayout. Views using the scroll flag should be declared and visually positioned before other views in the AppBarLayout.
  - For each child of AppBarLayout set desired scrolling flags.
  - Optionally set the Lift on Scroll effect. Set `app:liftOnScroll="true"` on your AppBarLayout.

```
<android.support.design.widget.CoordinatorLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
```

```
        android:layout_height="match_parent">

    <android.support.v4.widget.NestedScrollView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            app:layout_behavior="@string/appbar_scrolling_view_behavior">

        <!-- Your scrolling content -->

    </android.support.v4.widget.NestedScrollView>

    <android.support.design.widget.AppBarLayout
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            app:liftOnScroll="true">

        <android.support.v7.widget.Toolbar
                ...
                app:layout_scrollFlags="scroll|enterAlways"/>

        <android.support.design.widget.TabLayout
                ...
                app:layout_scrollFlags="scroll|enterAlways"/>

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

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

(Google n.d., "Android API Reference > Andoid Support Library", Accessed 2021-02-19. https://developer.android.com/reference/android/support/classes), AppBarLayout, https://developer.android.com/reference/android/support/design/widget/AppBarLayout

(Google, n.d., "Android Developers Blog", https://android-developers.googleblog.com/) 29 May 2015, "Android Design Support Library", "CoordinatorLayout and the app bar", https://android-developers.googleblog.com/2015/05/android-design-support-library.html

(Google 2014, "Android Design Guide", https://developer.android.com/design), "App bars: top", https://material.io/components/app-bars-top/android#using-top-app-bars

AppBar (formerly "Action Bar" using ActionBar or Toolbar)

Intro

The "App Bar" is the generic thing. An ActionBar or ToolBar are particular implementations of it.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Set up the app bar, https://developer.android.com/training/appbar/setting-up.html

The app bar basically holds:

1. (At the left in the following example):
    a. Navigation info;
    b. App Icon;
2. View Control;
3. Action Buttons (commands); and

4. Action Overflow (more commands).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Set up the app bar, https://developer.android.com/training/appbar/setting-up.html

(Google 2014, "Android Design Guide", https://developer.android.com/design), "App bars: top", https://material.io/components/app-bars-top

Toolbar (Replacement for ActionBar)

Intro

For the App bar use the support library's `Toolbar`, not the native toolbar nor the Action bar.

"you should use the support library's Toolbar class to implement your activities' app bars. Using the support library's toolbar helps ensure that your app will have consistent behavior across the widest range of devices. For example, the Toolbar widget provides a material design experience on devices running Android 2.1 (API level 7) or later, but the native action bar doesn't support material design unless the device is running Android 5.0 (API level 21) or later"

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Set up the app bar, https://developer.android.com/training/appbar/setting-up.html

The Toolbar was created in Android 5.0 API 21 and the AppCompat V7, 21 library (released with Android 5.0/API 21) to enable you to do some things that the ActionBar can't do. It serves as a replacement for the ActionBar and can take over most of the existing code for an Action bar (e.g. menu click handling).

(Banes 2014, "AppCompat V21 – Material Design for Pre-Lollipop Devices!", https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html) https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html

A Toolbar can take over Action Bar functions while also handling inheritance without problem.

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoExtendedActivity.java

The Toolbar can be used in two ways:

1. As an ActionBar: for when you want to use existing ActionBar featues (menu inflation, menu item handling, up button, etc) and you want more control over ActionBar appearance.
2. Standalone: For behaviour that an ActionBar wouldn't support, such as multiple toolbars on screen; displaying the toolbar on less than the full width of the screen.

(Banes 2014, "AppCompat V21 – Material Design for Pre-Lollipop Devices!", https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html) https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html

The best practice for new apps is to use an Toolbar rather than an ActionBar, in order to have greater control over appearance from the start. If you want to use Material design with a light them but a dark ActionBar (with light text) ... this is only achievable with a Toolbar, for example.

A John Bentley principle discovered through experiment.

Toolbar Setup

To use the Toolbar as an App Bar:

1. Add the AppCompat library as a dependency in your build.gradle (Module:app) file.

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.1.0'
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Set up the app bar, https://developer.android.com/training/appbar/setting-up.html

(Banes 2014, "AppCompat V21 – Material Design for Pre-Lollipop Devices!", https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html) https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

2. Extend your Activity from AppCompatActivity

```
public class MenusDemoActivity extends AppCompatActivity
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Set up the app bar, https://developer.android.com/training/appbar/setting-up.html

(Banes 2014, "AppCompat V21 – Material Design for Pre-Lollipop Devices!", https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html) https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

3. Your AndroidManifest.xml must reference a custom theme.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="au.com.softmake.mysecondapp" >

...

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity ...
```

(Banes 2014, "AppCompat V21 – Material Design for Pre-Lollipop Devices!", https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html) https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\AndroidManifest.xml

4. Your custom theme must extend "Theme.AppCompat.NoActionBar" or "Theme.AppCompat.Light.NoActionBar"

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">#3F51B5</item>

        <!--    darker variant of colorPrimary (for status bar, contextual app bars) -
->
        <item name="colorPrimaryDark">#303F9F</item>

        <!--   theme UI controls like checkboxes and text fields -->
        <item name="colorAccent">#ffab40</item>
    </style>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Set up the app bar, https://developer.android.com/training/appbar/setting-up.html

(Banes 2014, "AppCompat V21 – Material Design for Pre-Lollipop Devices!", https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html) https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\values\styles.xml

5. Provide standard material design colors. Provide Toolbar custom themes and styles (for the popup toolbar too).

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">#3F51B5</item>

        <!--    darker variant of colorPrimary (for status bar, contextual app bars) -
->
        <item name="colorPrimaryDark">#303F9F</item>

        <!--   theme UI controls like checkboxes and text fields -->
        <item name="colorAccent">#ffab40</item>
    </style>

    <!-- ToolBar -->
    <style name="AppTheme.ToolbarTheme" parent="ThemeOverlay.AppCompat.Dark.ActionBar"
/>
    <style name="AppTheme.PopupTheme" parent="ThemeOverlay.AppCompat.Light" />

    <!-- or, for further customization (and your layout.xml files must be changed to
reference this theme -->
    <style name="AppTheme.ToolbarThemeAnother"
parent="ThemeOverlay.AppCompat.Dark.ActionBar">
        <!--<item
name="android:textColorPrimary">@android:color/holo_red_light</item>-->
        <item name="android:textColorPrimary">@color/colorAccent</item>
    </style>

    <style name="AppTheme.PopupThemeAnother" parent="ThemeOverlay.AppCompat.Light">
        <!--<item name="android:background">@android:color/white</item>-->
        <item name="android:textColor">@android:color/black</item>
    </style>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\values\styles.xml

6. Create a toolbar instance, usually in your layout XML. You need to nest it an ViewGroup for standard padding. Reference two custom theme's

```
<LinearLayout 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:orientation="vertical"
    tools:context="au.com.softmake.mysecondapp.ActionBarDemo">

    <android.support.v7.widget.Toolbar
        android:id="@+id/my_toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        app:theme="@style/AppTheme.Toolbar"
        app:popupTheme="@style/AppTheme.Popup"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin">

        <ToggleButton
            android:id="@+id/toggleButton_action_help_available"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:checked="true" />
...
```

(Banes, 2014. AppCompat v21 — Material Design for Pre-Lollipop Devices!) http://android-developers.blogspot.com.au/2014/10/appcompat-v21-material-design-for-pre.html
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\activity_menus_demo.xml

7. In your Activity or Fragment's onCreate() set the ToolBar as your ActionBar.

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

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

(Banes 2014, "AppCompat V21 — Material Design for Pre-Lollipop Devices!", https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html) https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

8. Set your UP button after the Toolbar.
    a. In AndroidManifest.xml declare a parent activity.

```
<activity android:name=".ToolbarDemoActivity"

android:parentActivityName="au.com.softmake.mythirdapp.StartSettingsActivity" >
```

```
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="au.com.softmake.mythirdapp.StartSettingsActivity" />
</activity>
```

b. Enable the UP button in code ...

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

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

        // Must come after setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
```

(Banes 2014, "AppCompat V21 – Material Design for Pre-Lollipop Devices!", https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html) https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

For more styling options on the toolbar see ....

(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/) "Change Toolbar color in Appcompat 21", answer by Gabriele Mariotti at 2014-Oct-25 http://stackoverflow.com/a/26561223/872154

Drop Down Navigation (Spinner) for the Toolbar

Todo. See
1. http://developer.android.com/guide/topics/ui/actionbar.html#Dropdown

2. http://stackoverflow.com/questions/26906065/android-spinner-and-toolbar-android-support-v7-widget-toolbar

ActionBar (Deprecated but todo)

Transiation to newer versions. Replace any `ActionBarActivity` with `AppCompatActivity`; and change imports from:

- `import android.support.v7.app.ActionBarActivity;` to
- `import android.support.v7.app.AppCompatActivity;`

```
// From
import android.support.v7.app.ActionBarActivity;
public class ShapeResourcesActivity extends ActionBarActivity {

// To
import android.support.v7.app.AppCompatActivity;
public class ShapeResourcesActivity extends AppCompatActivity {
```

https://inducesmile.com/android-tutorials-for-nigerian-developer/how-to-fix-android-support-v7-actionbaractivity-is-deprecated/ "How to fix android.support.v7.actionbaractivity is deprecated".

Common

Setup

Setup the action bar for Android 2.1 (API Level 7) and greater:

Add the v7 appcompat Android Support Library (see above).

In Java: Change class to extend ActionBarActivity, rather than Activity.

```
package au.com.softmake.mysecondapp;

import android.support.v7.app.ActionBarActivity;
...

public class ActionBarDemo extends ActionBarActivity {
```

In AndroidManifest.xml set the theme for the Application or individual ActionBarActivity. For Android 2.1 (API Level 7) and above set Theme.AppCompat.

```
// Direct Application
<application ... android:theme="@style/Theme.AppCompat.Light" ... >

// Direct Activity
<activity ... android:theme="@style/Theme.AppCompat.Light" ... >


// Indirection Application (Recommended)
<!-- AndroidManifest.xml -->
<application ... android:theme="@style/AppTheme" ... >

<!-- res\values\styles.xml -->
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light">
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Set the API level in AndroidManifest (Or, rather, use Android Studio > Project Structure > Modules > App > Flavours > [Relevant Config > Min SDK Version; which sets a gradle value; which in turn sets the AndroidManifest).

```
<manifest ... >
    <uses-sdk android:minSdkVersion="7"  android:targetSdkVersion="18" />
    ...
</manifest>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Set up the app bar, https://developer.android.com/training/appbar/setting-up.html

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\ActionBarDemo.javas

Setup the action bar for Android 3.0 (API Level 11) and greater: All Activities get an ActionBar for free, if a Theme.Holo (or one of its descendants) or Theme.AppCompat (or one of its descendants) is used:

In Java: Your class just extends Activity.

```
package au.com.softmake.mysecondapp;
```

```
import android.app.Activity;
...

public class ActionBarDemo extends Activity {
```

In AndroidManifest.xml set the theme for the Application or individual ActionBarActivity. You can use Theme.Holo (Preferred) or Theme.AppCompat, or the descendants of those. Beware of the need to change referencing syntax (e.g. "@style/" V "@android:style/"; "" V "android:style/'") when choosing an AppCompat theme V Holo Them.

```
// Direct Application
<application ... android:theme="@android:style/Theme.Holo.Light.DarkActionBar" ... >

// Direct Activity
<activity ... android:theme="@android:style/Theme.Holo.Light.DarkActionBar"... >


// Indirection Application (Recommended)
<!-- AndroidManifest.xml -->
<application ... android:theme="@style/AppTheme" ... >

<!-- res\values\styles.xml -->
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="android:style/Theme.Holo.Light">
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Set the API level in AndroidManifest. (Or, rather, use Android Studio > Project Structure > Modules > App > Flavours > [Relevant Config > Min SDK Version; which sets a gradle value; which in turn sets the AndroidManifest).

```
<manifest ... >
    <uses-sdk android:minSdkVersion="11"  android:targetSdkVersion="18" />
    ...
</manifest>
```

Theme.Holo is the default theme when you set `targetSdkVersion` or `minSdkVersion` attribute is set to "11" or greater (in AndroidManifest.xml)

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Set up the app bar, https://developer.android.com/training/appbar/setting-up.html

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

Icons and Logos

Generally you accept the application icon as specified in AndroidManifest.xml, with the icon attribute of <application> or <activity>.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

In AndroidManifest.xml <application> and <activity> also have a logo attribute which, if set, overrides the icon attribute. A "logo should usually be wider than the icon. You should generally use a logo only when it represents your brand in a traditional format that users recognize."

"A good example is the YouTube app's logo—the logo represents the expected user brand, whereas the app's icon is a modified version that conforms to the square requirement for the launcher icon."

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

Adding a Menu with Action Items (commands)

See Menus.

The Up Button

The up button, a misnomer given the button points sideways (to the left), allows the user to navigate to a parent activity. The back button, by contrast, goes backward through the activities that the user has visited in chronological order.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Add an up action, https://developer.android.com/training/appbar/up-action

To setup an up button: ...

... declare a parent activity in the manifest

```
<!-- In AndroidManifest.xml -->
<application ... >
    ...
    <!-- The main/home activity (it has no parent activity) -->
    <activity
        android:name="com.example.myfirstapp.MainActivity" ...>
        ...
    </activity>
    <!-- A child of the main activity -->
    <activity
        android:name="com.example.myfirstapp.DisplayMessageActivity"
        android:label="@string/title_activity_display_message"
        android:parentActivityName="com.example.myfirstapp.MainActivity" >
        <!-- Parent activity meta-data to support 4.0 and lower -->
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.myfirstapp.MainActivity" />
    </activity>
</application>
```

... and enable the UP button in the Activities onCreate method

```
<!-- Autogenerated code -->
public class DisplayMessageActivity extends ActionBarActivity {
...
protected void onCreate(Bundle savedInstanceState)  {
          super.onCreate(savedInstanceState);
          // Show the Up button in the action bar.
          setupActionBar();
...

/**
 * Set up the {@link android.app.ActionBar}, if the API is available.
 */
@TargetApi(Build.VERSION_CODES.HONEYCOMB)
private void setupActionBar() {
          if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
                    getActionBar().setDisplayHomeAsUpEnabled(true);
          }
}
```

This was verified to work on API 10, API 19, and API 16. Specifically with a API 10 emulator the UP button was enabled, as expected.

Note the textbook has the following conflicting code

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity displaymessage);

    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    // If your minSdkVersion is 11 or higher, instead use:
    // getActionBar().setDisplayHomeAsUpEnabled(true);
}
```

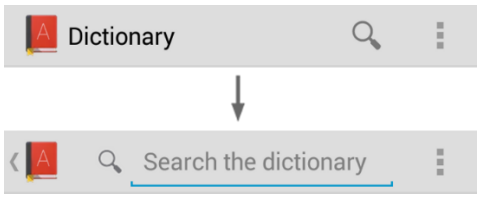
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Add an up action, https://developer.android.com/training/appbar/up-action

Rare

Action Views

An action view is a view (a UI component) on the App Bar, instead of an Action button (when expanded).

```
E.g. An Embedded Search view in the Action View ...
```



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Use action views and action providers, Add an Action View, https://developer.android.com/training/appbar/action-views#action-view

Action View coding procedure:

In the menu.xml specify either: a layout resource with `android:actionLayout`; or a widget class with `android:actionViewClass`. Optoinally include a "collapseActionView" in the `showAsAction` attribute.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
     xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context="au.com.softmake.mysecondapp.MenusDemoActivity">
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:showAsAction="ifRoom|collapseActionView"
        app:actionViewClass="android.widget.SearchView"
        android:actionViewClass="android.widget.SearchView"
        android:title="@string/action_search"
        tools:ignore="AppCompatResource" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Use action views and action providers, Add an Action View, https://developer.android.com/training/appbar/action-views#action-view

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\menu\menus_demo.xml

In your Activity (or Fragment) onCreateOptionsMenu() you'll generally want to retrieve and configure both the menu item and the action view embedded in the menu item.

```
public class MenusDemoActivity extends ActionBarActivity

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

    // Configure the menu item
    MenuItem searchMenuItem = menu.findItem(R.id.action_search);
    MenuItemCompat.setOnActionExpandListener(searchMenuItem,
                                             new MenuItemCompat.OnActionExpandListener() {
        @Override
        public boolean onMenuItemActionExpand(MenuItem item) {
            Toast.makeText(getApplicationContext(), "Expanded",
                           Toast.LENGTH_SHORT).show();
            return true; // Return true to collapse action view

        }

        @Override
        public boolean onMenuItemActionCollapse(MenuItem item) {
            Toast.makeText(getApplicationContext(), "Collapsed",
                           Toast.LENGTH_SHORT).show();
            return true;  // Return true to expand action view

        }
    });

    // Configure the action view embedded in the menu item
    SearchView searchView = (SearchView) MenuItemCompat.getActionView(searchMenuItem);
    // When not using AppCompat and for APIs > 11.
//    SearchView searchView = (SearchView) searchMenuItem.getActionView();

    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String query) {
            Toast.makeText(getApplicationContext(), "Searched " + query,
                           Toast.LENGTH_SHORT).show();
            return true;
        }

        @Override
        public boolean onQueryTextChange(String newText) {
            return false;
        }
    });

    return true;
}
```

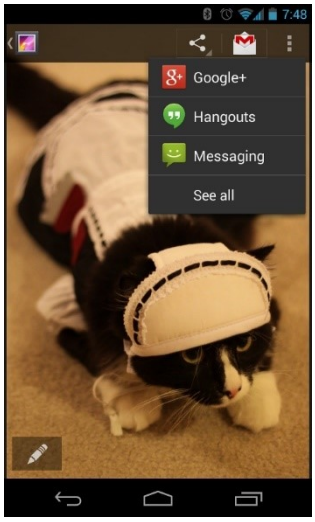
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Use action views and action providers, Add an Action View, https://developer.android.com/training/appbar/action-views#action-view

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

Action Providers

Basics

An Action Provider: replaces an action button with a customized layout; takes control of the action's behaviour; and can display a submenu.



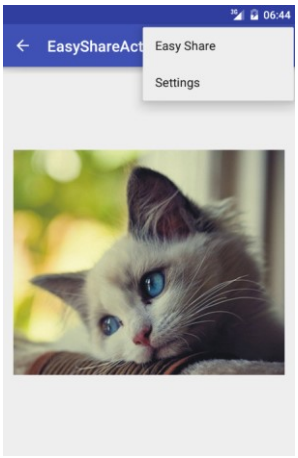
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Use action views and action providers, Add an Action Provider, https://developer.android.com/training/appbar/action-views#action-provider

You can create your own Action Provider (extend the ActionProvider class) or use some of the built in ActionProviders (e.g. ShareActionProvider).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Use action views and action providers, Add an Action Provider, https://developer.android.com/training/appbar/action-views#action-provider

ActionProviders are available natively from Android 4.0 (API Level 14) but can be made backward compatiable with android.support.v7.widget.ShareActionProvider.

Action Provider Procedure (ShareActionProvider example)



To use an Action Provider (we'll use a ShareActionProvider as the example) do the following:

In your menu layout xml set actionProviderClass to ShareActionProvider. Pay attention to the xml prefix, it is "app" for the support library or "android" for the native reference.

```
<!-- \res\menu\easy_share_action_demo.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="au.com.softmake.mysecondapp.EasyShareActionDemo" >
    <item android:id="@+id/action_easy_share"
        android:title="@string/easy_share"
        app:actionProviderClass="android.support.v7.widget.ShareActionProvider"
        android:orderInCategory="50"
        app:showAsAction="ifRoom" />
...
</menu>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Use action views and action providers, Add an Action Provider, https://developer.android.com/training/appbar/action-views#action-provider

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add data & files, Sharing simple data, overview, https://developer.android.com/training/sharing

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\menu\easy_share_action_demo.xml

Create a function which returns the intent you are interested in. Set all your desired EXTRAS, ACTIONS, etc.

```
public class EasyShareActionDemo extends ActionBarActivity {
...
    private Intent getShareIntent() {
        // Implicit intent.
        Intent sendIntent = new Intent();
        sendIntent.putExtra(Intent.EXTRA_TEXT, "Text to share");
        sendIntent.setAction(Intent.ACTION_SEND);
        sendIntent.setType("text/plain");
        return sendIntent;
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\EasyShareActionDemo.java

In your Activity, onCreateOptionsMenu, reference the ShareActionProvider of the menu item and set the Share Intent

```
    private android.support.v7.widget.ShareActionProvider mShareActionProvider;

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

        MenuItem shareItem = menu.findItem(R.id.action_easy_share);
        mShareActionProvider = (android.support.v7.widget.ShareActionProvider)
                MenuItemCompat.getActionProvider(
                shareItem);
        mShareActionProvider.setShareIntent(getShareIntent());
        return true;
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Use action views and action providers, Add an Action Provider, https://developer.android.com/training/appbar/action-views#action-provider

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add data & files, Sharing simple data, overview, https://developer.android.com/training/sharing

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\EasyShareActionDemo.java

You don't need to handle user interaction in onOptionsItemsSelected(). However, you can if you want, in order to fire a simultaneous action. But return false so that android processes the Action Provider nevertheless.

```
@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();
        switch (id) {
            case R.id.action_easy_share:
                Toast.makeText(getApplicationContext(), "Firing easy share intent",
                               Toast.LENGTH_SHORT).show();

                return false;
            case R.id.action_settings:
                Toast.makeText(getApplicationContext(), "Not implemented",
                               Toast.LENGTH SHORT).show();
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Use action views and action providers, Add an Action Provider, https://developer.android.com/training/appbar/action-views#action-provider

Custom Action Provider

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Use action views and action providers, Add an Action Provider, https://developer.android.com/training/appbar/action-views#action-provider

Hide Action Bar

You can show and hide the Action bar.

```
// Api level 7 and higher
ActionBar actionBar = getSupportActionBar();
actionBar.hide();
...
actionBar.show();

// Api level 11 and higher
ActionBar actionBar = getActionBar();
actionBar.hide();
...
actionBar.show();
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

When hiding and showing the Action Bar you might want to use overlay mode. This draws the action bar in front of your activity layout. To do so create a custom theme for your activity and set `windowActionBarOverlay` to True.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

Overlaying the Action Bar

To make the action bar background transparent see Colors.

To set a background image see Background Image.

To overlay your actionbar, rather than by default pushing the window beneath the actionbar, use android:windowActionBarOverlay

```
<style name="AppTheme" parent="AppBaseTheme">
    <!-- All customizations that are NOT specific to a
    particular API-level can go here. -->
    <item name="android:windowBackground">@drawable/monuriki_island_fiji</item>
    <item name="android:actionBarStyle">@style/ActionBarCustom</item>
    <item name="actionBarStyle">@style/ActionBarCustom</item>
</style>

<style name="ActionBarCustom" parent="@style/Widget.AppCompat.ActionBar">
    <item name="android:titleTextStyle">@style/ActionBarTitleTextCustom</item>
    <item name="titleTextStyle">@style/ActionBarTitleTextCustom</item>

    <item name="android:background">@color/actionbarColorCustom</item>
    <!-- Support library compatibility -->
    <item name="background">@color/actionbarColorCustom</item>

    <item name="android:windowActionBarOverlay">True</item>
     <!-- Support library compatibility -->
     <item name="windowActionBarOverlay">True</item>
</style>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

If the action bar, in overlay mode, obscures any of your UI views then you may like to set the paddingTop

```
<!-- Support library compatibility -->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="?attr/actionBarSize">
    ...
</RelativeLayout>
```

In practice I found this uncessary.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

Using a Split Action Bar

This seems to be deprecated in Material Design ...

(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/) http://stackoverflow.com/questions/27046951/split-action-bar-with-appcompat-v7-using-material-design

Styling the Action Bar

Set a desired android theme. See Basic Themes above.

The http://www.actionbarstylegenerator.com/ is a useful tool (open up in Chrome only).

From Android 2.1 (API level 7) you can use a support library, otherwise, from Android 3.0 (API level 11) you can use the Holo theme.

For the action bar background.

```
<!-- Using Holo from Android 3.0 (API level 11) -->
<!-- the theme applied to the application or activity -->
<style name="CustomActionBarTheme"
       parent="@style/Theme.Holo.Light.DarkActionBar">
    <item name="android:actionBarStyle">@style/MyActionBar</item>
</style>

<!-- ActionBar styles -->
<style name="MyActionBar"
       parent="@style/Widget.Holo.Light.ActionBar.Solid.Inverse">
    <item name="android:background">@drawable/actionbar_background</item>
</style>


<!-- Using support library from Android 2.1 (API level 7) -->
<style name="AppTheme" parent="AppBaseTheme">
    <!-- All customizations that are NOT specific to a
    particular API-level can go here. -->
    <item name="android:actionBarStyle">@style/ActionBarCustom</item>
    <item name="actionBarStyle">@style/ActionBarCustom</item>
</style>

<style name="ActionBarCustom" parent="@style/Widget.AppCompat.ActionBar.Solid">
    <item name="android:background">@color/custom_actionbar_color</item>
    <!-- Support library compatibility -->
    <item name="background">@color/custom_actionbar_color</item>
</style>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar [Formerly http://developer.android.com/training/basics/actionbar/styling.html#CustomBackground]

For the action bar title text

```
<!-- Using Holo from Android 3.0 (API level 11) -->
<!-- the theme applied to the application or activity -->
<style name="CustomActionBarTheme"
       parent="@style/Theme.Holo">
    <item name="android:actionBarStyle">@style/MyActionBar</item>
    <item name="android:actionBarTabTextStyle">@style/MyActionBarTabText</item>
    <item name="android:actionMenuTextColor">@color/actionbar_text</item>
</style>

<!-- ActionBar styles -->
<style name="MyActionBar"
       parent="@style/Widget.Holo.ActionBar">
    <item name="android:titleTextStyle">@style/MyActionBarTitleText</item>
</style>

<!-- ActionBar title text -->
<style name="MyActionBarTitleText"
       parent="@style/TextAppearance.Holo.Widget.ActionBar.Title">
    <item name="android:textColor">@color/actionbar_text</item>
</style>

<!-- ActionBar tabs text styles -->
<style name="MyActionBarTabText"
       parent="@style/Widget.Holo.ActionBar.TabText">
    <item name="android:textColor">@color/actionbar_text</item>
</style>

<!-- Using support library from Android 2.1 (API level 7) -->
<style name="AppTheme" parent="AppBaseTheme">
    <!-- All customizations that are NOT specific to a
    particular API-level can go here. -->
    <item name="android:actionBarStyle">@style/ActionBarCustom</item>
    <item name="actionBarStyle">@style/ActionBarCustom</item>
```

```
</style>

<style name="ActionBarCustom" parent="@style/Widget.AppCompat.ActionBar">
    <item name="android:titleTextStyle">@style/ActionBarTitleTextCustom</item>
    <!-- Support library compatibility -->
     <item name="titleTextStyle">@style/ActionBarTitleTextCustom</item>
    <item name="android:background">@color/actionbarColorCustom</item>
    <!-- Support library compatibility -->
    <item name="background">@color/actionbarColorCustom</item>
</style>

<style name="ActionBarTitleTextCustom"
parent="@style/TextAppearance.AppCompat.Widget.ActionBar.Title">
    <item name="android:textColor">@color/actionbarTextColorCustom</item>
    <!--  android:textColor is backward compatible with support library -->
</style>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar [Formerly http://developer.android.com/training/basics/actionbar/styling.html#CustomText]

Todo http://developer.android.com/training/basics/actionbar/styling.html#CustomTabs

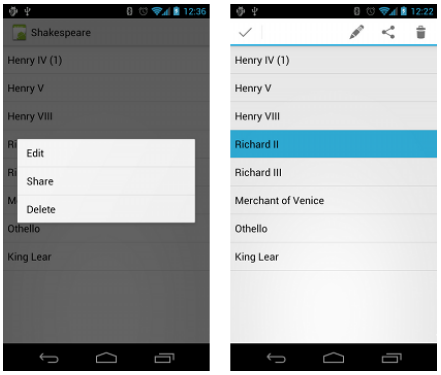
Menus

Overview

As a matter of design, from Android 3.0 (API 11): you should provide an app bar, with action buttons and an overflow menu; and the device based menu button is deprecated.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, https://developer.android.com/guide/topics/ui/menus

There are three types of menu:

Menu Type	Purpose	Notes	Screen Shot.
Options Menu	Holds menu items for an Activity that have a global impact on the app (e.g. Search, Compose email, Settings).	From Android 3.0 (API 11) this is presented on the action bar as action buttons and an overflow menu.	
Context Menu	For actions that affect the selected context or context frame.	- Before Android 3.0: floating menus (although you still might like to use them). - From Android 3.0: Contextual action mode.	
Pop up menu	List of items with respect to a view (e.g. Button or Text). Relates to	Good for "an overflow" of actions relating to content or options for a second part of a command.	

	content but does not affect content.		

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, https://developer.android.com/guide/topics/ui/menus

Coding procedure

Common

General

Regardless of the type of menu:

(If you are using icons) Download Icon pack from http://developer.android.com/design/downloads/index.html#action-bar-icon-pack

(If you are using icons) Install your selected icons to your project's res/drawable directory. Define the action buttons in an XML menu resource (for Android 3.0 API 11 and higher)

```
<!-- Defined in res/menu/main_actions.xml -->
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <!-- Appear as action button, if room. -->
    <item android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:title="@string/action_search"
        android:showAsAction="ifRoom"
        tools:ignore="AppCompatResource" />
    <!-- Force into overflow -->
    <item android:id="@+id/action_settings"
        android:icon="@drawable/ic_action_settings"
        android:title="@string/action_settings"
        android:showAsAction="never"
        tools:ignore="AppCompatResource" />
</menu>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, https://developer.android.com/guide/topics/ui/menus

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Add and handle actions, https://developer.android.com/training/appbar/actions

Define actions if using the Appcompat library to support Android 2.1 (API level 7) and higher (including Android 3.0 API 11 and higher). Use a namespace uri: http://schemas.android.com/apk/res-auto. Use a custom namespace declared prefix (the conventions is to use your app name). Change the declared prefix for `showAsAction` to match that which you have defined.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto" >
    <!-- Appear as action button, if room. -->
    <item android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:title="@string/action_search"
        app:showAsAction="ifRoom" />
    <!-- Force into overflow -->
    <item android:id="@+id/action_settings"
        android:icon="@drawable/ic_action_settings"
        android:title="@string/action_settings"
        app:showAsAction="never"
```

```
</menu>
```

In your activity should extend Activity (for Android 3.0 (API 11) or greater) or ActionBarActivity (if you want to use features in the AppCombat library, like material design theming for an ActionBar in versions prior to Android 5.0 (API 21).

```
// For Android 3.0 (API 11) without using AppCompat library.
import android.app.Activity;
....

public class MenusDemo extends Activity {


// Use AppCompat library
import android.support.v7.app.ActionBarActivity;

public class MenusDemo extends ActionBarActivity {
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Set up the app bar, https://developer.android.com/training/appbar/setting-up

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Add and handle actions, https://developer.android.com/training/appbar/actions

(Banes 2014, "AppCompat V21 — Material Design for Pre-Lollipop Devices!", https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html) https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html

Use Java code to "inflate" your xml defined action buttons into the action bar as follows …

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

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Add and handle actions, https://developer.android.com/training/appbar/actions

Defining the menu

Three elements comprise a menu XML file:

- <menu>. Root node for menu.
- <item>. Create a menu item. This may contain a nested <menu> to create a submenu.
- <group>. A container used for setting properties in Java against a group of items. Does not directly relate to creating submenus.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context="au.com.softmake.mysecondapp.MenusDemoActivity">
    <item
        android:id="@+id/action_search"
        app:actionViewClass="android.widget.SearchView"
        android:actionViewClass="android.widget.SearchView"
        android:icon="@drawable/ic_action_search"
        app:showAsAction="ifRoom|collapseActionView"
        android:showAsAction="ifRoom|collapseActionView"
        android:title="@string/action_search"
```

```
        tools:ignore="AppCompatResource" />
    <item
        android:id="@+id/action_settings"
        android:icon="@drawable/ic_action_settings"
        android:orderInCategory="100"
        app:showAsAction="ifRoom"
        android:showAsAction="ifRoom"
        android:title="@string/action_settings"
        tools:ignore="AppCompatResource" />

    <item
        android:id="@+id/action_plane_submenu"
        app:showAsAction="ifRoom"
        android:showAsAction="ifRoom"
        android:title="@string/action_plane_submenu"
        tools:ignore="AppCompatResource">
        <menu>
            <group
                android:id="@+id/menu_group_plane"
                android:checkableBehavior="all">
                <item
                    android:id="@+id/action_plane_boeing"
                    android:checked="true"
                    app:showAsAction="ifRoom"
                    android:showAsAction="never"
                    android:title="@string/action plane boeing"
                    tools:ignore="AppCompatResource" />
                <item
                    android:id="@+id/action_plane_airbus"
                    app:showAsAction="ifRoom"
                    android:showAsAction="never"
                    android:title="@string/action plane airbus"
                    tools:ignore="AppCompatResource" />
            </group>
        </menu>
    </item>

    <group android:id="@+id/menu group informational">
        <item
            android:id="@+id/action_about"
            app:showAsAction="ifRoom"
            android:showAsAction="ifRoom"
            android:title="@string/action_about"
            tools:ignore="AppCompatResource" />
        <item
            android:id="@+id/action_help"
            app:showAsAction="ifRoom"
            android:showAsAction="ifRoom"
            android:title="@string/action_help"
            tools:ignore="AppCompatResource" />
    </group>
</menu>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Defining a Menu in XML, https://developer.android.com/guide/topics/ui/menus#xml

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\menu\menus_demo.xml

Menu (Item) Groups

General

Grouping menu items allows you to share characteristics and operate on all the items with the group at once (e.g. Toogle visibility, enabled, checkable).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus,Creating Menu Groups, https://developer.android.com/guide/topics/ui/menus#groups

Define a group of items in xml with the <group> element.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="au.com.softmake.mysecondapp.MenusDemo">
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:title="@string/action_search"
        android:showAsAction="ifRoom"
        tools:ignore="AppCompatResource" />
    <item
        android:id="@+id/action_settings"
        android:icon="@drawable/ic_action_settings"
        android:title="@string/action_settings"
        android:orderInCategory="100"
        android:showAsAction="never"
        tools:ignore="AppCompatResource" />

    <item
        android:id="@+id/action_plane_submenu"
        android:title="@string/action_plane_submenu"
        android:showAsAction="never"
        tools:ignore="AppCompatResource">
        <menu>
            <item
                android:id="@+id/action_plane_boeing"
                android:title="@string/action plane boeing"
                android:showAsAction="never"
                tools:ignore="AppCompatResource" />
            <item
                android:id="@+id/action_plane_airbus"
                android:title="@string/action_plane_airbus"
                android:showAsAction="never"
                tools:ignore="AppCompatResource" />
        </menu>
    </item>

    <group android:id="@+id/menu_group_informational">
        <item
            android:id="@+id/action_about"
            app:showAsAction="ifRoom"
            android:showAsAction="ifRoom"
            android:title="@string/action_about"
            tools:ignore="AppCompatResource" />
        <item
            android:id="@+id/action_help"
            app:showAsAction="ifRoom"
            android:showAsAction="ifRoom"
            android:title="@string/action_help"
            tools:ignore="AppCompatResource" />
    </group>
</menu>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus,Creating Menu Groups, https://developer.android.com/guide/topics/ui/menus#groups

Items within a group remain sibling with menu items outside the group (at the same level).

```
In the above @+id/action_about is a sibling with @+id/action_settings
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus,Creating Menu Groups, https://developer.android.com/guide/topics/ui/menus#groups

Operate on Menu (item) group

Operating on a group of menu items at once.

```
// XML Menu as above.

public class MenusDemoActivity extends Activity implements CompoundButton
```

```
        .OnCheckedChangeListener, View.OnClickListener, View.OnLongClickListener,
        PopupMenu.OnMenuItemClickListener {

    private ToggleButton mToggleButtonMenuItemGroupEnabled;

   @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        String message = "";

            case R.id.toggleButton_menu_item_group_enabled:
                if (isChecked) {
                    message = "toggleButton_menu_item_group_enabled on";
                } else {
                    message = "toggleButton_menu_item_group_enabled off";
                }
                // In effect calls onCreateOptionsMenu() then opPrepareOptionsMenu().
                // See: http://developer.android.com/guide/topics/ui/menus
                // .html#ChangingTheMenu
                invalidateOptionsMenu();

            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    }

    /**
     * For changes to the menu after initial creation.
     */
    @Override
    public boolean onPrepareOptionsMenu(Menu menu) {
        if (mToggleButtonActionHelpAvailable.isChecked()) {
            // Do nothing. The system restores the original menu structure.
            // as invalidateOptionsMenu() has already called onCreateOptionsMenu();
        } else {
            menu.removeItem(R.id.action_help);
        }

        menu.setGroupEnabled(R.id.menu_group_informational,
                             mToggleButtonMenuItemGroupEnabled.isChecked());
        return super.onPrepareOptionsMenu(menu);
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus,Creating Menu Groups, https://developer.android.com/guide/topics/ui/menus#groups

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

Make Menu (item) group checkable

To make a menu (item) group checkable, in XML set android:checkableBehavior. "all" for independent checkability, "single" for radio button behaviour. Set a default value for a menu item with android:checked.

```
...
<menu>
    <group android:id="@+id/menu_group_plane" android:checkableBehavior="all" >
        <item
            android:id="@+id/action_plane_boeing"
            android:checked="true"
            android:showAsAction="never"
            android:title="@string/action_plane_boeing"
            app:showAsAction="ifRoom" />
        <item
            android:id="@+id/action_plane_airbus"
            android:showAsAction="never"
            android:title="@string/action_plane_airbus"
```

```
            app:showAsAction="ifRoom" />
    </group>
</menu>
...
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus,Creating Menu Groups, Using checkable menu items, https://developer.android.com/guide/topics/ui/menus.html#checkable

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\menu\menus_demo.xml

When handling a group checkable menu item you need to explicitly set the new value. You can achieve this with `item.setChecked(!item.isChecked())`.

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
..
        switch (item.getItemId()) {
..
            case R.id.action_plane_airbus:
                // This proves that a submenu item selected event can be reached.
                Toast.makeText(getApplicationContext(), "Airbus",
                               Toast.LENGTH_SHORT).show();
                // Checked state otherwise not preserved by the system throughout the
                // session (and the state is not preserved across sessions without
                // explicitly saving to Shared Preferences. Checked menu items are not
                // meant to be preserved across sessions as a general rule.
                item.setChecked(!item.isChecked());
                return true;

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

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus,Creating Menu Groups, Using checkable menu items, https://developer.android.com/guide/topics/ui/menus.html#checkable

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

Dynamically include Implicit Intents

You can dynamically include an Activity in a menu. Generally the included menu item would launch an Activity. Do the following:

- In `AndroidManifest.xml` include `category.ALTERNATIVE` and `category.SELECTED_ALTERNATIVE` in your intent filter (you must also include at least one Action otherwise the Activity will not match).

```
<activity
    android:name=".DialogDemoActivity"
    android:label="@string/title_activity_dialog_demo"
    android:parentActivityName=".MainActivity">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="au.com.softmake.mysecondapp.MainActivity" />
    <intent-filter>
        <!-- Must have at least one action even when not specified in source
        intent otherwise the activity will not match -->
        <action android:name="android.intent.action.VIEW" />

        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.ALTERNATIVE" />
        <category android:name="android.intent.category.SELECTED_ALTERNATIVE" />
```

```
    </intent-filter>
</activity>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Adding Menu Items Based on an Intent, https://developer.android.com/guide/topics/ui/menus.html#intents

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\AndroidManifest.xml

- In your target XML menu layout file include a group element.

```
    </item>

    <group android:id="@+id/menu_group_informational">
        <item
            android:id="@+id/action_about"
            android:showAsAction="ifRoom"
            android:title="@string/action about"
            app:showAsAction="ifRoom"
            tools:ignore="AppCompatResource" />
        <item
            android:id="@+id/action_help"
            android:showAsAction="ifRoom"
            android:title="@string/action_help"
            app:showAsAction="ifRoom"
            tools:ignore="AppCompatResource" />
    </group>

    <group android:id="@+id/menu_group_dynamic_intents" />
</menu>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\menu\menus_demo.xml

- In onCreateOptionsMenu(): inflate your target menu xml layout; create an implicit intent with Intent.CATEGORY_ALTERNATIVE; call menu.addIntentOptions with the group target.

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

    Intent intent = new Intent();
    intent.setAction(Intent.ACTION_VIEW);
    intent.addCategory(Intent.CATEGORY_ALTERNATIVE);
    if (intent.resolveActivity(getPackageManager()) == null) {
        try {
            throw new Exception(
                    "Intent does not resolve to an Activity when we think " + "it " +
                            "should");
        } catch (Exception e) {
            Log.e(au.com.softmake.standardapplibrary.java.SalGlobals.LOG_TAG,
                  e.getMessage());
        }
    }

    // Search and populate the menu with acceptable offering applications.
    menu.addIntentOptions(R.id.menu_group_dynamic_intents,
                          // Menu group to which new items will be added
                          0,      // Unique item ID (none)
                          0,      // Order for the items (none)
                          this.getComponentName(),   // The current activity name
                          null,   // Specific items to place first (none)
                          intent,
                          // Intent created above that describes our requirements
                          0,      // Additional flags to control items (none)
```

```
                        null);  // Array of MenuItems that correlate to specific items
(none)

    return true;
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Adding Menu Items Based on an Intent, https://developer.android.com/guide/topics/ui/menus.html#intents

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

Options Menu (on the Action Bar)

The items from a menu on the Action bar appear either as "Action buttons" or as items on the "Overflow menu".

Action icons only ever appear when the menu item is presented as an action button, not on the overflow menu.

Options menu coding procedure:

- Override onCreationOptionsMenu() and inflate the XML menu.

```
public class MenusDemo extends Activity {

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


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

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating an Options Menu, https://developer.android.com/guide/topics/ui/menus.html#options-menu

- Use Java code to respond to action buttions in onOptionsItemSelected.

```
public class MenusDemo extends Activity {

...

    @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.
        switch (item.getItemId()) {
            case R.id.action_search:
                Toast.makeText(getApplicationContext(), "Search",
                               Toast.LENGTH_SHORT).show();
                return true;
            case R.id.action_plane_airbus:
                // This proves that a submenu item selected event can be reached.
                Toast.makeText(getApplicationContext(), "Airbus",
                               Toast.LENGTH_SHORT).show();
                return true;
            case R.id.action_settings:
                // Explicit intent
```

```
                startActivity(new Intent(this, SettingsActivity.class));
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Add and handle actions, Respond to Actions
https://developer.android.com/training/appbar/actions#handle-actions

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating an Options Menu, Handling click events,
https://developer.android.com/guide/topics/ui/menus.html#RespondingOptionsMenu

If you have an options menu that is common to many activities, where each of those activities may have additional or slightly different menus, then do the following.

- Create a base activity with the the common menu (and menu items) that implements nothing but optionsCreateOptionsMenu() and onOptionsItemSelected(), (based on an XML menu file).
- For each child activity extend it from the base activity.
- In the child activity optionally override optionsCreateOptionsMenu() and onOptionsItemSelected(), and ensure you call super.onCreateOptionsMenu(menu) and super.onOptionsItemSelected(item) respectively. You can have an additional XML menu file or programmatically add menu items. Menu items from both the base and child activities will be merged.

```
public class MenusDemoExtendedActivity extends MenusDemoActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Whether this comes before or after inflate() doesn't matter.
        super.onCreateOptionsMenu(menu);
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_menus_demo_extended, 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();

        if (id == R.id.action_fonzi) {
            Toast.makeText(getApplicationContext(), "fonzi", Toast.LENGTH_SHORT).show();
            return true;
        }

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

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating an Options Menu, Handling click events,
https://developer.android.com/guide/topics/ui/menus.html#RespondingOptionsMenu

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

An options menu can be declared in an OnCreateOptionsMenu() in either the activity or a fragment. If options menus are declared in both an activity and an fragment then the menus are combinded in the UI. "the system first calls onOptionsItemSelected() for the activity then for each fragment (in the order each fragment was added) until one returns true or all fragments have been called."

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating an Options Menu, https://developer.android.com/guide/topics/ui/menus.html#options-menu

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating an Options Menu, Handling click events, https://developer.android.com/guide/topics/ui/menus.html#RespondingOptionsMenu

Create menu items at runtime:

Define an id in res/values/ids.xml.

- In onCreateOptionsMenu use menu.add().
- Repond to click even on onOptionsItemSelected().

```
// res/values/ids.xms
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item type="id" name="action_cool" />
</resources>

// Activity or Fragment
public class MenusDemo extends Activity {

...

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menus_demo, menu);
        // ID comes from res/values/ids.xml
        menu.add(Menu.NONE /* Group ID */, R.id.action_cool, Menu.NONE /* Order */,
                 "Cool Action");
        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.
        switch (item.getItemId()) {
....
            case R.id.action_cool:
                Toast.makeText(getApplicationContext(), "Cool",
                               Toast.LENGTH_SHORT).show();
                return true;
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating an Options Menu, https://developer.android.com/guide/topics/ui/menus.html#options-menu

Change menu items at runtime, during activity lifecycle (after initial menu creation):

Make menu changes by overriding onPrepareOptionsMenu() in your activity or fragment.

- To invoke a call to onPrepareOptionsMenu() as a response to an event, call invalidateOptionsMenu().
- invalidateOptionsMenu() calls onCreateOptionsMenu() then onPrepareOptionsMenu().

```
public class MenusDemoActivity extends Activity implements CompoundButton
        .OnCheckedChangeListener {

    private ToggleButton mToggleButtonActionHelpAvailable;

...

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menus_demo, menu);
        // ID comes from res/values/ids.xml
        menu.add(Menu.NONE /* Group ID */, R.id.action_cool, Menu.NONE /* Order */,
                 "Cool Action");
        return true;
    }

...

    /**
     * For changes to the menu after initial creation.
     */
    @Override
    public boolean onPrepareOptionsMenu(Menu menu) {
        if (mToggleButtonActionHelpAvailable.isChecked()) {
            // Do nothing. The system restores the original menu structure.
            // as invalidateOptionsMenu() has already called onCreateOptionsMenu();
        } else {
            menu.removeItem(R.id.action help);
        }
        return super.onPrepareOptionsMenu(menu);
    }

    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        String message = "";
        switch (buttonView.getId()) {
            case R.id.toggleButton_action_help_available:
                if (isChecked) {
                    message = "toggleButton_action_help_available on";
                } else {
                    message = "toggleButton_action_help_available of";
                }
                // In effect calls onCreateOptionsMenu() then opPrepareOptionsMenu().
                // See: http://developer.android.com/guide/topics/ui/menus
                // .html#ChangingTheMenu
                invalidateOptionsMenu();
                break;

            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();

    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Changing menu items at runtime, https://developer.android.com/guide/topics/ui/menus#ChangingTheMenu

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

Context Menu

There are two context menus: floating; and contextual action mode. Use a contextual action mode from Android 3.0 (API 11) . The floating context menu is deprecated (but you need to fall back to one for devices less than API 11).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating Contextual Menus, https://developer.android.com/guide/topics/ui/menus#context-menu

Context menus can be initiated from any view although it's common to do so from an item in a list or grid.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating Contextual Menus, Using the contextual action mode https://developer.android.com/guide/topics/ui/menus#CAB

Floating Context Menu

To create a floating context menu.

Create your menu as an XML resource.

```
// res/menu/menu_context_countries.xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:title="@string/action_search" />
    <item
        android:id="@+id/action_flash"
        android:icon="@drawable/ic action flash on"
        android:title="@string/action_flash" />
</menu>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\res\menu\menu_context_countries.xml

In your Activity or Fragment, in onCreate(), pass the relevant View to registerForContextMenu(). The View passed will normally be the ListView or GridView, the items of which you wish to provide a floating context menu for.

```
public class ListActivityDemo extends ListActivity {

...

    @Override
    protected void onCreate(Bundle savedInstanceState) {
...
        ListView listView = this.getListView();
        listView.setAdapter(adapter);
        listView.setOnItemClickListener(mItemClickListener);
        registerForContextMenu(listView);
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating Contextual Menus, Creating a floating context menu, https://developer.android.com/guide/topics/ui/menus#FloatingContextMenu

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\ListActivityDemo.java

In your Activity or Fragment implement onCreateContextMenu() and inflate your XML resource.

```
    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,
                                    ContextMenu.ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);
        getMenuInflater().inflate(R.menu.menu_context_countries, menu);
        menu.setHeaderTitle("Country");
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating Contextual Menus, Creating a floating context menu, https://developer.android.com/guide/topics/ui/menus#FloatingContextMenu

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\ListActivityDemo.java

In your Activity or Fragment implement onContextItemSelected() and handle each menu item selected with respect to each list item.

```
    @Override
    public boolean onContextItemSelected(MenuItem item) {
        AdapterContextMenuInfo menuInfo = (AdapterContextMenuInfo) item.getMenuInfo();
        String message = "";
        switch (item.getItemId()) {
            case R.id.action_search:
                // For the first list item: Search 213129673 0
                message = String.format("%s %d %d", "Search", item.getItemId(),
                                        menuInfo.id);
                Toast.makeText(getApplicationContext(), message,
                               Toast.LENGTH_SHORT).show();
                return true;
            case R.id.action_flash:
                // For the first list item: Flash 213129674 0
                message = String.format("%s %d %d", "Flash", item.getItemId(),
                                        menuInfo.id);
                Toast.makeText(getApplicationContext(), message,
                               Toast.LENGTH_SHORT).show();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating Contextual Menus, Creating a floating context menu, https://developer.android.com/guide/topics/ui/menus#FloatingContextMenu

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\ListActivityDemo.java

Contextual Action Mode (Menu) - For Individual Views.

To create a contextual action mode (menu) - for individual views:

Create the XML menu resource (as above).

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android">
```

```
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:title="@string/action_search" />
    <item
        android:id="@+id/action_flash"
        android:icon="@drawable/ic_action_flash_on"
        android:title="@string/action_flash" />
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:title="@string/action_share"
        android:showAsAction="never"
        tools:ignore="AppCompatResource" />
</menu>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\menu\menu_context.xml

In your activity or fragment: Implement the ActionMode.CallBack interface. Use with a class level ActionMode variable.

```
 private ActionMode mActionMode = null;

private ActionMode.Callback mActionModeCallBack = new ActionMode.Callback() {

    // Called when the action mode is created; startActionMode() was called
    @Override
    public boolean onCreateActionMode(ActionMode mode, Menu menu) {
        getMenuInflater().inflate(R.menu.menu_context, menu);
        return true;
    }
    // Called each time the action mode is shown. Always called after
    // onCreateActionMode, but
    // may be called multiple times if the mode is invalidated.

    @Override
    public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
        return false; // Return false if nothing is done.
    }

    @Override
    public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
        String message = "";
        switch (item.getItemId()) {
            case R.id.action_search:
                message = String.format("%s %d", "Search", item.getItemId());
                break;
            case R.id.action_flash:
                message = String.format("%s %d", "Flash", item.getItemId());
                break;
            case R.id.action_share:
                // item.getTitle() returns "share"
                message = String.format("%s %d %s", "Share", item.getItemId(),
                                        item.getTitle());
                break;
            default:
                return false;
        }
        Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).show();
        return true;
    }

    @Override
    public void onDestroyActionMode(ActionMode mode) {
        mActionMode = null;
    }
};
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating Contextual Menus, Using the contextual action mode
https://developer.android.com/guide/topics/ui/menus#CAB

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

In the fragment or activity implement a LongClickListener and use startActionMode to call the action mode context menu.

```
public class MenusDemoActivity extends Activity implements View.OnLongClickListener {
...
    @Override
    public boolean onLongClick(View v) {
        Toast.makeText(getApplicationContext(), "In Long Click",
                       Toast.LENGTH_SHORT).show();
        switch (v.getId()) {
            case R.id.button_contextual_mode_action_demo:
                // Only set mActionMode if instance not already created.
                if (mActionMode == null) {
                    mActionMode = startActionMode(mActionModeCallBack);
                    mActionMode.setTitle("My Title");
                    v.setSelected(true);
                    return true;
                } else {
                    return false;
                }

            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
                return false;
        }
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating Contextual Menus, Using the contextual action mode
https://developer.android.com/guide/topics/ui/menus#CAB

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

Contextual Action Mode (Menu) - For ListView or GridView

To create a contextual action mode (menu) - for items in a ListView or GridView:

Create the XML menu resource (as above).

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:title="@string/action_search" />
    <item
        android:id="@+id/action_flash"
        android:icon="@drawable/ic_action_flash_on"
        android:title="@string/action_flash" />
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:title="@string/action share"
        android:showAsAction="never"
        tools:ignore="AppCompatResource" />
</menu>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\menu\menu_context.xml

In your Activity or Fragment, Implement the AbsListView.MultiChoiceModeListener (for either ListView or GridView, AbsListView is the parent of either).

```
private AbsListView.MultiChoiceModeListener mMultiChoiceModeListener = new AbsListView
        .MultiChoiceModeListener() {

    @Override
    public void onItemCheckedStateChanged(ActionMode mode, int position, long id,
                                          boolean checked) {
        // Here you can do something when items are selected/de-selected,
        // such as update the title in the CAB
    }

    @Override
    public boolean onCreateActionMode(ActionMode mode, Menu menu) {
        getMenuInflater().inflate(R.menu.menu context, menu);
        return true;

    }

    @Override
    public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
        // Here you can perform updates to the CAB due to
        // an invalidate() request
        return false;
    }

    @Override
    public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
        String message = "";
        switch (item.getItemId()) {
            case R.id.action_search:
                message = String.format("%s %d", "Search", item.getItemId());
                break;
            case R.id.action_flash:
                message = String.format("%s %d", "Flash", item.getItemId());
                break;
            case R.id.action_share:
                // item.getTitle() returns "share"
                message = String.format("%s %d %s", "Share", item.getItemId(),
                                        item.getTitle());
                break;
            default:
                return false;
        }
        Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).show();
        mode.finish(); // Action picked, so close the CAB
        return true;
    }

    @Override
    public void onDestroyActionMode(ActionMode mode) {
        // Here you can make any necessary updates to the activity when
        // the CAB is removed. By default, selected items are deselected/unchecked.
    }
};
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating Contextual Menus, Using the contextual action mode
https://developer.android.com/guide/topics/ui/menus#CAB

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\ListActivityCursorDemo.java

With your listView (or gridView) setMultiChoiceModeListener and setChoiceMode.

```
public class ListActivityCursorDemo extends ListActivity {

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

```
                ListView listView = getListView();
                listView.setAdapter(adapter);

                listView.setMultiChoiceModeListener(mMultiChoiceModeListener);
                listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE_MODAL);

...
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating Contextual Menus, Using the contextual action mode
https://developer.android.com/guide/topics/ui/menus#CAB

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\ListActivityCursorDemo.java

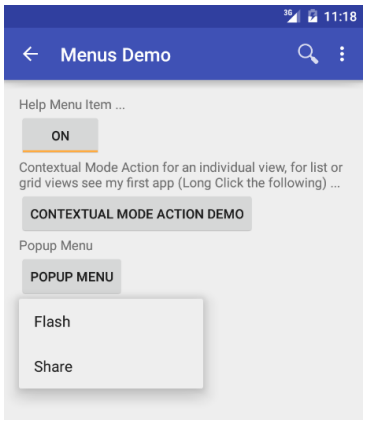
Style your List items (or gird items) to make it obvious when selected (todo).

Optionally provide a UI element, such as checkbox for each item, if long click behaviour not obvious. In this case invoke contextual action mode by setting the list item with setItemChecked().

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating Contextual Menus, Using the contextual action mode
https://developer.android.com/guide/topics/ui/menus#CAB

Popup Menu

A popup menu is a modal menu anchored to a view.



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating a Popup Menu, https://developer.android.com/guide/topics/ui/menus#PopupMenu

Popup menu coding procedure:

Create any old menu in XML

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_flash"
        android:icon="@drawable/ic_action_flash_on"
```

```
        android:title="@string/action_flash" />
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:title="@string/action_share"
        android:showAsAction="never"
        tools:ignore="AppCompatResource" />
</menu>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Menus, Creating a Popup Menu, https://developer.android.com/guide/topics/ui/menus#PopupMenu

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\menu\menu_popup.xml

In your activity create an onclick handler of whatever view will initiate your popop menu. Pass execution to a custom showPopupMenu().

```
public class MenusDemoActivity extends Activity implements View.OnClickListener ...

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

        ...

        Button buttonPopupMenu = (Button) findViewById(R.id.button_popup_menu);
        buttonPopupMenu.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button popup menu:
                showPopupMenu(v);
                break;

            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

Code to show the popup menu in response to a user action on a view.

```
    private void showPopupMenu(View v) {
        PopupMenu popupMenu = new PopupMenu(this, v);

        MenuInflater inflater = popupMenu.getMenuInflater();
        inflater.inflate(R.menu.menu_popup, popupMenu.getMenu());
        popupMenu.show();
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

Handle popup menu item clicks.

```
public class MenusDemoActivity extends Activity implements View.OnClickListener,
                                                    PopupMenu.OnMenuItemClickListener {
```

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

        Button buttonPopupMenu = (Button) findViewById(R.id.button_popup_menu);
        buttonPopupMenu.setOnClickListener(this);
    }

    private void showPopupMenu(View v) {
        PopupMenu popupMenu = new PopupMenu(this, v);
        popupMenu.setOnMenuItemClickListener(this);
        MenuInflater inflater = popupMenu.getMenuInflater();
        inflater.inflate(R.menu.menu_popup, popupMenu.getMenu());
        popupMenu.show();
    }

    /**
     * Popup Menu Click Listener
     */
    @Override
    public boolean onMenuItemClick(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_flash:
                Toast.makeText(getApplicationContext(), "Flash",
                               Toast.LENGTH_SHORT).show();
                return true;

            case R.id.action_share:
                Toast.makeText(getApplicationContext(), "Share",
                               Toast.LENGTH_SHORT).show();
                return true;

            default:
                return false;
        }
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MenusDemoActivity.java

Navigation Drawer

Navigations Drawers can be:

- Permanently onscreen (suitable for tablets) ... a "standard drawer"; or
- Temporarily onscreen (suitable for mobiles) ... a "modal drawer"; if so they can be opened:
    - Using an top app bar menu icon; or
    - Swiping in from the left (or right if using a right-to-left language); or
    - From the bottom when opened by a bottom app bar's menu icon.

(Google 2014, "Android Design Guide", https://developer.android.com/design), Navigation Drawer, Usage, Types, https://material.io/components/navigation-drawer#usage

Create a navigation drawer:

1. Add dependencies to your app's build.gradle ...

```
dependencies {
  implementation 'com.android.support:drawerlayout:28.0.0'

  // or ?
  implementation 'com.android.support:appcompat-v7:28.0.0'
  implementation 'com.android.support:design:28.0.0'
```

```
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Update UI components with NavigationUI, Add a navigation drawer, https://developer.android.com/guide/navigation/navigation-ui#add_a_navigation_drawer

(Google n.d., "Android API Reference > Andoid Support Library", Accessed 2021-02-19. https://developer.android.com/reference/android/support/classes), https://developer.android.com/reference/android/support/v4/widget/DrawerLayout

2. To your Activity's layout file surround your main content in a `DrawLayout` element (as the new root). At the same level as your main content add a `NavigationView` element.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Use DrawerLayout as root container for activity -->
<android.support.v4.widget.DrawerLayout
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:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context=".NavigationDrawerDemoActivity">

    <!-- Layout to contain contents of main body of screen (drawer will slide over
this) -->
    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello World" />
    </android.support.constraint.ConstraintLayout>

    <!-- Container for contents of drawer - use NavigationView to make configuration
easier -->
    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true" />

</android.support.v4.widget.DrawerLayout>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Update UI components with NavigationUI, Add a navigation drawer, https://developer.android.com/guide/navigation/navigation-ui#add_a_navigation_drawer

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_navigation_drawer_demo.xml

3. Create a menu resource file:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_camera"
            android:icon="@drawable/ic_menu_camera"
            android:title="import" />
        <item
            android:id="@+id/nav_gallery"
```

```
            android:icon="@drawable/ic_menu_gallery"
            android:title="gallery" />
        <item
            android:id="@+id/nav_slideshow"
            android:icon="@drawable/ic_menu_slideshow"
            android:title="slideshow" />
        <item
            android:id="@+id/nav_manage"
            android:icon="@drawable/ic_menu_manage"
            android:title="tools" />
    </group>
</menu>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Update UI components with NavigationUI, Add a navigation drawer, https://developer.android.com/guide/navigation/navigation-ui#add_a_navigation_drawer

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\menu\navigation_drawer.xml

4. Reference the menu resource file from the Navigation Drawer element.

```
    <android.support.design.widget.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:menu="@menu/navigation_drawer" />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Update UI components with NavigationUI, Add a navigation drawer, https://developer.android.com/guide/navigation/navigation-ui#add_a_navigation_drawer

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_navigation_drawer_demo.xml

5. Add a toolbar to your layout

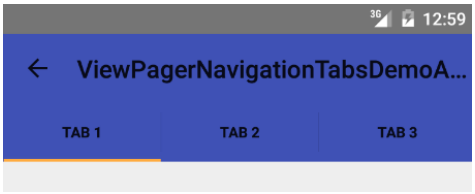
Tabs and ViewPager (Sliding View)

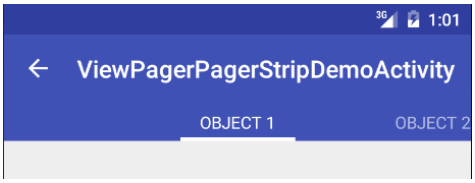
Overview

To achieve Tabs, often with a ViewPager (Sliding View), you have the following options:

Navigation Tabs - Basic (deprecated). No Viewpager. Uses ActionBar.

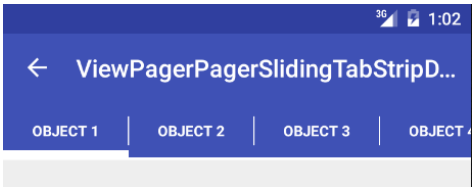
ViewPager with Navigation Tabs (deprecated). Uses ActionBar.





ViewPager with a PagerTabStrip, or PagerTitleStrip. Toolbar compatible.

(Recommended) ViewPager with PagerSlidingTabStrip. Third party library. Toolbar compatible. https://github.com/astuetz/PagerSlidingTabStrip



Coding Procedures

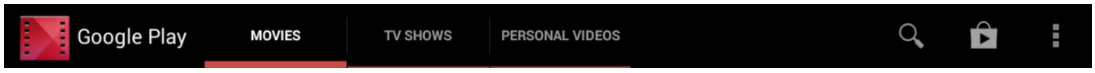
Navigation Tabs (with ActionBar). Deprecated.

Navigation views allow your users to switch between views.

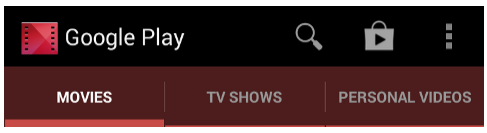
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

Navigation views will be presented differently depending on screen size:

For wide screens as tabs next to action items:



For narrow screens as a row below the action bar



For narrow screens, in some circumstances, as a drop down list.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

Compatibility

Basic Navigation Tabs work only with an ActionBar, not a Toolbar. Basic Navigation Tabs (ActionBar.Tab) is deprecated. To use a Toolbar with Navigation tabs use the ViewPager Technique.

http://developer.android.com/reference/android/support/v7/app/ActionBar.html
http://stackoverflow.com/questions/26540078/use-tab-with-new-toolbar-appcompat-v7-21
John Bentley: experimentally found a Toolbar doesn't work with Basic Navigation Tabs.

For Basic Navigation Tabs, to support down to API 11 must use ActionBarActivty; support libraries; and getSupportActionBar().

```
// To support back to API 11 (perhaps earlier)
// Include android.app.Activity as we use
// public TabListener(Activity activity, String tag, Class<T> pClass) {
import android.app.Activity;
import android.support.v7.app.ActionBar;
import android.support.v7.app.ActionBarActivity;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentTransaction;
....

public class NavigationTabsBasicDemoActivity extends ActionBarActivity {
```

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

        ActionBar actionBar = getSupportActionBar();
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\NavigationTabsBasicDemoActivity.java

For Basic Navigation Tabs, to support down to API 13 Basic Navigation Tabs must use Activty; native libraries; and getActionBar().

```
// From API 13
import android.app.ActionBar;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentTransaction;
...

public class NavigationTabsBasicDemoActivity extends Activity {

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

        ActionBar actionBar = getActionBar();
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\NavigationTabsBasicDemoActivity.java (With modfications to test this).

Navigation Tabs Basic (with ActionBar) - Depcrecated

Navigation Tabs Basic is used with an ActionBar. For backward compatibility down to API 11 we use the ActionBar, and associated objects, in the AppCompat Library. You can't use Basic Navigation Tabs with a Toolbar.

Navigation Tabs Basic coding procedure:

Use an AppCompat theme.

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

....

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

// values/styles.xml
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">#3F51B5</item>

        <!--    darker variant of colorPrimary (for status bar, contextual app bars) -->
        <item name="colorPrimaryDark">#303F9F</item>

        <!--   theme UI controls like checkboxes and text fields -->
        <item name="colorAccent">#ffab40</item>
    </style>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\values\styles.xml
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

Create an ActionBarActivity() with imports from the support library.

```
package au.com.softmake.mysecondapp;

// Include android.app.Activity as we use
// public TabListener(Activity activity, String tag, Class<T> pClass) {
import android.app.Activity;
import android.support.v7.app.ActionBar;
import android.support.v7.app.ActionBarActivity;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentTransaction;
import android.os.Bundle;
import android.view.*;
import android.widget.TextView;

public class NavigationTabsBasicDemoActivity extends ActionBarActivity {
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\NavigationTabsBasicDemoActivity.java
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

Implement a tab listener (perhaps this could live as a class on its own).

```
public class NavigationTabsBasicDemoActivity extends ActionBarActivity {

    static public class TabListener<T extends Fragment> implements ActionBar.TabListener {

        private Fragment mFragment;
        private final Activity mActivity;
        private final String mTag;
        private final Class<T> mClass;

        /**
         * Constructor used each time a new tab is created.
         *
         * @param activity The host Activity, used to instantiate the
         *                 fragment
         * @param tag      The identifier tag for the fragment
         * @param pClass   The fragment's Class, used to instantiate the
         *                 fragment
         * @see <a
         * href="http://developer.android.com/guide/topics/ui/actionbar
         * .html#Tabs">
         * Developers Guide > Action Bar > Adding Navigation Tabs</a>
         */
        public TabListener(Activity activity, String tag, Class<T> pClass) {
            mActivity = activity;
            mTag = tag;
            mClass = pClass;
        }

        @Override
        public void onTabSelected(ActionBar.Tab tab, FragmentTransaction ft) {
            if (mFragment == null) {
                mFragment = Fragment.instantiate(mActivity, mClass.getName());
                ft.add(android.R.id.content, mFragment, mTag);
            } else {
                // If it exists, attach it in order to show it
                ft.attach(mFragment);
            }
        }

        @Override
        public void onTabUnselected(ActionBar.Tab tab, FragmentTransaction ft) {
            if (mFragment != null) {
                // Detach the fragment, because another one is about to be
```

```
                // attached.
                ft.detach(mFragment);
            }
        }

        @Override
        public void onTabReselected(ActionBar.Tab tab, FragmentTransaction ft) {
            // Do nothing.
        }
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\NavigationTabsBasicDemoActivity.java
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

Decide if you want an XML layout for the activity. Generally you don't. In which case you don't need to use any setContentView() in your Activity's onCreate(). Instead, in your Tab Listener, you reference the default root view with android.R.id.content ID.

```
public class NavigationTabsBasicDemoActivity extends ActionBarActivity {

    static public class TabListener<T extends Fragment> implements ActionBar.TabListener {
...

        @Override
        public void onTabSelected(ActionBar.Tab tab, FragmentTransaction ft) {
            if (mFragment == null) {
                mFragment = Fragment.instantiate(mActivity, mClass.getName());
                ft.add(android.R.id.content, mFragment, mTag);
            } else {
...

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // No need for setContentView() to be used, Instead  we use the root
        // android.R.id.content as the container for each fragment,
        // which is set in the TabListener
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\NavigationTabsBasicDemoActivity.java
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

In your activity's onCreate() get an ActionBar and set the Navigation Mode to NAVIGATION_MODE_TABS

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // No need for setContentView() to be used, Instead  we use the root
        // android.R.id.content as the container for each fragment,
        // which is set in the TabListener

        ActionBar actionBar = getSupportActionBar();
        actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
        actionBar.setDisplayShowTitleEnabled(true);
        actionBar.setDisplayHomeAsUpEnabled(true);
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\NavigationTabsBasicDemoActivity.java
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

Implement fragments for your tabs.

```
    /**
     * In this example use one Fragment but display different data based
     * on which tab is shown. In production you'd probably use a separate
     * fragment for each tab.
     */
    public static class PlaceholderFragment extends Fragment {

        public PlaceholderFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            View rootView = inflater.inflate(R.layout.fragment_navigation_tabs_basic_demo,
                                             container,
                                             false);
            TextView outputTextView = (TextView)
                    rootView.findViewById(R.id.output_textView);
            outputTextView.setText("Hello " + getTag());
            return rootView;
        }
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\NavigationTabsBasicDemoActivity.java
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

In your Activity's onCreate() Add individual tabs to the ActionbBar.tab, setting the previously implemented Tab Listener and individual tab fragments.

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        ActionBar actionBar = getSupportActionBar();

        ActionBar.Tab tab = actionBar.newTab()
                                .setText("Artist")
                                .setTabListener(new TabListener<PlaceholderFragment>(
                                    this,
                                    "artist",
                                    PlaceholderFragment.class));
        actionBar.addTab(tab);

        tab = actionBar.newTab()
                                .setText("Album")
                                .setTabListener(new TabListener<PlaceholderFragment>(
                                    this,
                                    "album",
                                    PlaceholderFragment.class));
        actionBar.addTab(tab);
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\NavigationTabsBasicDemoActivity.java
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User Interface, Add the app bar, Overview, https://developer.android.com/training/appbar

ViewPager with Navigation Tabs (With ActionBar). Deprecated; use ViewPager2

Coding Procedure:

Add the v4 support library to your gradle file.

```
dependencies {
    compile 'com.android.support:support-v4:22.0.0'
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, https://developer.android.com/guide/navigation/navigation-swipe-view#horizontal-paging

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\build.gradle

Use a ViewPager (SwipeView) in your XML Layout. You only need the ViewPager and nothing else.

```
<!-- \res\layout\activity_navigation_tabs_view_pager_demo.xml -->
<android.support.v4.view.ViewPager
    android:id="@+id/viewPager"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, Implement Swipe View, https://developer.android.com/guide/navigation/navigation-swipe-view#implement_swipe_views

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\activity_navigation_tabs_view_pager_demo.xml

Create a PageDisplayFragment (here "DemoFragment") to display data on each page (for Pages that are not displaying collection data you'll probably need to create a number of different fragments).

```
...
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentStatePagerAdapter;
import android.support.v4.view.PagerTabStrip;
import android.support.v4.view.ViewPager;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.widget.Toolbar;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
...

public class ViewPagerNavigationTabsDemoActivity extends ActionBarActivity {

    DemoCollectionPagerAdapter mDemoCollectionPagerAdapter;

    public static class DemoFragment extends Fragment {
        public static final String ARG_OBJECT = "object";

        public DemoFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            // The last two arguments ensure LayoutParams are inflated
            // properly.
            View rootView = inflater.inflate(R.layout.fragment_collection_object,
                                             container,
                                             false);

            Bundle args = getArguments();
            TextView outputTextView = (TextView)
rootView.findViewById(R.id.textView_output);
            CharSequence output = Integer.toString(args.getInt(ARG_OBJECT));
            outputTextView.setText(output);

            return rootView;
```

```
        }
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, https://developer.android.com/guide/navigation/navigation-swipe-view#horizontal-paging

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerNavigationTabsDemoActivity.java

Extend a PagerAdapter. the FragmentPagerAdapter is for a small number of pages; the FragmentStatePagerAdapter is for a collection of objects for which the number of pages is undetermined (this destroys fragments as the user pages, for efficiency). This PagerAdapter returns a PageDisplayFrament (from above).

```
public class ViewPagerNavigationTabsDemoActivity extends ActionBarActivity {

    DemoCollectionPagerAdapter mDemoCollectionPagerAdapter;

    // Use FragmentStatePageAdapter for a collection of objects (destroys fragments as
    // user navigates to other pages, minimizing memory use), rather than a
    // FragmentPagerAdapter.
    public class DemoCollectionPagerAdapter extends FragmentStatePagerAdapter {

        public DemoCollectionPagerAdapter(FragmentManager fm) {
            super(fm);
        }

        @Override
        public Fragment getItem(int position) {
            Fragment fragment = new DemoFragment();
            Bundle args = new Bundle();
            // Our object will just contain (and display) an integer
            args.putInt(DemoFragment.ARG_OBJECT, position + 1);
            fragment.setArguments(args);
            return fragment;
        }

        @Override
        public int getCount() {
            return 100;
        }

        @Override
        public CharSequence getPageTitle(int position) {
            return "OBJECT " + (position + 1);
        }
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, Implement Swipe View, https://developer.android.com/guide/navigation/navigation-swipe-view#implement_swipe_views

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerNavigationTabsDemoActivity.java

Declare a class level ViewPager. Implement a tab listener, setting the ViewPager on onTabSelected.

```
public class ViewPagerNavigationTabsDemoActivity extends ActionBarActivity {

...
    public ViewPager mViewPager;
...

    public class TabListener<T extends Fragment> implements ActionBar.TabListener {

        @Override
        public void onTabSelected(ActionBar.Tab tab,
```

```
                                    FragmentTransaction fragmentTransaction) {
            // When the tab is selected, switch to the corresponding page in the
            // ViewPager.
            mViewPager.setCurrentItem(tab.getPosition());
        }

        @Override
        public void onTabUnselected(ActionBar.Tab tab,
                                    FragmentTransaction fragmentTransaction) {

        }

        @Override
        public void onTabReselected(ActionBar.Tab tab,
                                    FragmentTransaction fragmentTransaction) {

        }
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, Implement Swipe View, https://developer.android.com/guide/navigation/navigation-swipe-view#implement_swipe_views

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerNavigationTabsDemoActivity.java

In your onCreate():

- Hook the ViewPager up to the PagerAdapter;
- Setup Navigation Tabs;
- Add some tabs;
- Change the tabs as you swipe the page;

```
public class ViewPagerNavigationTabsDemoActivity extends ActionBarActivity {
....

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity navigation tabs view pager demo);

        // Hook the ViewPager up to the PagerAdpater
        mDemoCollectionPagerAdapter = new DemoCollectionPagerAdapter(
                getSupportFragmentManager());
        mViewPager = (ViewPager) findViewById(R.id.viewPager);
        mViewPager.setAdapter(mDemoCollectionPagerAdapter);

        // Setup Navigation Tabs
        ActionBar actionBar = getSupportActionBar();
        actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
        ActionBar.TabListener tabListener = new TabListener<>();

        // Add some tabs
        for (int i = 1; i <= 3; i++) {
            actionBar.addTab(actionBar.newTab().setText("Tab " + i).setTabListener(
                    tabListener));
        }

        // Change the Tabs as you swipe the page.
        mViewPager.setOnPageChangeListener(new ViewPager.SimpleOnPageChangeListener() {
            @Override
            public void onPageSelected(int position) {
                getSupportActionBar().setSelectedNavigationItem(position);
            }
        });
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, Implement Swipe View, https://developer.android.com/guide/navigation/navigation-swipe-view#implement_swipe_views

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerNavigationTabsDemoActivity.java

ViewPager with Pager Strip (With Toolbar)

You can use a ViewPager with a Pager Strip, either a PagerTitleStrip or a PagerTabStrip. Using it with a PagerTabStrip is recommended as you can respond to touch events on the tab.

Coding Procedure:

Add the v4 support library to your gradle file.

```
dependencies {
    compile 'com.android.support:support-v4:22.0.0'
}
```

Create a layout file with a Toolbar, ViewPager and a nested Pager Strip.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:orientation="vertical"
    tools:context="au.com.softmake.mysecondapp.ActionBarDemo"
    >

    <android.support.v7.widget.Toolbar
        android:id="@+id/my_toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        app:popupTheme="@style/ToolbarPopupCustomTheme"
        app:theme="@style/ToolbarCustomTheme"
        />

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

        <android.support.v4.view.ViewPager
            android:id="@+id/viewPager"
            xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            >

            <android.support.v4.view.PagerTabStrip
                android:id="@+id/pager_strip"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="top"
                android:paddingBottom="4dp"
                android:paddingTop="4dp"
                android:theme="@style/PagerStripTheme"
                />
                <!--android:background="#33b5e5"-->
                <!--android:textColor="#fff"-->

        </android.support.v4.view.ViewPager>
    </LinearLayout>

</LinearLayout>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\activity_view_pager_pager_strip_demo.xml

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, Add Tabs using a TabLayout, https://developer.android.com/guide/navigation/navigation-swipe-view#add_tabs_using_a_tablayout

See Toolbar Setup to do things like use an Appcompat theme, to support the ToolBar.

Create a PageDisplayFragment (here "DemoFragment") to display data on each page (for Pages that are not displaying collection data you'll probably need to create a number of different fragments).

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentStatePagerAdapter;
import android.support.v4.view.PagerTabStrip;
import android.support.v4.view.ViewPager;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.widget.Toolbar;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import au.com.softmake.mysecondapp.R;

public class ViewPagerPagerStripDemoActivity extends ActionBarActivity {

    ...

    public static class DemoFragment extends Fragment {
        public static final String ARG_OBJECT = "object";

        public DemoFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            // The last two arguments ensure LayoutParams are inflated
            // properly.
            View rootView = inflater.inflate(R.layout.fragment_collection_object,
                                             container,
                                             false);

            Bundle args = getArguments();
            TextView outputTextView = (TextView)
rootView.findViewById(R.id.textView_output);
            CharSequence output = Integer.toString(args.getInt(ARG_OBJECT));
            outputTextView.setText(output);

            return rootView;
        }
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, Add Tabs using a TabLayout, https://developer.android.com/guide/navigation/navigation-swipe-view#add_tabs_using_a_tablayout

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerPagerStripDemoActivity.java

Extend a PagerAdapter. the FragmentPagerAdapter is for a small number of pages; the FragmentStatePagerAdapter is for a collection of objects for which the number of pages is undetermined (this destroys fragments as the user pages, for efficiency). This PagerAdapter returns a PageDisplayFrament (from above).

```
public class ViewPagerPagerStripDemoActivity extends ActionBarActivity {

    DemoCollectionPagerAdapter mDemoCollectionPagerAdapter;
   ...

    // Use FragmentStatePageAdapter for a collection of objects (destroys fragments as
    // user navigates to other pages, minimizing memory use), rather than a
    // FragmentPagerAdapter.
    public class DemoCollectionPagerAdapter extends FragmentStatePagerAdapter {

        public DemoCollectionPagerAdapter(FragmentManager fm) {
            super(fm);
        }

        @Override
        public Fragment getItem(int position) {
            Fragment fragment = new DemoFragment();
            Bundle args = new Bundle();
            // Our object will just contain (and display) an integer
            args.putInt(DemoFragment.ARG_OBJECT, position + 1);
            fragment.setArguments(args);
            return fragment;
        }

        @Override
        public int getCount() {
            return 100;
        }

        @Override
        public CharSequence getPageTitle(int position) {
            return "OBJECT " + (position + 1);
        }
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, Add Tabs using a TabLayout, https://developer.android.com/guide/navigation/navigation-swipe-view#add_tabs_using_a_tablayout

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerNavigationTabsDemoActivity.java

Declare a class level ViewPager. No need for a tab listener, as we are not using Navigation tabs.

```
public class ViewPagerPagerStripDemoActivity extends ActionBarActivity {

    DemoCollectionPagerAdapter mDemoCollectionPagerAdapter;
    public ViewPager mViewPager;
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, https://developer.android.com/guide/navigation/navigation-swipe-view#horizontal-paging
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerNavigationTabsDemoActivity.java

In your main onCreate():

- Setup Toolbar;
- Hook the ViewPager up to the PagerAdapter.

```
public class ViewPagerPagerStripDemoActivity extends ActionBarActivity {

    DemoCollectionPagerAdapter mDemoCollectionPagerAdapter;
    public ViewPager mViewPager;

   ...

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_pager_pager_strip_demo);

        // Setup Toolbar
        Toolbar toolbar = (Toolbar) findViewById(R.id.my_toolbar);
        setSupportActionBar(toolbar);

        // Must come after setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        // Hook the ViewPager up to the PagerAdapter;
        mDemoCollectionPagerAdapter = new DemoCollectionPagerAdapter(
                getSupportFragmentManager());
        mViewPager = (ViewPager) findViewById(R.id.viewPager);
        mViewPager.setAdapter(mDemoCollectionPagerAdapter);

        // Moving the tabs when you swipe a page, comes for free.
        // Pressing a tab and moving to the page, comes for free.
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, https://developer.android.com/guide/navigation/navigation-swipe-view#horizontal-paging

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerNavigationTabsDemoActivity.java

ViewPager with PagerSlidingTabStrip (With Toolbar). Recommended.

The PagerSlidingTabStrip is a third party UI component that gives you the following benefits:

- It works with a ViewPager;
- It works with a toolbar;
- Unlike Pager Strips (PagerTabStrip or PagerTitleStrip) it provides compact tabs (if you wish); and these tabs also slide (if there are too many of them to fit on one screen).

*(Stuetz 2013, *Android PagerSlidingTabStrip*, https://github.com/astuetz/PagerSlidingTabStrip), https://github.com/astuetz/PagerSlidingTabStrip*

Coding Procedure:

Add the v4 support library and pagerslidingtabstrip to your gradle file.

```
dependencies {
    compile 'com.android.support:support-v4:22.0.0'
    compile 'com.astuetz:pagerslidingtabstrip:1.0.1'
}
```

*(Stuetz 2013, *Android PagerSlidingTabStrip*, https://github.com/astuetz/PagerSlidingTabStrip), https://github.com/astuetz/PagerSlidingTabStrip*

Create a layout file with a Toolbar, PagerSlidingTabStrip and ViewPager.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:orientation="vertical"
    tools:context="au.com.softmake.mysecondapp.ActionBarDemo"
    >
```

```
    <android.support.v7.widget.Toolbar
        android:id="@+id/my_toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        app:popupTheme="@style/ToolbarPopupCustomTheme"
        app:theme="@style/ToolbarCustomTheme"
        />

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

        <com.astuetz.PagerSlidingTabStrip
            android:id="@+id/pagerSlidingTabStrip"
            style="@style/PagerSlidingTabStripTheme"
            android:layout_width="match_parent"
            android:layout_height="48dip"
            app:pstsDividerColor="@color/action_bar_text_color"
            app:pstsIndicatorColor="@color/action_bar_text_color"
            app:pstsUnderlineColor="@color/action_bar_text_color"
            />

        <android.support.v4.view.ViewPager
            android:id="@+id/viewPager"
            xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            />

    </LinearLayout>
</LinearLayout>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\activity_view_pager_pager_sliding_tab_strip_demo.xml
*(Stuetz 2013, *Android PagerSlidingTabStrip*, https://github.com/astuetz/PagerSlidingTabStrip),*
https://github.com/astuetz/PagerSlidingTabStrip

See Toolbar Setup to do things like use an Appcompat theme, to support the ToolBar.

Create a PageDisplayFragment (here "DemoFragment") to display data on each page (for Pages that are not displaying collection data you'll probably need to create a number of different fragments) in an ActionBarActivity.

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentStatePagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.widget.Toolbar;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import au.com.softmake.mysecondapp.R;
import com.astuetz.PagerSlidingTabStrip;

public class ViewPagerPagerSlidingTabStripDemoActivity extends ActionBarActivity {

...

    public static class DemoFragment extends Fragment {
        public static final String ARG_OBJECT = "object";

        public DemoFragment() {
        }
```

```
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            // The last two arguments ensure LayoutParams are inflated
            // properly.
            View rootView = inflater.inflate(R.layout.fragment_collection_object,
                                             container,
                                             false);

            Bundle args = getArguments();
            TextView outputTextView = (TextView)
rootView.findViewById(R.id.textView_output);
            CharSequence output = Integer.toString(args.getInt(ARG_OBJECT));
            outputTextView.setText(output);

            return rootView;
        }
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerPagerSlidingTabStripDemoActivity.java

*(Stuetz 2013, *Android PagerSlidingTabStrip*, https://github.com/astuetz/PagerSlidingTabStrip), https://github.com/astuetz/PagerSlidingTabStrip*

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, https://developer.android.com/guide/navigation/navigation-swipe-view#horizontal-paging

Extend a PagerAdapter. the FragmentPagerAdapter is for a small number of pages; the FragmentStatePagerAdapter is for a collection of objects for which the number of pages is undetermined (this destroys fragments as the user pages, for efficiency). This PagerAdapter returns a PageDisplayFrament (from above).

```
public class ViewPagerPagerSlidingTabStripDemoActivity extends ActionBarActivity {

    DemoCollectionPagerAdapter mDemoCollectionPagerAdapter;

    // Use FragmentStatePageAdapter for a collection of objects (destroys fragments as
    // user navigates to other pages, minimizing memory use), rather than a
    // FragmentPagerAdapter.
    public class DemoCollectionPagerAdapter extends FragmentStatePagerAdapter {

        public DemoCollectionPagerAdapter(FragmentManager fm) {
            super(fm);
        }

        @Override
        public Fragment getItem(int position) {
            Fragment fragment = new DemoFragment();
            Bundle args = new Bundle();
            // Our object will just contain (and display) an integer
            args.putInt(DemoFragment.ARG_OBJECT, position + 1);
            fragment.setArguments(args);
            return fragment;
        }

        @Override
        public int getCount() {
            return 100;
        }

        @Override
        public CharSequence getPageTitle(int position) {
            return "OBJECT " + (position + 1);
        }
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerPagerSlidingTabStripDemoActivity.java

*(Stuetz 2013, *Android PagerSlidingTabStrip*, https://github.com/astuetz/PagerSlidingTabStrip), https://github.com/astuetz/PagerSlidingTabStrip*

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, https://developer.android.com/guide/navigation/navigation-swipe-view#horizontal-paging

Declare a class level ViewPager. No need for a tab listener, as we are not using Navigation tabs.

```
public class ViewPagerPagerSlidingTabStripDemoActivity extends ActionBarActivity {

    DemoCollectionPagerAdapter mDemoCollectionPagerAdapter;
    public ViewPager mViewPager;
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerPagerSlidingTabStripDemoActivity.java

*(Stuetz 2013, *Android PagerSlidingTabStrip*, https://github.com/astuetz/PagerSlidingTabStrip), https://github.com/astuetz/PagerSlidingTabStrip*

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, https://developer.android.com/guide/navigation/navigation-swipe-view#horizontal-paging

In your main onCreate():

Setup Toolbar;

Hook the ViewPager up to the PagerAdapter.

Hook the PagerSligingTabStrip up to the ViewPager.

```
public class ViewPagerPagerSlidingTabStripDemoActivity extends ActionBarActivity {

    DemoCollectionPagerAdapter mDemoCollectionPagerAdapter;
    public ViewPager mViewPager;


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

        // Setup Toolbar
        Toolbar toolbar = (Toolbar) findViewById(R.id.my_toolbar);
        setSupportActionBar(toolbar);

        // Must come after setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        // Hook the ViewPager up to the PagerAdapter
        mDemoCollectionPagerAdapter = new DemoCollectionPagerAdapter(
                getSupportFragmentManager());
        mViewPager = (ViewPager) findViewById(R.id.viewPager);
        mViewPager.setAdapter(mDemoCollectionPagerAdapter);

        // Hook the PagerSlidingTabStrip up to the ViewPager.
        PagerSlidingTabStrip pagerSlidingTabStrip = (PagerSlidingTabStrip) findViewById
                (R.id.pagerSlidingTabStrip);
        pagerSlidingTabStrip.setViewPager(mViewPager);
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerPagerSlidingTabStripDemoActivity.java

*(Stuetz 2013, *Android PagerSlidingTabStrip*, https://github.com/astuetz/PagerSlidingTabStrip), https://github.com/astuetz/PagerSlidingTabStrip*

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, https://developer.android.com/guide/navigation/navigation-swipe-view#horizontal-paging

To properly style the PagerSligingTabStrip:

- Use a style;
- Use custom styling attributes in the XML layout file.
- Custom color settings.

```
<!-- layout/activity_view_pager_pager_sliding_tab_strip_demo.xml -->
<LinearLayout 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:orientation="vertical"
    tools:context="au.com.softmake.mysecondapp.ActionBarDemo"
    >

    ...

    <com.astuetz.PagerSlidingTabStrip
        android:id="@+id/pagerSlidingTabStrip"
        style="@style/PagerSlidingTabStripTheme"
        android:layout_width="match_parent"
        android:layout_height="48dip"
        app:pstsDividerColor="@color/action_bar_text_color"
        app:pstsIndicatorColor="@color/action_bar_text_color"
        app:pstsUnderlineColor="@color/action_bar_text_color"
        />

<!-- values/styles.xml -->
<resources xmlns:app="http://schemas.android.com/apk/res-auto">

    ...

    <style name="PagerSlidingTabStripTheme">
        <item name="android:background">@color/colorPrimary</item>
        <item name="android:textColor">@color/action_bar_text_color</item>
    </style>

<!-- values/colors.xml -->
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="action_bar_text_color">#fff</color>
</resources>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\activity_view_pager_pager_sliding_tab_strip_demo.xml

*(Stuetz 2013, *Android PagerSlidingTabStrip*, https://github.com/astuetz/PagerSlidingTabStrip), https://github.com/astuetz/PagerSlidingTabStrip*

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, https://developer.android.com/guide/navigation/navigation-swipe-view#horizontal-paging

With FragmentPagerAdapter (for a small number of pages)

There are two types of PagerAdapter:

- FragmentPagerAdapter is for a small number of pages;
- FragmentStatePagerAdapter (as above) is for a collection of objects for which the number of pages is undetermined (this destroys fragments as the user pages, for efficiency).

Two change from a FragmentStatePagerAdapter (see examples above) to a FragmentPagerAdapter all you need to do is:

Extend from FragmentPagerAdapter.

```
// Do this
public class DemoFragmentPagerAdapter extends FragmentPagerAdapter {

// Instead of this
public class DemoCollectionPagerAdapter extends FragmentStatePagerAdapter {
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, https://developer.android.com/guide/navigation/navigation-swipe-view#horizontal-paging

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerFragmentPagerAdapterDemoActivity.java

Reduce the count returned by getCount(). A FragmentPagerAdapter is for handling a lower number of pages. Leave the other methods alone.

```
    public class DemoFragmentPagerAdapter extends FragmentPagerAdapter {

....

        @Override
        public int getCount() {
            return 3;
        }
```

Refactor your variable names and set your ViewPager to the new adapter.

```
public class ViewPagerFragmentPagerAdapterDemoActivity extends ActionBarActivity {

    DemoCollectionPagerAdapter mDemoCollectionPagerAdapter; // Old.
    DemoFragmentPagerAdapter mDemoFragmentPagerAdapter;  // New.

    @Override
    protected void onCreate(Bundle savedInstanceState) {
...

        mDemoFragmentPagerAdapter = new DemoFragmentPagerAdapter
                (getSupportFragmentManager());
        mViewPager.setAdapter(mDemoFragmentPagerAdapter);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Navigations component, Create swipe views with tabs using ViewPager, https://developer.android.com/guide/navigation/navigation-swipe-view#horizontal-paging

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\tabviews\ViewPagerFragmentPagerAdapterDemoActivity.java

Settings (Preferences)

Basics

The UI for settings is implemented with the AndroidX Preference Library (Android's `Preference` API is deprecated).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, https://developer.android.com/guide/topics/ui/settings

A single setting is implemented as a subclass of a `Preference` object. Common subclasses include `CheckBoxPreference`, `ListPreference`, and `EditTextPreference`. Settings are nested under a `PreferenceScreen` object. All this is done in an xml file located at `/res/xml/` not `res/layout/`. The conventional file name in `/res/xml/` is `preferences.xml`.

```
<!-- res/xml/preferences.xml  -->
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android"
                  android:layout_width="match_parent"
                  android:layout height="match parent">

    <CheckBoxPreference
        android:key="preference_is_happy"
        android:title="Happy"
        android:summary="Currently happy"
        android:defaultValue="true" />

    <!-- We are using the same array for entries and values -->
    <ListPreference
        android:dependency="preference_is_happy"
        android:key="preference happy scale"
        android:title="Happy Scale"
        android:summary="@string/overwritten"
        android:defaultValue="Very"
        android:dialogTitle="Happy Scale"
        android:entries="@array/array_happy_scale"
        android:entryValues="@array/array_happy_scale" />

    <EditTextPreference
        android:key="preference_cool_word"
        android:title="Cool Word"
        android:summary="@string/overwritten"
        android:defaultValue="Nice"
        android:dialogTitle="Cool Word" />

</PreferenceScreen>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, https://developer.android.com/guide/topics/ui/settings

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\res\xml\preferences.xml

Initialising and Invoking Preferences

If the app is targeting Android 3.0 (API 11 HONEYCOMB) or greater you can invoke the preferences through an activity that calls a PreferenceFragement as follows.

```
public class SettingsActivity extends Activity {

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

```
        // We don't need a user created activity_settings.xml layout file,
        // we use android's content resource
        if (savedInstanceState == null) {
            getFragmentManager().beginTransaction().replace(android.R.id.content,
                    new SettingsFragment()).commit();
        }
    }

    /**
     * A placeholder fragment containing a simple view.
     */
    public static class SettingsFragment extends PreferenceFragment {

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

            // Preferences must ba an xml resource, not a layout resource.
            addPreferencesFromResource(R.xml.preferences);
        }
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Overview, Getting Started, https://developer.android.com/guide/topics/ui/settings.html#getting_started

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\SettingsActivity.java

If the app is targeting below Android 3.0 (API 11 HONEYCOMB) then extend PreferencesActivity as follows

```
public class SettingsActivity extends PreferenceActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Overview, Getting Started, https://developer.android.com/guide/topics/ui/settings.html#getting_started

Call your SettingsActivity from a "Settings" option menu item as follows.

```
public class MainActivity extends ActionBarActivity {

...

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.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();
        if (id == R.id.action_settings) {
            // Explicit intent
            startActivity(new Intent(this, SettingsActivity.class));
            return true;
        }
        return super.onOptionsItemSelected(item);
```

```
    }
...
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\MainActivity.java

Settings persist in a default SharedPreferences file. This is handled by the system, you don't have to do anything. However you'll probably need to read from this default SharedPreferences to adjust your app's behaviour based on the user chosen settings. You can code a listener to respond to a user change of a setting.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Overview, Getting Started, https://developer.android.com/guide/topics/ui/settings.html#getting_started

See Key-Value Sets (SharedPreferences API)

Default values

To properly set default values you must do this in two places: in the xml markup and from onCreate() in the MainActivity in code.

```
<!-- res/xml/preferences.xml  -->
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android"
                  ...>

    <EditTextPreference
        android:key="preference_cool_word"
        android:title="Cool Word"
        android:summary="@string/overwritten"
        android:defaultValue="Nice"
        android:dialogTitle="Cool Word" />

</PreferenceScreen>


// MainActivity.java
public class MainActivity extends ActionBarActivity {

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

        ...

        // Idealisation of default preferences must take place in the MainActivity,
        // Not SettingsActivity.
        // @readAgain=false ensures user saved preferences are not overridden.
        PreferenceManager.setDefaultValues(this, R.xml.preferences, false);
                    ...
    }
...
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Overview, Getting Started, https://developer.android.com/guide/topics/ui/settings.html#getting_started

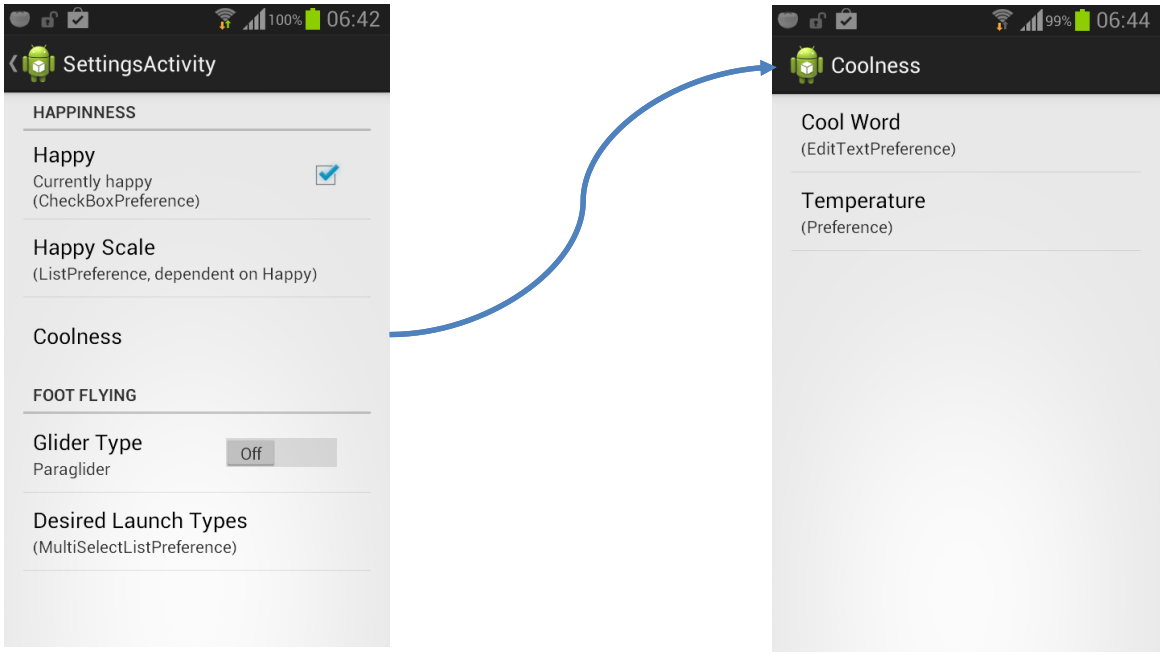
Grouping

Using PreferenceCategory or PreferenceScreen

Settings listed under a PreferenceScreen can be grouped by title, using a `PreferenceCategory` Object, or by a subcreen, using a nested `PreferenceScreen` Object.

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

    <PreferenceCategory
        android:key="preference_category_happiness"
        android:title="Happinness"
        >

        <CheckBoxPreference
            android:key="preference_is_happy"
            android:title="Happy"
            android:summary="Currently happy (CheckBoxPreference)"
            android:defaultValue="true"
            />

        <!-- We are using the same array for entries and values -->
        <ListPreference
            android:dependency="preference is happy"
            android:key="preference_happy_scale"
            android:title="Happy Scale"
            android:summary="(ListPreference, dependent on Happy)"
            android:defaultValue="Very"
            android:dialogTitle="Happy Scale"
            android:entries="@array/array_happy_scale"
            android:entryValues="@array/array_happy_scale"
            />
    </PreferenceCategory>

    <PreferenceScreen
        android:key="preference_screen_coolness"
        android:title="Coolness"
        >
        <EditTextPreference
            android:key="preference_cool_word"
            android:title="Cool Word"
            android:summary="(EditTextPreference)"
            android:defaultValue="Nice"
            android:dialogTitle="Cool Word"
            />

        <Preference
            android:key="preference_temperature"
            android:title="Temperature"
            android:summary="(Preference)"
            android:defaultValue="27"
            />
    </PreferenceScreen>

    <PreferenceCategory
        android:key="preference_category_foot_flying"
        android:title="Foot flying"
        >
        <SwitchPreference
            android:key="preference_glider_type"
            android:title="Glider Type"
            android:summaryOff="Paraglider"
            android:summaryOn="Hang gilder"
            android:switchTextOff="Off"
            android:switchTextOn="On"
            />

        <!-- We are using the same array for entries and values -->
        <MultiSelectListPreference
            android:key="preference_desired_launch_types"
            android:title="Desired Launch Types"
            android:summary="(MultiSelectListPreference)"
            android:defaultValue="@array/array_desired_launch_types_defaults"
```

```
            android:dialogTitle="Desired Launch Types"
            android:entries="@array/array_desired_launch_types"
            android:entryValues="@array/array_desired_launch_types"
            />
    </PreferenceCategory>

</PreferenceScreen>

<!-- /res/values/preferences_values.xml -->
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="array_happy_scale">
        <item>Mildly</item>
        <item>Very</item>
        <item>Extremely</item>
    </string-array>
    <string-array name="array_desired_launch_types">
        <item>Coastal</item>
        <item>Mountain</item>
        <item>Tow</item>
        <item>Winch</item>
    </string-array>
    <string-array name="array_desired_launch_types_defaults">
        <item>Coastal</item>
        <item>Winch</item>
    </string-array>
</resources>
```



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Organize your settings, https://developer.android.com/guide/topics/ui/settings/organize-your-settings

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\res\xml\preferences.xml
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\res\values\preferences_values.xml

Using Preference Headers

When you want your first screen to only (?) consist of preference subscreens you can use Preference Headers in place of PreferenceScreens. The advantage is that on tablets a two-pane layout is offered automatically.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Organize your settings, https://developer.android.com/guide/topics/ui/settings/organize-your-settings

Preference Headers are available from Android 3.0 (API 11) but there are workarounds if you are wanting to suppot earlier Android versions.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Organize your settings, https://developer.android.com/guide/topics/ui/settings/organize-your-settings

Opening Activities (with Intents)

You can use a preference to invoke an activity via an explicit or implicit intent (rather than another preference screen or dialogue box).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Organize your settings, https://developer.android.com/guide/topics/ui/settings/organize-your-settings

Implicit intent preference.

```
<!-- res/xml/preferences.xml -->
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android"
                  android:layout_width="match_parent"
                  android:layout_height="match_parent"
    >
...
<Preference
    android:title="Implicit Intent"
    android:summary="Open BOM webpage"
    >
    <intent
        android:action="android.intent.action.VIEW"
        android:data="http://m.bom.gov.au"
        />
</Preference>
...
</PreferenceScreen>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Organize your settings, https://developer.android.com/guide/topics/ui/settings/organize-your-settings

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\res\xml\preferences.xml

Explicit intent preference.

```
<!-- res/xml/preferences.xml -->
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android"
                  android:layout_width="match_parent"
                  android:layout_height="match_parent"
    >
...
<Preference
    android:title="Explicit Intent"
    android:summary="Open an app Activity"
    >
    <!-- * targetPackage and targetClass are both required
         * targetClass must start with package name even though it is declared
           in targetPackage
         * Ensure your activity is declared in the manifest -->
    <intent
        android:targetPackage="au.com.softmake.myfirstapp02"
        android:targetClass="au.com.softmake.myfirstapp02.BoundServiceClientActivity"
        />
</Preference>
```

```
...
</PreferenceScreen>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Organize your settings, https://developer.android.com/guide/topics/ui/settings/organize-your-settings

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\res\xml\preferences.xml

Reading Preferences

The `SharePreferences` API is different from the `Preference` API. The `Preference` API helps you build the UI for your app settings, although in turn, the `Preference` API uses the `SharedPreferences` API to save settings.

For coding with the `SharedPreferences` API see Key-Value Sets (SharedPreferences API).

To return and read Perference created through the UI use `PreferenceManager.getDefaultSharedPreferences()` to return a `SharedPreferences` object.

```
public void readSettings(View view) {
    SharedPreferences defaultSharedPreferences = PreferenceManager
            .getDefaultSharedPreferences(
            this);

    String coolWord = defaultSharedPreferences.getString(
            getResources().getString(R.string.preference_cool_word),
            getResources().getString(R.string.preference_cool_word_default_value));
    Boolean isHappy = defaultSharedPreferences.getBoolean(
            getResources().getString(R.string.preference is happy), Boolean.valueOf(
                    getResources().getString(
                            R.string.preference_is_happy_default_value)
            )
    );

    String message = "";
    message += String.format("coolWord: %s\n", coolWord);
    message += String.format("isHappy: %b\n", isHappy);

    TextView textView = (TextView) findViewById(R.id.textView_output);
    textView.setText(message);
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Organize your settings, https://developer.android.com/guide/topics/ui/settings/organize-your-settings

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\MainActivity.java

Listening for preference changes

To listen to preference changes you need to do several things:

- Implement an SharedPreferences.OnSharedPreferenceChangeListener in your PreferenceFragment or Activity as an Anonymous class. This enables you to have a module level listener variable which will resist garbage collection.
- Register this listener in onResume and onPause of your PreferenceFragement or Activity.
- Implement the onSharedPreferenceChanged() method of the listener class.

Implement an SharedPreferences.OnSharedPreferenceChangeListener in your PreferenceFragment or Activity as an Anonymous class. This enables you to have a module level listener variable which will resist garbage collection.

```
public class SettingsActivity extends Activity {

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

        // We don't need a user created activity_settings.xml layout file,
        // we use android's content resource
        if (savedInstanceState == null) {
            getFragmentManager().beginTransaction().replace(
                    android.R.id.content, new SettingsFragment()).commit();
        }
    }


    /**
     * A placeholder fragment containing a simple view.
     */
    public static class SettingsFragment extends PreferenceFragment {

        SharedPreferences.OnSharedPreferenceChangeListener listener = new
                SharedPreferences.OnSharedPreferenceChangeListener() {

            public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
                                                  String key) {

            // TODO
            }
        };

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

            // Preferences must ba an xml resource, not a layout resource.
            addPreferencesFromResource(R.xml.preferences);
        }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Organize your settings, https://developer.android.com/guide/topics/ui/settings/organize-your-settings

Register this listener in onResume and onPause of your PreferenceFragement or Activity.

```
public class SettingsActivity extends Activity {

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

        // We don't need a user created activity_settings.xml layout file,
        // we use android's content resource
        if (savedInstanceState == null) {
            getFragmentManager().beginTransaction().replace(
                    android.R.id.content, new SettingsFragment()).commit();
        }
    }
```

```
    /**
     * A placeholder fragment containing a simple view.
     */
    public static class SettingsFragment extends PreferenceFragment {

        SharedPreferences.OnSharedPreferenceChangeListener listener = new
                SharedPreferences.OnSharedPreferenceChangeListener() {

            public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
                                                  String key) {

               //TODO
            }
        };

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

            // Preferences must ba an xml resource, not a layout resource.
            addPreferencesFromResource(R.xml.preferences);
        }

        @Override
        public void onResume() {
            super.onResume();
            getPreferenceScreen().getSharedPreferences()
                    .registerOnSharedPreferenceChangeListener(
                    listener);

        }

        @Override
        public void onPause() {
            super.onPause();
            getPreferenceScreen().getSharedPreferences()
                    .unregisterOnSharedPreferenceChangeListener(
                    listener);
        }
    }
}
```

Implement the onSharedPreferenceChanged() method of the listener class (Full Example).

```
public class SettingsActivity extends Activity {

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

        // Preferences must ba an xml resource, not a layout resource.
//        addPreferencesFromResource(R.xml.preferences);

        // We don't need a user created activity_settings.xml layout file,
        // we use android's content resource
        if (savedInstanceState == null) {
            getFragmentManager().beginTransaction().replace(
                    android.R.id.content, new SettingsFragment()).commit();
        }
    }


    /**
     * A placeholder fragment containing a simple view.
     */
    public static class SettingsFragment extends PreferenceFragment {

        SharedPreferences.OnSharedPreferenceChangeListener listener = new
                SharedPreferences.OnSharedPreferenceChangeListener() {

            public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
                                                  String key) {

                // CheckBoxPreference
```

```
                if (key.equals(getResources().getString(R.string.preference_is_happy))) {
                    Preference isHappyPreference = findPreference(key);
                    Boolean isHappyPreferenceDefaultValue = Boolean.valueOf(
                            getResources().getString(
                                    R.string.preference_is_happy_default_value)
                    );

                    String text = String.valueOf(
                            sharedPreferences.getBoolean(
                                    key, isHappyPreferenceDefaultValue)
                    );
                    isHappyPreference.setSummary(text);

                // EditTextPreference
                } else if (key.equals(
                        getResources().getString(
                                R.string.preference_cool_word))) {

                    Preference coolWordPreference = findPreference(key);
                    String coolWordPreferenceDefaultValue = getResources().getString(
                            R.string.preference_is_happy_default_value);

                    String text = sharedPreferences.getString(
                            key, coolWordPreferenceDefaultValue);
                    coolWordPreference.setSummary(text);

                }
            }
        };

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

            // Preferences must ba an xml resource, not a layout resource.
            addPreferencesFromResource(R.xml.preferences);
        }

        @Override
        public void onResume() {
            super.onResume();
            getPreferenceScreen().getSharedPreferences()
                    .registerOnSharedPreferenceChangeListener(
                    listener);

        }

        @Override
        public void onPause() {
            super.onPause();
            getPreferenceScreen().getSharedPreferences()
                    .unregisterOnSharedPreferenceChangeListener(
                    listener);
        }
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Organize your settings, https://developer.android.com/guide/topics/ui/settings/organize-your-settings

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\SettingsActivity.java

Newtork Usage

If your app uses network data you should provide settings for the user to control when this occurs. For example, to choose between:

- Wifi or Mobile
- Wifi only
- None.

[TODO: Implement example of this]

From Android 4.0 Users can go Settings > Data Usage > [Click on App] > View App Settings. Register your relevant Settings screen with the "View App Settings" button with

```
<activity android:name="SettingsActivity" ... >
    <intent-filter>
       <action android:name="android.intent.action.MANAGE_NETWORK_USAGE" />
       <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Organize your settings, https://developer.android.com/guide/topics/ui/settings/organize-your-settings

Custom Preferences

You can create custom preferences.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Settings, Organize your settings, https://developer.android.com/guide/topics/ui/settings/organize-your-settings

Notifications

We are referring to notifications in the notification drawer (not toast notifications).

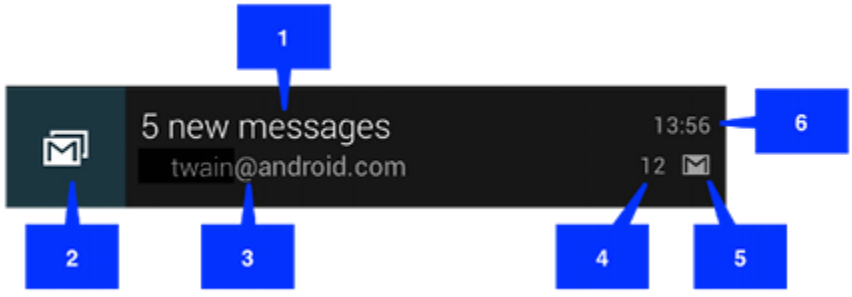
Notifications can either be ongoing or ad hoc.

Display Elements

Notifications appear in normal view or, as of Android 4.1, big view. A notification created with a Big View style will appear as a normal view until expanded.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Notifications, https://developer.android.com/guide/topics/ui/notifiers/notifications

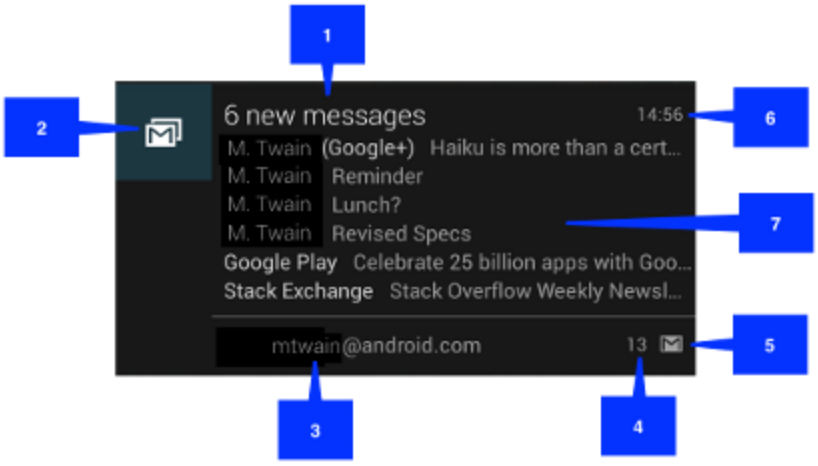
Normal View display elements...



1. Content title.

2. Large icon.
3. Content text.
4. Content info (or "Number").
5. Small icon.
6. Time of notification issue.

Big View adds an additional display element, the details area (7 below).



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Notifications, https://developer.android.com/guide/topics/ui/notifiers/notifications

The details area can be set with different Big View styles:

1. Big Picture style. A bitmap up to 256 dp tall.
2. Big text style. A large text block.
3. Inbox style. Lines of text.
4. Big view styles also have the following content options (not available in normal view):
5. Big content title. Overrided the normal view's content title when expanded into a Big View.
6. Summary text. A line of text below the details area.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Notifications, https://developer.android.com/guide/topics/ui/notifiers/notifications

Version support

Android 1.0: Use NotificationCompat.Builder from Support Library version 4.

Android 4.1.x (level 16): Use Notification.Builder.

Official documentation says ...

Android 1.0: Use NotificationCompat.Builder from Support Library version 4.

Android 3.0: Use Notification.Builder.

... but Notification.Builder.build() only supported from Android 4.1.x (level 16).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Notifications, https://developer.android.com/guide/topics/ui/notifiers/notifications

Creating a Notification

Create a normal notification by:

Import an icon into all the drawable-zzzz directories (to serve as the "small icon"). Setting the small icon will actually automatically resuse the relevant icon for the large icon display element (2 above).

Code as follows.

```
// A module level Builder object facilitates updating the
// notification
NotificationCompat.Builder mBuilder;

// An Id allows you to update the notification later.
int mNotificationID = 66;

public void goNotification(View view) {

           mBuilder = new NotificationCompat.Builder(this);
           mBuilder.setSmallIcon(R.drawable.ic_action_flash_on);
           mBuilder.setContentTitle("John Bentley Notification");
           mBuilder.setContentText("Everything is groovy and relaxed");
           // mBuilder.setContentInfo("12");
           mBuilder.setNumber(14); // or use setContentInfo
           // Dismiss the notification when the user presses it.
           mBuilder.setAutoCancel(true);

           setBigViewNotification(mBuilder);

           // Create an explicit intent for the target activity when the
           // notification is pressed.
           Intent targetIntent = new Intent(this, MainActivity.class);

           // Create a stack builder object in order to:
           // 1. Create an artificial back stack for the target activity. This
           // ensures that navigating backwards from the target activity leads out
           // of your application to the home screen.
           // 2. Get a pending intent object.
           TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
           // Add to the back stack the target class, not the target intent.
           stackBuilder.addParentStack(MainActivity.class);
           // Add the target intent.
           stackBuilder.addNextIntent(targetIntent);
           // Retrieve a pendingIntent Object
           PendingIntent pendingIntent =
                                   stackBuilder.getPendingIntent(0,

           PendingIntent.FLAG_UPDATE_CURRENT);

           // Set the pending intent for a user action, here pressing on the
           // notification.
           mBuilder.setContentIntent(pendingIntent);

           // Execute the notification
           NotificationManager notificationManager =
                                    (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
           notificationManager.notify(mNotificationID, mBuilder.build());
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Notifications, https://developer.android.com/guide/topics/ui/notifiers/notifications
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\MainActivity.java

Create a big view style notification as follows.

```
// Uncomment "setBigViewNotification(builder);" from  goNotificationNormal() above, to
// call the following.

private void setBigViewNotification(NotificationCompat.Builder builder) {
          NotificationCompat.InboxStyle inboxStyle =
                                    new NotificationCompat.InboxStyle();
          inboxStyle.setBigContentTitle("John Bentley Big View Content Title");
          String[] items = { "Apple", "Banana", "Tomato", "Theodilite" };
          for (String item : items) {
                    inboxStyle.addLine(item);
          }
          builder.setStyle(inboxStyle);
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Notifications, https://developer.android.com/guide/topics/ui/notifiers/notifications
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\MainActivity.java

Updating a notification

Update a notification by availing yourself of a module level (class level) Builder object and a module level NotificationID.

```
int mCount = 1;

public void goNotificationUpdate(View view) {
          if (mBuilder != null) {
                    mBuilder.setContentText("Notification updatedx ...");
                    mBuilder.setNumber(mCount++);
                    // Execute the notification
                    NotificationManager notificationManager =
                                              (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
                    notificationManager.notify(mNotificationID, mBuilder.build());
          } else {
                    Toast.makeText(getApplicationContext(),
                                        "Send a notification before updating it",
                                        Toast.LENGTH_SHORT).show();
          }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Notifications, https://developer.android.com/guide/topics/ui/notifiers/notifications
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\MainActivity.java

Progress indicators

Progress indicators can be determinate (aka "progress bar") and indeterminate (aka "activity indicator").

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Notifications, https://developer.android.com/guide/topics/ui/notifiers/notifications

Prior to Android 4.0, just use version handling code to not call setProgress(). After Android 4.0 call `setProgress()`.

For handling devices prior to Android 4 you could custom notification layouts. However, it is probably simpler just use version handling code to not call setProgress(). Instead let the updating of the notification provide the feedback (e.g. by changing the contextText to display "Notification updated ... 20%").

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Notifications, https://developer.android.com/guide/topics/ui/notifiers/notifications

However, for handling devices prior to Android 4 you could just use version handling code to not call setProgress(). Instead let the updating of the notification provide the feedback (e.g. by changing the contextText to display "Notification updated ... 20%").

For a determinate progress indicator using setProgress(): Call setProgress as your operation proceeds, updating the notification each time.

```
// E.g within an IntentService

@Override
protected void onHandleIntent(Intent intent) {

           String contentText = "";

           if (mBuilder != null) {

                      NotificationManager notificationManager =
                                                 (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

                      for (int i = 0; i <= 100; i += 20) {
                                 // Normally work is done, but just sleep for 5 seconds in
this
                                 // example
                                 sleep(1);

                                 contentText = String.format("Notification updated ...
%d%%", i);
                                 mBuilder.setContentText(contentText);
                                 mBuilder.setProgress(100, i, false);
                                 notificationManager.notify(mNotificationID,
mBuilder.build());

                      } // for (int i ...

           } else {
                      Toast.makeText(getApplicationContext(),
                                                 "Send a notification before updating it",
                                                 Toast.LENGTH_SHORT).show();
           }

           // Stops the service after all start requests have been handled,
           // so you never have to call stopSelf().
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Notifications, https://developer.android.com/guide/topics/ui/notifiers/notifications
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\HelloIntentService.java

To display an indeterminate progress indicator (aka "a continuing activity") use ...

```
mBuilder.setProgress(0, 0, true);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Notifications, https://developer.android.com/guide/topics/ui/notifiers/notifications
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\HelloIntentService.java

To remove a progress bar with setProgress() ...

```
mBuilder.setProgress(0, 0, false);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Notifications, https://developer.android.com/guide/topics/ui/notifiers/notifications

Other Notification techniques

Other things you can do, or have to do, with notifications:

- More fully handling the activity started with the notification (and handling the backstack). http://developer.android.com/guide/topics/ui/notifiers/notifications.html#NotificationResponse
- Create a custom notification layout (that normal and big view styles don't accommodate). http://developer.android.com/guide/topics/ui/notifiers/notifications.html#CustomNotification

Toast (Deprecated)

Basic toast.

```
String text = "No such field in contact to return";
Toast toast = Toast.makeText(getApplicationContext(), text, Toast.LENGTH_SHORT);
toast.show();
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Toasts, https://developer.android.com/guide/topics/ui/notifiers/toasts

Basic toast on one line.

```
Toast.makeText(getApplicationContext(), "Hello", Toast.LENGTH_SHORT).show();
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Toasts, https://developer.android.com/guide/topics/ui/notifiers/toasts

You can also position your toast, rather than rely on the default position.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Toasts, https://developer.android.com/guide/topics/ui/notifiers/toasts

You can create a custom toast view.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Toasts, https://developer.android.com/guide/topics/ui/notifiers/toasts

Pop-up messages (SnackBar)

For brief pop-up messages use a SnackBar, not a Toast. Toast is deprecated.

> The Snackbar class supersedes Toast. While Toast is currently still supported, Snackbar is now the preferred way to display brief, transient messages to the user.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Pop up messages, https://developer.android.com/training/snackbar/

SnackBar has the advantage over Toasts in optionally providing an action for the user to preform as a response to the brief message.

To create a snackbar:

1. Define your UI within a CoordinatorLayout with some event to trigger the pop-up message (usually some kind of button).

```
<?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"
    tools:context=".SnackbarDemoActivity"
    android:id="@+id/myCoordinatorLayout">

  ....

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout behavior="@string/appbar scrolling view behavior">

        <Button
            android:id="@+id/button_fixed_go"
            android:layout_width="wrap_content"
            android:layout height="wrap content"
            android:layout_marginStart="8dp"
            android:layout_marginLeft="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginRight="8dp"
            android:layout marginBottom="8dp"
            android:text="Fixed Go"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.303"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.266" />

    </android.support.constraint.ConstraintLayout>

    <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"
        app:srcCompat="@android:drawable/ic_media_play" />

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

The Snackbar gets additional features, user can dismiss the message and the layout can move other UI elements, when attached to a CoordinatorLayout.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Pop up messages, Build and display a pop-up message, https://developer.android.com/training/snackbar/showing

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_snackbar_demo.xml

2. From within event listener, e.g. a button OnClick listener, pass the CoordinatorLayout to the SnackBar make constructor.

```
public class SnackbarDemoActivity extends Activity implements View.OnClickListener {
```

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

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(this);

        Button buttonFixedGo = (Button) findViewById(R.id.button_fixed_go);
        buttonFixedGo.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {

        String userMessage = "";
        switch (v.getId()) {
            case R.id.button_fixed_go:
                userMessage = "From button_fixed_go";
                break;
            case R.id.fab:
                userMessage = "From fab";
                break;
            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }

        View myCoordinatorView = findViewById(R.id.myCoordinatorLayout);
        Snackbar snackbar = Snackbar.make(myCoordinatorView, userMessage,
Snackbar.LENGTH_LONG);
        snackbar.show();

    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Pop up messages, Build and display a pop-up message, https://developer.android.com/training/snackbar/showing

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\SnackbarDemoActivity.java

3. If you want to include an action the user can respond to then: create a class that implements `View.OnClickListener` and perform the action in `onClick`; pass an instance of this class the the `snackbar.setAction` method.

```
public class MySnackBarAction implements View.OnClickListener {

    @Override
    public void onClick(View v) {
        Toast.makeText(getApplicationContext(), "Done MySnackBarAction",
Toast.LENGTH_SHORT).show();
    }
}

...

View myCoordinatorView = findViewById(R.id.myCoordinatorLayout);

Snackbar snackbar = Snackbar.make(myCoordinatorView, userMessage,
Snackbar.LENGTH_LONG);
snackbar.setAction("MySnackBarAction", new MySnackBarAction());
snackbar.show();
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Pop up messages, Build and display a pop-up message, https://developer.android.com/training/snackbar/showing

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\SnackbarDemoActivity.java

With chaining you can create the Snackbar in one line.

```
Snackbar.make(findViewById(R.id.myCoordinatorLayout), userMessage,
Snackbar.LENGTH_LONG).setAction("MySnackBarActionx", new MySnackBarAction()).show()
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Pop up messages, Build and display a pop-up message, https://developer.android.com/training/snackbar/showing

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\SnackbarDemoActivity.java

Floating Action Button

> A floating action button (FAB) is a circular button that triggers the primary action in your app's UI.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Look and feel, Add a Floating Action Button, https://developer.android.com/guide/topics/ui/floating-action-button

Floating action button XML

```
<?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"
    tools:context=".SnackbarDemoActivity"
    android:id="@+id/myCoordinatorLayout">

   ....

    <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"
        app:srcCompat="@android:drawable/ic_media_play" />

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

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Look and feel, Add a Floating Action Button, https://developer.android.com/guide/topics/ui/floating-action-button
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_snackbar_demo.xml

Respond to a click example ...

```
public class SnackbarDemoActivity extends Activity implements View.OnClickListener {

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

```
        setContentView(R.layout.activity_snackbar_demo);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(this);

...

    }

    @Override
    public void onClick(View v) {

        String userMessage = "";
        switch (v.getId()) {
            case R.id.button_fixed_go:
                userMessage = "From button_fixed_go";
                break;
            case R.id.fab:
                userMessage = "From fab";
                break;
            default:
                try {
                    throw new Exception("Unhandled case in switch");
                } catch (Exception e) {
                    e.printStackTrace();
                }
        }

        View myCoordinatorView = findViewById(R.id.myCoordinatorLayout);
        Snackbar snackbar = Snackbar.make(myCoordinatorView, userMessage,
Snackbar.LENGTH_LONG);
        snackbar.show();

    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Look and feel, Add a Floating Action Button, https://developer.android.com/guide/topics/ui/floating-action-button
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\SnackbarDemoActivity.java

Search

Overview

For a search example that uses a Loader see Refresh data (Exemplified by implementing Search) .

The Android Search framework offers two modes of search input:

1. Search dialog; or
2. Search widget (a SearchView that you can embed in your activity layout).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, https://developer.android.com/guide/topics/search/

You can optionally configure features of the search input to:

- Enable voice search.
- Provide search suggestions: based on recent user queries; and/or based on results from application data.
- Offer your applications search suggestions to the entire system.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, https://developer.android.com/guide/topics/search/

Search checklist:

- The search framework does not do the searching as such. You must hook it up to a data API (e.g. android.database.sqlite).
- When setting up a search input you'll need to manually provide a search button.
- Always provide a clearHistory() function, so the user can protect their privacy.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, https://developer.android.com/guide/topics/search/

Creating Search

The workflow:

- The user executes search using a Search Dialog or Search Widget. The system creates an Intent with the user's query stored.
- The system starts the "Searchable Activity" that you've declared in the manifest.
- The system passes the Intent to the Searchable Activity.
- The Searchable Activity receives the query, performs the search, and displays the results.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, Creating a Search Interface, https://developer.android.com/guide/topics/search/search-dialog

Create a search initiating component (Search dialog or search widget)

For Android 3.0 and above prefer a search widget rather than search dialog.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, Creating a Search Interface, https://developer.android.com/guide/topics/search/search-dialog

If your app uses an App Bar then you should use a search widget (as a collapsible view in the app bar) rather than search dialog.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, Creating a Search Interface, https://developer.android.com/guide/topics/search/search-dialog

You can choose to have the search initiating UI attached to the "Searchable activity" or attach it to another activity, that passes the query to the "Searchable activity".

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, Creating a Search Interface, https://developer.android.com/guide/topics/search/search-dialog

Search Dialog

Don't use. Use the Search Widget instead given you'll, at this point, be targeting android versions higher than 3.0.

Search(View) Widget

To search with a Search(View) Widget:

1. Create a Searchable Configuration, configuration info for the Search Widget (or Search Dialog). See Searchable Configuration.

   Searchable configuration. An xml file with settings for the search dialog or search widget. Create \res\xml\searchable.xml

```
<?xml version="1.0" encoding="utf-8"?>
<searchable xmlns:android="http://schemas.android.com/apk/res/android"
    android:label="@string/app_name"
    android:hint="@string/search_hint">
</searchable>
```

   (Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, Creating a Search Interface, https://developer.android.com/guide/topics/search/search-dialog
   C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\xml\searchable.xml

2. Create a Searchable Activity, an activity that receives the search string, performs the search, and displays results.
   - Create blank activity.
   - Create a manifest entry with the relevant searchable intent and metadata. Be sure to point to the Searchable configuration.

```
<activity
    android:name=".SearchableActivity"
    android:label="@string/title_activity_searchable">
    <intent-filter>
        <action android:name="android.intent.action.SEARCH" />
    </intent-filter>
    <meta-data
        android:name="android.app.searchable"
        android:resource="@xml/searchable" />
</activity>
```

   (Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, Creating a Search Interface, https://developer.android.com/guide/topics/search/search-dialog
   C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\AndroidManifest.xml

3. With the SearchableActvitiy process the query:
   1. Receive the query.
   2. Search your data.
   3. Present the results.

   (Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, Creating a Search Interface, Creating a Searchable Activity, Performing a search, https://developer.android.com/guide/topics/search/search-dialog.html#PerformingSearch

To receive the query you test for an ACTION_SEARCH intent, extract the bundled query string, and use that query string to perform your search and return results.

```
package au.com.softmake.mysecondapp;

import android.app.Activity;
import android.app.SearchManager;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;

import java.util.regex.Pattern;

public class SearchableActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity searchable);
        handleSearch(getIntent());
    }

    private void handleSearch(Intent intent) {
        StringBuilder result = new StringBuilder();

        if (Intent.ACTION_SEARCH.equals(intent.getAction())) {
            Boolean ismatch = false;

            // Simplified Database is just a string array
            String[] countriesDB =
getResources().getStringArray(R.array.countries_array);
            // Fetch searchString (SearchManager.QUERY) from intent.
            String searchString = intent.getStringExtra(SearchManager.QUERY);

            // Perform search and build result
            String regex = new String(searchString);
            String stringToSearch = new String();
            for (String country : countriesDB) {
                stringToSearch = country;
                // Does the regex match part of the stringToSearch
                if (Pattern.compile(regex).matcher(stringToSearch).find()) {
                    ismatch = true;
                    result.append(country + "\r\n");
                }
            }
            if (!ismatch) {
                result.append("No countries matched against " + regex
                        .toString());
            }
        } else {
            result.append("No search string was passed to " + this
                    .getLocalClassName());
        }

        // Display search result
        TextView searchResultTextView = (TextView) findViewById(R.id.search_result);
        searchResultTextView.setText(result.toString());
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, Creating a Search Interface, Creating a Searchable Activity, Performing a search, https://developer.android.com/guide/topics/search/search-dialog.html#PerformingSearch

Searching your data is particular to your app. You could search against a SQLite database or an online database. Use the relevant Android classes for that. E.g. `SQLiteDatabase` or `android.net`.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, Creating a Search Interface, Creating a Searchable Activity, Performing a search, https://developer.android.com/guide/topics/search/search-dialog.html#PerformingSearch

Presenting your results could also be done in any format but a `ListView` is the usual format. In which case you'll generally want to use an `Adapter` (e.g. a `CursorAdapter` or `BaseAdapter`).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, Creating a Search Interface, Creating a Searchable Activity, Performing a search, https://developer.android.com/guide/topics/search/search-dialog.html#PerformingSearch

4. Optionally create a search initiating Activity, an activity which launches your search, and fetches the search string from the user.

```
public class SearchInitiatingActivity extends AppCompatActivity {

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

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

        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    }

    @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_initiating, menu);

        SearchManager searchManager = (SearchManager) getSystemService
                (Context.SEARCH_SERVICE);
        SearchView searchView = (SearchView)
menu.findItem(R.id.app_bar_search).getActionView();

        // Explicitly fetch SearchableActivity name
        ComponentName SearchableActivityComponentName = new ComponentName(this,

SearchableActivity.class);
        searchView.setSearchableInfo(searchManager.getSearchableInfo(
                SearchableActivityComponentName));
        // Do not iconify the widget; expand it by default
        searchView.setIconifiedByDefault(false);

        return true;
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\SearchableActivity.java

5. Alternatively, or in addition, search initiating can be enabled from the Searchable Activity, so the Searchable Activity does all of the search work: search initialization, receive search string, perform the search, and display the results.

```
public class SearchableActivity extends AppCompatActivity {
   @Override
    protected void onCreate(Bundle savedInstanceState) {
        // Layout
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_searchable);

        // Toolbar
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        // Work
        handleSearch(getIntent());
    }
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the options menu from XML
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_searchable, menu);

        // Get the SearchView and set the searchable configuration
        SearchManager searchManager = (SearchManager) getSystemService
                (Context.SEARCH_SERVICE);
        SearchView searchView = (SearchView)
menu.findItem(R.id.app_bar_search).getActionView();
        // Assumes current activity is the searchable activity
        searchView.setSearchableInfo(searchManager.getSearchableInfo(
                getComponentName()));
        getComponentName();

        // For use in a SearchableActivity
        ComponentName SearchableActivityComponentName = getComponentName();
        searchView.setSearchableInfo(searchManager.getSearchableInfo(
                SearchableActivityComponentName));

        searchView.setIconifiedByDefault(false); // Do not iconify the
        // widget; expand it by default

        return true;
    }

    private void handleSearch(Intent intent) {
...
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, Creating a Search Interface, Using the Search widget, Configuraing the search widget, https://developer.android.com/guide/topics/search/search-dialog.html#ConfiguringWidget

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\SearchableActivity.java

6. Embed a SearchView (or the other search initiating component Search Dialog) as an action view on the app bar of your search initiating activity.

```
<?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="au.com.softmake.mysecondapp.SearchInitiatingActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        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"
            android:minHeight="?attr/actionBarSize"
            android:theme="@style/ToolbarCustomTheme"
            app:popupTheme="@style/ToolbarPopupCustomTheme" />

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

    <RelativeLayout
        android:id="@+id/content_search_initiating"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
```

```
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        android:paddingBottom="@dimen/activity_vertical_margin"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        tools:showIn="@layout/activity_search_initiating">

        <TextView
            android:text="TextView Blahx"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true"
            android:layout_marginTop="19dp"
            android:id="@+id/textView" />

    </RelativeLayout>

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

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\activity_search_initiating.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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="au.com.softmake.mysecondapp.SearchInitiatingActivity">

    <item
        android:id="@+id/app_bar_search"
        android:icon="@drawable/ic_action_search"
        android:title="@string/action_search"
        app:actionViewClass="android.widget.SearchView"
        app:showAsAction="ifRoom|collapseActionView" />

</menu>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\menu\menu_search_initiating.xml

7. Alternatively, or in addition, embed a SearchView (or the other search initiating component Search Dialog) as an action view on the app bar of your SearchableActivity.

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\SearchableActivity.java
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\activity_searchable.xml
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\menu\menu_searchable.xml

Add Search features

Optionally add any search features (voice search, search suggestions, a submit button, query refinement for search suggestions, toggle search box visibility).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Search, Creating a Search Interface, Using the Search widget, Othe search widget features, https://developer.android.com/guide/topics/search/search-dialog.html#WidgetFeatures

https://developer.android.com/guide/topics/search/search-dialog.html#VoiceSearch
https://developer.android.com/guide/topics/search/search-dialog.html#SearchSuggestions

Drag and Drop

Drag and Drop.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Drag and drop, https://developer.android.com/guide/topics/ui/drag-drop

Add drag and drop support to a UI as follows:

1. Create UI having some Views to drag.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
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"
    tools:context=".draganddropdemo.DragAndDropDemoActivity">

    <EditText
        android:id="@+id/editText01"
        android:layout_width="wrap_content"
        android:layout_height="40sp"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="@android:color/holo_blue_light"
        android:ems="10"
        android:enabled="true"
        android:gravity="center"
        android:inputType="none"
        android:selectAllOnFocus="false"
        android:text="editText01"
        android:textColor="@android:color/white"
        app:layout_constraintBottom_toTopOf="@+id/editText02"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/editText02"
        android:layout_width="wrap_content"
        android:layout_height="40sp"
        android:background="@android:color/holo_blue_light"
        android:ems="10"
        android:enabled="true"
        android:gravity="center"
        android:inputType="none"
        android:selectAllOnFocus="false"
```

```
        android:text="editText02"
        android:textColor="@android:color/white"
        android:textIsSelectable="false"
        app:layout_constraintBottom_toTopOf="@+id/editText03"
        app:layout_constraintStart_toStartOf="@+id/editText01"
        app:layout_constraintTop_toBottomOf="@+id/editText01" />

    <EditText
        android:id="@+id/editText03"
        android:layout_width="wrap_content"
        android:layout_height="40sp"
        android:background="@android:color/holo_blue_light"
        android:ems="10"
        android:enabled="true"
        android:gravity="center"
        android:inputType="none"
        android:selectAllOnFocus="false"
        android:text="editText03"
        android:textColor="@android:color/white"
        android:textIsSelectable="false"
        app:layout_constraintBottom_toTopOf="@+id/editText_output"
        app:layout_constraintStart_toStartOf="@+id/editText01"
        app:layout_constraintTop_toBottomOf="@+id/editText02" />

    <EditText
        android:id="@+id/editText output"
        android:layout_width="0dp"
        android:layout_height="89dp"
        android:layout_marginBottom="8dp"
        android:background="@android:color/background_light"
        android:ems="10"
        android:focusable="false"
        android:gravity="top"
        android:inputType="textMultiLine"
        android:padding="8sp"
        android:text="editText_output"
        android:textSize="14sp"
        app:layout constraintBottom toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="@+id/editText03"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="@+id/editText03"
        app:layout_constraintTop_toBottomOf="@+id/editText03" />

</android.support.constraint.ConstraintLayout>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_drag_and_drop_demo.xml

2. Create an Activity and set the UI.

```
public class DragAndDropDemoActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_drag_and_drop_demo);
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\draganddropdemo\DragAndDropDemoActivity.java

3. Create your own DragShowdowBuilder by subclassing View.DragShadowBuilder. You can do this in an independent class.

```
public class MyDragShadowBuilder extends View.DragShadowBuilder {

    // The drag shadow image, defined as a drawable thing.
    private static Drawable shadowDrawable;


    public MyDragShadowBuilder(View view) {
        super(view);

        // Creates a draggable image that will fill the Canvas provided by the system.
        shadowDrawable = new ColorDrawable(Color.LTGRAY);
    }

    // Defines a callback that sends the drag shadow dimensions and touch point back
to the
    // system.
    @Override
    public void onProvideShadowMetrics(Point outShadowSize, Point outShadowTouchPoint)
{
        // Unclear if super should be called, but it works with or without the call.
        super.onProvideShadowMetrics(outShadowSize, outShadowTouchPoint);
        int width, height;

        // Sets the width of the shadow to half the width of the original View
        width = getView().getWidth() / 2;
        height = getView().getHeight() / 2;

        // The drag shadow is a ColorDrawable. This sets its dimensions to be the same
as the
        // Canvas that the system will provide. As a result, the drag shadow will fill
the
        // Canvas.
        shadowDrawable.setBounds(0, 0, width, height);

        // Sets the size parameter's width and height values. These get back to the
system
        // through the outShadowSize parameter.
        outShadowSize.set(width, height);

        // Sets the touch point's position to be in the middle of the drag shadow
        outShadowTouchPoint.set(width / 2, height / 2);
    }

    // Defines a callback that draws the drag shadow in a Canvas that the system
constructs
    // from the dimensions passed in onProvideShadowMetrics().
    @Override
    public void onDrawShadow(Canvas canvas) {
        // Unclear if super should be called, but it works with or without the call.
        super.onDrawShadow(canvas);
        shadowDrawable.draw(canvas);
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\draganddropdemo\MyDragShadowBuilder.java

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Drag and drop, Designing a Drag and Drop Operation, Starting a drag, https://developer.android.com/guide/topics/ui/drag-drop#StartDrag

4. In your DragAndDropDemoActivity, from a View's event, typically a Long Click,: build some ClipData; instantiate your custom drag shadow builder; and start the drag. Set this up for all the views that you want to drag from.

```
public class DragAndDropDemoActivity extends Activity {

    class MyOnLongClickListener implements View.OnLongClickListener {
        @Override
        public boolean onLongClick(View view) {

            // Build ClipData
            ClipData.Item clipDataItem = new ClipData.Item((String) view.getTag());
            String[] mimeTypes = {ClipDescription.MIMETYPE_TEXT_PLAIN};
            ClipData clipData = new ClipData((String) view.getTag(), mimeTypes,
clipDataItem);

            Toast.makeText(getApplicationContext(), clipData.toString(),
Toast.LENGTH_SHORT).show();
            EditText editTextOutput = findViewById(R.id.editText_output);
            editTextOutput.setText(clipData.toString());

            // Instantiate the drag shadow builder.
            View.DragShadowBuilder myDragShadowBuilder = new
MyDragShadowBuilder(view);

            // Start the drag
            view.startDrag(clipData,
                    myDragShadowBuilder,
                    null, // No need to use local data
                    0); // Flags (not currently used, set to 0
            return false;
        }
    }

    EditText editText01 = null;
    EditText editText02 = null;
    EditText editText03 = null;

    private void setupEditText(EditText editText, String tagString) {
        editText.setTag(tagString);
        editText.setOnLongClickListener(new MyOnLongClickListener());
    }

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

        // Prevent on-screen keyboard from popping up.
        // Do this in the manifest.
//
this.getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);

        editText01 = findViewById(R.id.editText01);
        editText02 = findViewById(R.id.editText02);
        editText03 = findViewById(R.id.editText03);

        setupEditText(editText01, "Some String 01");
        setupEditText(editText02, "Some String 02");
        setupEditText(editText03, "Some String 03");
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\draganddropdemo\DragAndDropDemoActivity.java

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics,

User interface, Drag and drop, Designing a Drag and Drop Operation, Starting a drag, https://developer.android.com/guide/topics/ui/drag-drop#StartDrag

5. Respond to drag events by implementing a custom View.OnDragListener. For the views that you want to respond to events, set this custom View.OnDragListener.

```
public class DragAndDropDemoActivity extends Activity {

    class MyOnLongClickListener implements View.OnLongClickListener {
        @Override
        public boolean onLongClick(View view) {
...
        }
    }

    class MyDragListener implements View.OnDragListener {

        @Override
        public boolean onDrag(View v, DragEvent event) {
            final int action = event.getAction();

            switch(action){
                case DragEvent.ACTION_DRAG_STARTED:

                    // Determines if this View can accept the dragged data
                    if
(event.getClipDescription().hasMimeType(ClipDescription.MIMETYPE_TEXT_PLAIN)) {

                        // An example of what your app might do:
                        // apply a tint to the View to indicate it can accept data
                        v.getBackground().setColorFilter(Color.GREEN,
PorterDuff.Mode.DARKEN);

                        // Invalidate the View to force a redraw in the new tint
                        v.invalidate();

                        // Return true to indicat that the view can accept the dragged
data
                        return true;
                    }

                case DragEvent.ACTION_DRAG_ENTERED:
                    v.getBackground().setColorFilter(Color.MAGENTA,
PorterDuff.Mode.DARKEN);
                    v.invalidate();
                    // The return value is ignored
                    return true;

                case DragEvent.ACTION_DRAG_LOCATION:
                    // Ignore the event
                    return true;

                case DragEvent.ACTION_DRAG_EXITED:
                    // Re-sets the color tint.
                    v.getBackground().setColorFilter(Color.GREEN,
PorterDuff.Mode.DARKEN);
                    v.invalidate();
                    return true;

                case DragEvent.ACTION_DROP:

                    ClipData.Item clipDataItem = event.getClipData().getItemAt(0);

                    String clipDataItemText = clipDataItem.getText().toString();

                    String displayText = "The dragged data: " + clipDataItemText;

                    Toast.makeText(getApplicationContext(), displayText,
Toast.LENGTH_SHORT).show();
                    EditText editTextOutput = findViewById(R.id.editText_output);
                    editTextOutput.setText(displayText);

                    // Set the target view's text with the dragged data.
                    EditText editText = (EditText)v;
```

```
                    editText.setText(clipDataItemText);

                    v.getBackground().clearColorFilter();
                    v.invalidate();

                    // Returns true. DragEvent.getResult() will return true.
                    return true;

                case DragEvent.ACTION_DRAG_ENDED:

                    v.getBackground().clearColorFilter();
                    v.invalidate();

                    if (event.getResult()) {
                        Toast.makeText(getApplicationContext(), "The drop was
handled.", Toast.LENGTH_LONG);
                    } else {
                        Toast.makeText(getApplicationContext(), "No drop was made.",
Toast.LENGTH_LONG);
                    }

                    // returns true; the value is ignored.
                    return true;
                default:
                    Log.e("DragDrop Example","Unknown action type received by
OnDragListener.");
                    break;
            }
            return false;
        }
    }

    EditText editText01 = null;
    EditText editText02 = null;
    EditText editText03 = null;

    private void setupEditText(EditText editText, String tagString) {
        editText.setTag(tagString);
        editText.setOnLongClickListener(new MyOnLongClickListener());
        editText.setOnDragListener(new MyDragListener());
    }

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

        // Prevent on-screen keyboard from popping up.
        // Do this in the manifest.
//
this.getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);

        editText01 = findViewById(R.id.editText01);
        editText02 = findViewById(R.id.editText02);
        editText03 = findViewById(R.id.editText03);

        setupEditText(editText01, "Some String 01");
        setupEditText(editText02, "Some String 02");
        setupEditText(editText03, "Some String 03");
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\draganddropdemo\DragAndDropDemoActivity.java

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User interface, Drag and drop, Designing a Drag and Drop Operation, Responding to drag events: an example, https://developer.android.com/guide/topics/ui/drag-drop#RespondEventSample

2D-Drawing

Images and Graphics

Drawables

See also Drawables, above.

Vector Drawables

Overview

A `VectorDrawable` is an Android specific file type: an XML file defining points, lines, curves and color information.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Images & Graphics, Vector drawables, https://developer.android.com/guide/topics/graphics/vector-drawable-resources

The major advantage of a `VectorDrawable`, as for an SVG file, is scalability without loss of resolution. VectorDrawables are also generally smaller in file size compared to bitmaps.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Images & Graphics, Vector drawables, https://developer.android.com/guide/topics/graphics/vector-drawable-resources

To optimize VectorDrawables:

- Limit to a maximum of 200 X 200 dp ?; and
- Use black (android:fillColor="#FF000000") then add a tint ?

(Google, n.d., "Android Studio User Guide", https://developer.android.com/studio/intro), Write your app, Add multi-density vector graphics, https://developer.android.com/studio/write/vector-asset-studio

A VectorDrawable file looks like this:

```
<!-- res/drawable/battery_charging.xml -->
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    <!-- intrinsic size of the drawable -->
    android:height="24dp"
    android:width="24dp"
    <!-- size of the virtual canvas -->
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">
   <group
         android:name="rotationGroup"
         android:pivotX="10.0"
         android:pivotY="10.0"
         android:rotation="15.0" >
      <path
        android:name="vect"
        android:fillColor="#FF000000"
        android:pathData="M15.67,4H14V2h-4v2H8.33C7.6,4 7,4.6
7,5.33V9h4.93L13,7v2h4V5.33C17,4.6 16.4,4 15.67,4z"
        android:fillAlpha=".3"/>
      <path
        android:name="draw"
        android:fillColor="#FF000000"
        android:pathData="M13,12.5h2L11,20v-5.5H9L11.93,9H7v11.67C7,21.4 7.6,22
8.33,22h7.33c0.74,0 1.34,-0.6 1.34,-1.33V9h-4v3.5z"/>
   </group>
```

```
</vector>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\drawable\battery_charging.xml

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Images & Graphics, Vector drawables, https://developer.android.com/guide/topics/graphics/vector-drawable-resources

Stucture details

The structure of the `vectorDrawable` is a heirachy of groups and paths. "`path` contains the geometry of the object's outline and `group` contains details for transformation."

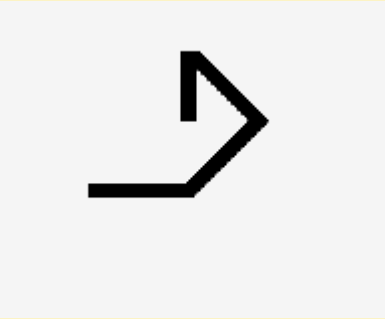
That is, the basic unit is a path. A path either defines a series of lines that, if connected, form a shape.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Images & Graphics, Vector drawables, https://developer.android.com/guide/topics/graphics/vector-drawable-resources

(Lockwood 2016, "An Introduction to Icon Animation Techniques", https://www.androiddesignpatterns.com/2016/11/introduction-to-icon-animation-techniques.html), https://www.androiddesignpatterns.com/2016/11/introduction-to-icon-animation-techniques.html

A path's geometry is defined by the pathData attribute. The pathDate contains some (?) subset of pathData defined at https://www.w3.org/TR/SVG11/paths.html#PathData

```
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:height="64dp"
    android:width="64dp"
    android:viewportHeight="600"
    android:viewportWidth="600">
    <!--android:fillColor="#000000"-->
    <group
        android:name="rotationGroup"
        android:pivotX="300.0"
        android:pivotY="300.0"
        android:rotation="0.0">
        <path
            android:name="vectorPath"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:strokeWidth="15"
            android:strokeColor="#000000"
            android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70 h-100" />
    </group>
</vector>
```



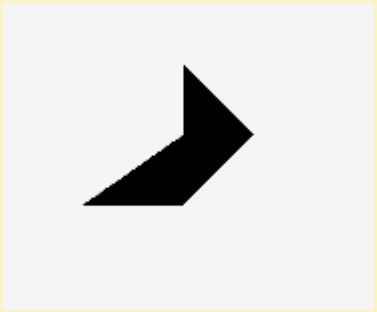
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\drawable\vector_drawable.xml
(W3C, The Latest. Scalable Vector Graphics (SVG) 1.1 (Second Edition), https://www.w3.org/TR/SVG11/)
https://www.w3.org/TR/SVG11/paths.html#PathData

(W3C The latest, "Scalable Vector Graphics (SVG) 1.1 (Second Edition)", https://www.w3.org/TR/SVG11/), https://www.w3.org/TR/SVG11/

(Lockwood 2016, "An Introduction to Icon Animation Techniques", https://www.androiddesignpatterns.com/2016/11/introduction-to-icon-animation-techniques.html), https://www.androiddesignpatterns.com/2016/11/introduction-to-icon-animation-techniques.html

The path's shape is either stroked or filled. For example, following on from above we can fill the shape ...

```
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="64dp"
    android:height="64dp"
    android:viewportHeight="600"
    android:viewportWidth="600">
    <group
        android:name="rotationGroup"
        android:pivotX="300.0"
        android:pivotY="300.0"
        android:rotation="0.0">
        <!--android:strokeWidth="15"-->
        <!--android:strokeColor="#000000"-->
        <path
            android:name="vectorPath"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:fillColor="#000000"
            android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70 h-100" />
    </group>
</vector>
```



(Lockwood 2016, "An Introduction to Icon Animation Techniques", https://www.androiddesignpatterns.com/2016/11/introduction-to-icon-animation-techniques.html), https://www.androiddesignpatterns.com/2016/11/introduction-to-icon-animation-techniques.html

For pathData syntax see (Bentley, 2018. Html5-PoloyglotMarkup-Reference.docx, file://Atlas/Users/John/Documents/Sda/Info/Web/KB/Xhtml/Reference/Html5-PolyglotMarkup-Reference.docx), SVG > Synatx.

Tools

Use Android Studio's preview window when working on an xml vector drawable directly (that is, when tweaking the pathData manually).

Use Android Studio's Vector Asset Studio to convert and import as a VectorDrawable: a SVG (and PSD) file; or a Material Design icon.

- Android Studio > Project Pane > app > res > drawable > [Right click]
- New > Vector Asset. This open's (Vector) Asset Studio.

- Either:
  - Import a local file (SVG, PSD); or
  - Import a material design vector icon. Clip Art > Click on "Clip Art" Button.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Images & Graphics, Vector drawables, https://developer.android.com/guide/topics/graphics/vector-drawable-resources

(Google, n.d., "Material Design", https://material.io/), Icons, https://material.io/resources/icons/

Create SVG files in:

- InkScape.

Animations and Transitions

Animated Vector Drawables

Overview

See also Resources > Resource Tyes > File Resources > Animation, above.

Versioning:

- If Min SDK Version >= Api 21. Native support. Supports both ImageView and TextView.
- If Min SDK Version < Api 21 (and >= API 11). Need support library. Supports only ImageView.

> As of Android Support Library 23.3.0, support vector drawables can only be loaded via app:srcCompat or setImageResource().. [i.e. Only ImageView]

(Google, n.d., "Android Developers Blog", https://android-developers.googleblog.com/), 2016-02-24 "Android Support Library 23.2", https://android-developers.googleblog.com/2016/02/android-support-library-232.html

Min SDK Version >= 21

Setting up the XML drawables and animators

Animating VectorDrawables entails defining three parts:

1. An image in a `VectorDrawable` (e.g. in an XML file).
2. One or more property animations that defines the animations (using `Animator` and extensions `ValueAnimator`, `ObjectAnimator`, or `AnimatorSet`). E.g. In `res/animator/filename.xml`
3. An `AnimatedVectorDrawable` that glues together the VectorDrawable with the property animation(s).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Images & Graphics, Vector drawables overview, About AnimatedVectorDrawable class, https://developer.android.com/guide/topics/graphics/vector-drawable-resources#animated-vector-drawable-class

The three parts to animating VectorDrawables can be done in multiple files or a single file.

Animating VectorDrawables with multiple files:

1. Define a VectorDrawable XML file.

```
<!-- // res/drawable/vector_drawable.xml -->
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:height="64dp"
    android:width="64dp"
    android:viewportHeight="600"
    android:viewportWidth="600" >
    <group
        android:name="rotationGroup"
        android:pivotX="300.0"
        android:pivotY="300.0"
        android:rotation="45.0" >
        <path
            android:name="vectorPath"
            android:fillColor="#000000"
            android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70z" />
    </group>
</vector>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\drawable\vector_drawable.xml
(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Images & Graphics, Vector drawables overview, About AnimatedVectorDrawable class, https://developer.android.com/guide/topics/graphics/vector-drawable-resources#animated-vector-drawable-class

2. Define one or more property animations.

```
// res/animator/rotation.xml
<?xml version="1.0" encoding="utf-8"?>
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="6000"
    android:propertyName="rotation"
    android:valueFrom="0"
    android:valueTo="360"
    android:interpolator="@android:anim/linear_interpolator" />

// res/animator/path_morph.xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        android:duration="3000"
        android:propertyName="pathData"
        android:valueFrom="M300,70 l 0,-70 70,70 0,0   -70,70z"
        android:valueTo="  M300,70 l 0,-70 70,0  0,140 -70,0 z"
        android:valueType="pathType" />
</set>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Images & Graphics, Vector drawables overview, About AnimatedVectorDrawable class, Multiples XML files, https://developer.android.com/guide/topics/graphics/vector-drawable-resources#multiple-files

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\animator\rotation.xml
res/animator/path_morph.xml

3. An `AnimatedVectorDrawable` that glues together the VectorDrawable with the property animations. `android:name` is the name of the target VectorDrawable group, path, or clip-path `android:name`.

```
// res/drawable/animated_vector_drawable.xml
<?xml version="1.0" encoding="utf-8"?>
<animated-vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/vector_drawable">
    <target
        android:name="rotationGroup"
        android:animation="@animator/rotation" />
    <target
        android:name="vectorPath"
        android:animation="@animator/path_morph" />
</animated-vector>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\drawable\animated_vector_drawable.xml

Animating VectorDrawables with a single file. You can combine the vector drawable and the animation(s) into one animated vector drawable.

```
<?xml version="1.0" encoding="utf-8"?>
<animated-vector xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:aapt="http://schemas.android.com/aapt">
    <aapt:attr name="android:drawable">
        <vector
            android:width="64dp"
            android:height="64dp"
            android:viewportHeight="600"
            android:viewportWidth="600">
            <group
                android:name="rotationGroup"
                android:pivotX="300.0"
                android:pivotY="300.0"
                android:rotation="45.0">
                <path
                    android:name="vectorPath"
                    android:fillColor="#000000"
                    android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70z" />
            </group>
        </vector>
    </aapt:attr>
    <target android:name="rotationGroup">
        <aapt:attr name="android:animation">
            <objectAnimator
                android:duration="6000"
                android:propertyName="rotation"
                android:valueFrom="0"
                android:valueTo="360" />
        </aapt:attr>
    </target>
    <target android:name="vectorPath">
        <aapt:attr name="android:animation">
            <objectAnimator
                android:duration="3000"
                android:propertyName="pathData"
                android:valueFrom="M300,70 l 0,-70 70,70 0,0   -70,70z"
                android:valueTo="  M300,70 l 0,-70 70,0  0,140 -70,0 z"
                android:valueType="pathType" />
        </aapt:attr>
    </target>
</animated-vector>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\drawable\animated_vector_drawable_single_file.xml

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Images & Graphics, Vector drawables overview, About AnimatedVectorDrawable class, Single XML file, https://developer.android.com/guide/topics/graphics/vector-drawable-resources#single-file

Apart from the above to switch from a multiple file AnimatedVectorDrawable to a SingleVectorDrawable only requires changing the references in your /layout/*.xml file to point to the relevant file. See "Hooking up the animation" below.

```
<!-- In layout/activity_vector_drawable_demo.xml change to -->

android:drawableBottom="@drawable/animated_vector_drawable_single_file"

...

android:src="@drawable/animated_vector_drawable_single_file"
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_vector_drawable_demo.xml

Hooking up the animation

Using an Animated Vector Drawable from the UI (Where min SDK version >= 21).

1. Create a layout file with ImageView(s) and/or TextViews(s).
    a. TextViews hookup to animated_vector_drawable.xml with the `android:drawable*` attribute.
    b. ImageViews hookup to animated_vector_drawable.xml with the `android:src` attribute.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
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"
    tools:context=".VectorDrawableDemoActivity">

    <TextView
        android:id="@+id/textView_output"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="32dp"
        android:layout_marginTop="8dp"
        android:text="TextView Output"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.158"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.104"
        app:layout_constraintVertical_chainStyle="spread" />

    <TextView
        android:id="@+id/textView_target"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginStart="32dp"
        android:drawableBottom="@drawable/animated_vector_drawable"
        android:text="TextView Target"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView_output"
        app:layout_constraintVertical_bias="0.141" />

    <ImageView
        android:id="@+id/imageView_target"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="#66ac2653"
        android:contentDescription="A vector drawable"
```

```
        android:src="@drawable/animated_vector_drawable"
        android:visibility="visible"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="@+id/textView_target"
        app:layout_constraintStart_toStartOf="@+id/textView_target"
        app:layout_constraintTop_toBottomOf="@+id/textView_output"
        app:layout_constraintVertical_bias="0.511" />

    <Button
        android:id="@+id/button_animate_textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:text="Animate TextView"
        app:layout_constraintBottom_toBottomOf="@+id/textView_target"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/textView_target"
        app:layout_constraintTop_toTopOf="@+id/textView_target"
        app:layout_constraintVertical_bias="0.526" />

    <Button
        android:id="@+id/button_animate_imageView"
        android:layout width="wrap content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:text="Animate ImageView"
        app:layout constraintBottom toBottomOf="@+id/imageView target"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.76"
        app:layout_constraintStart_toEndOf="@+id/imageView_target"
        app:layout_constraintTop_toTopOf="@+id/imageView_target" />

</android.support.constraint.ConstraintLayout>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_vector_drawable_demo.xml

2. In your Activity start the animations as follows.

```
public class VectorDrawableDemoActivity extends Activity implements
View.OnClickListener {

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

        Button buttonAnimateTextView = findViewById(R.id.button_animate_textView);
        buttonAnimateTextView.setOnClickListener(this);

        Button buttonAnimateImageView = findViewById(R.id.button_animate_imageView);
        buttonAnimateImageView.setOnClickListener(this);
    }

    // A central onClick listener
    @Override
    public void onClick(View v) {
        try {
            switch (v.getId()) {
                case R.id.button_animate_textView:
                    startAnimation(findViewById(R.id.textView_target));
                    break;
                case R.id.button_animate_imageView:
                    startAnimation(findViewById(R.id.imageView_target));
                    break;
                default:
                    throw new Exception("Unsupported view type");
            }
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }

    public void startAnimation(View view) throws Exception {
        TextView output = findViewById(R.id.textView_output);
        output.setText(au.com.softmake.sjlmodule.DateTimeHelper.getNowString());

        if (view instanceof ImageView) {
            ImageView passedImageView = (ImageView) view;
            Drawable drawable = passedImageView.getDrawable();
            if (drawable instanceof Animatable) {
                ((Animatable) drawable).start();
            }
        } else if (view instanceof TextView) {
            TextView passedTextView = (TextView) view;
            for (Drawable drawableLoop : passedTextView.getCompoundDrawables()) {
                if (drawableLoop instanceof Animatable) {
                    ((Animatable) drawableLoop).start();
                }
            }
        } else {
            throw new Exception("Unsupported View type.");
        }
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\VectorDrawableDemoActivity.java

Min SDK Version < 21

To support AnimatedVectorDrawables below Min SDK version 21 (and above Min SDK version 11) do the following:

1. Observe that your Min SDK Version for your app and any android library is less that 21 and greater than 11. File > Project Settings ... > Modules > [Click on each module in turn] > Flavors > [config, e.g. defaultConfig] > Min Sdk Version.
2. build.gradle (Module.app). Set the `vectorDrawables.useSupportLibrary` flag. Reference the `AppCompat` library.

```
android {
    defaultConfig {
        vectorDrawables.useSupportLibrary = true
    }
}

dependencies {
    implementation 'com.android.support:appcompat-v7:27.1.1'
    // You don't need the following
    implementation 'com.android.support:support-vector-drawable:27.1.1'
    implementation 'com.android.support:animated-vector-drawable:27.1.1'
}
```

3. Layout XML code. Use ImageView only (not TextView). Use `app:scrCompat` instead of `android:src`.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
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"
    tools:context=".VectorDrawableDemoCompatActivity">

    <TextView
        android:id="@+id/textView_output"
```

```
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="32dp"
        android:layout_marginTop="8dp"
        android:text="TextView Output"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.158"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.104"
        app:layout_constraintVertical_chainStyle="spread"
        android:layout_marginRight="8dp"
        android:layout_marginLeft="32dp" />

    <ImageView
        android:id="@+id/imageView_target"
        android:layout_width="74dp"
        android:layout_height="72dp"
        android:layout_marginBottom="8dp"
        android:layout_marginTop="8dp"
        android:background="#66ac2653"
        android:contentDescription="A vector drawable"
        android:visibility="visible"
        app:srcCompat="@drawable/animated_vector_drawable"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="@+id/textView_output"
        app:layout_constraintTop_toBottomOf="@+id/textView_output"
        app:layout_constraintVertical_bias="0.184"
        />

    <Button
        android:id="@+id/button_animate_imageView"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:text="Animate"
        app:layout_constraintBottom_toBottomOf="@+id/imageView_target"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.146"
        app:layout_constraintStart_toEndOf="@+id/imageView_target"
        app:layout_constraintTop_toTopOf="@+id/imageView_target"
        app:layout_constraintVertical_bias="0.0" />

</android.support.constraint.ConstraintLayout>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_vector_drawable_compat_demo.xml

4. Make your activity use the `AppCompatActivity` class rather than the Activity class.

```
public class VectorDrawableDemoCompatActivity extends AppCompatActivity implements
View.OnClickListener {

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

        Button buttonAnimateImageView = findViewById(R.id.button_animate_imageView);
        buttonAnimateImageView.setOnClickListener(this);
    }

    // A central onClick listener
    @Override
    public void onClick(View v) {
        try {
            switch (v.getId()) {
```

```
                case R.id.button_animate_imageView:
                    startAnimation(findViewById(R.id.imageView_target));
                    break;
                default:
                    throw new Exception("Unsupported view type");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void startAnimation(View view) throws Exception {
        TextView output = findViewById(R.id.textView_output);
        output.setText(au.com.softmake.sjlmodule.DateTimeHelper.getNowString());

        if (view instanceof ImageView) {
            ImageView passedImageView = (ImageView) view;
            Drawable drawable = passedImageView.getDrawable();
            if (drawable instanceof Animatable) {
                ((Animatable) drawable).start();
            }
        } else {
            throw new Exception("Unsupported View type.");
        }
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\VectorDrawableDemoCompatActivity.java

5. Optionally you can use a single file AnimatedVectorDrawable (as above).

```
    <ImageView
        ...
        app:srcCompat="@drawable/animated vector drawable single file"
        ...
        />
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\layout\activity_vector_drawable_compat_demo.xml (Lockwood, 2016. An Introduction to Icon Animation Techniques)

Techniques

Several examples

For several vector drawable animation techniques see ...

(Lockwood 2016, "An Introduction to Icon Animation Techniques", https://www.androiddesignpatterns.com/2016/11/introduction-to-icon-animation-techniques.html), https://www.androiddesignpatterns.com/2016/11/introduction-to-icon-animation-techniques.html#trimming-stroked-paths

Clock Example (Min SDK Version >= 21)

Let's animate a red circle acting as a clock...

Derived from (Lockwood 2016, "An Introduction to Icon Animation Techniques", https://www.androiddesignpatterns.com/2016/11/introduction-to-icon-animation-techniques.html), https://www.androiddesignpatterns.com/2016/11/introduction-to-icon-animation-techniques.html#trimming-stroked-paths

1. The Vector Drawable.

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

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="256dp"
    android:height="256dp"
    android:viewportHeight="300"
    android:viewportWidth="300">
    <group android:name="redCircle">
        <path
            android:name="vectorPath"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:strokeColor="@android:color/holo_red_dark"
            android:strokeWidth="15"
            android:pathData="m7.82411,149.66667c0,-78.91855 63.92403,-142.84258
142.84258,-142.84258c78.91855,0 142.84258,63.92403 142.84258,142.84258c0,78.91855 -
63.92403,142.84258 -142.84258,142.84258c-78.91855,0 -142.84258,-63.92403 -142.84258,-
142.84258z" />
    </group>
</vector>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\drawable\vector_drawable_circle.xml

2. The property animation.

```
<?xml version="1.0" encoding="utf-8"?>
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="60000"
    android:propertyName="trimPathEnd"
    android:valueFrom="0"
    android:valueTo="1"
    android:interpolator="@android:anim/linear_interpolator" />
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\animator\clock_tick.xml

3. The AnimatedVectorDrawable, which glues the vector drawable to the property animator.

```
<?xml version="1.0" encoding="utf-8"?>
<animated-vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/vector_drawable_circle">
    <target
        android:name="vectorPath"
        android:animation="@animator/clock_tick" />
</animated-vector>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\drawable\animated_vector_drawable_red_circle_clock_tick.xml

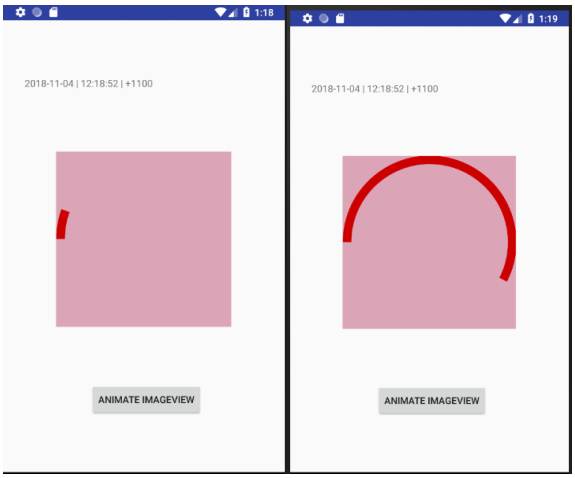
4. Hook it up and start it, in code.

```
package au.com.softmake.mythirdapp.vectordrawabledemo;

import android.app.Activity;
import android.graphics.drawable.Animatable;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import au.com.softmake.mythirdapp.R;

public class RedCircleAnimationActivity extends Activity implements
View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_red_circle_animation);

        Button buttonAnimateImageView = findViewById(R.id.button_animate_imageView);
        buttonAnimateImageView.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        try {
            switch (v.getId()) {
                case R.id.button_animate_imageView:
                    startAnimation(findViewById(R.id.imageView_target));
                    break;
                default:
                    throw new Exception("Unsupported view type");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

    public void startAnimation(View view) throws Exception {
        TextView output = findViewById(R.id.textView_output);
        output.setText(au.com.softmake.sjlmodule.DateTimeHelper.getNowString());

        if (view instanceof ImageView) {
            ImageView passedImageView = (ImageView) view;
            Drawable drawable = passedImageView.getDrawable();
            if (drawable instanceof Animatable) {
                ((Animatable) drawable).start();
            }
        } else if (view instanceof TextView) {
            TextView passedTextView = (TextView) view;
            for (Drawable drawableLoop : passedTextView.getCompoundDrawables()) {
                if (drawableLoop instanceof Animatable) {
                    ((Animatable) drawableLoop).start();
                }
            }
        } else {
            throw new Exception("Unsupported View type.");
        }
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\vectordrawabledemo\RedCircleAnimationActivity.java

5. Result.



Tools

App Widgets

Basics

Widgets (aka "App Widgets") come in two basic forms: plain (john bentley's terminology); and backed with a Collection.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, https://developer.android.com/guide/topics/appwidgets

The necessary ingredients for a plain widget:

- App Widget layout file. An initial layout for the widget, defined in XML. In res/layout/.
- AppWidgetProviderInfo xml. Defines metadata about the Widget. In res/xml/.
- AppWidgetProvider class. Programmatically interact with the app widget.
- AndroidManifest.xml declaration. This references the AppWidgetProvider class and AppWidgetProviderInfo xml.
- Obtaining and setting a preview image.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, https://developer.android.com/guide/topics/appwidgets

The optional ingredients for a plain widget:

- Use a 9 patch drawable as a background.
- Enabling on the lockscreen.
- Providing an update button.
- Widget Configuration Activity. Allows the user to configure the widget at create-time.
- Changing the update interval in code.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, https://developer.android.com/guide/topics/appwidgets

The following sections contain a numbered procedure for creating an app widget ...

Necessities

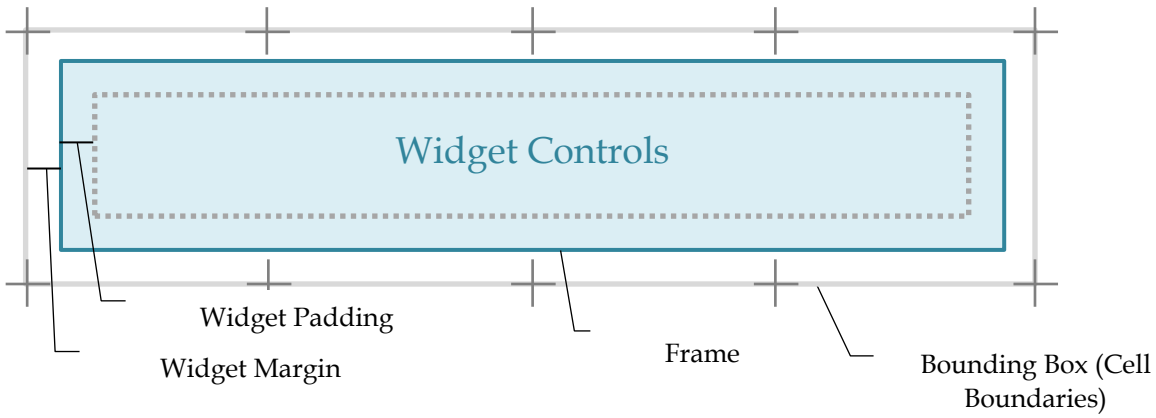
App Widget Layout File

An app widget layout file supports a subset of layouts and UI elements.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, Creating the App Widget Layout, https://developer.android.com/guide/topics/appwidgets#CreatingLayout

Margins and padding

The margins and padding for a widget on a home screen should look like this ...



(Google 2020, "App Widget Design Guidelines", https://developer.android.com/guide/practices/ui_guidelines/widget_design), Standard Widget Anatomy, https://developer.android.com/guide/practices/ui_guidelines/widget_design#anatomy

- App Widget layout File.
- Create App Widget layout File. E.g. res/layout/my_widget.xml
- Set widget padding and margins. To set widget padding and margins we use android:layout_margin and android:padding of the outermost containing layout (of the Widget Layout File).

```
<!-- res/layout/my widget.xml (The Widget Layout file) -->
<?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:layout_margin="@dimen/widget_margin"
    android:padding="@dimen/widget_padding"
    android:background="#ffffb27e"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView_output"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_gravity="top"
        android:layout_weight="1"
        android:background="#ff00beff"
        android:text="@string/overwritten" />

    <Button
        android:id="@+id/button_update"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
```

```
        android:text="Update" />
</LinearLayout>
```

In http://developer.android.com/guide/topics/appwidgets/index.html#CreatingLayout, "Adding margins to App Widgets" the author hasn't taken enough drugs in the 60's and suggests, wrongly, you should further embed this outter most layout (here LinearLayout) in a FrameLayout and use the **padding** attribute for **widget margins**. This is done to handle differences between how the api levels handle margins before and after API 14/Android 4.0. While handling these differences is important handling it in this way seem uncessarily convulated and confusing.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, Creating the App Widget Layout, https://developer.android.com/guide/topics/appwidgets#CreatingLayout

Handle App Widget margins for different platform verions. From Android 4.0 (Api 14) App widget margins (between frame and bounding box) are automatically inserted. Therefore you need to add margins to versions less than API 14, and set zero margins for version API 14 and above.

```
<!-- res/values/dimens.xml -->
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="widget_margin">8dp</dimen>
    <dimen name="widget_padding">5dp</dimen>
</resources>

<!-- res/values-v14/dimens.xml -->
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="widget_margin">0dp</dimen>
</resources>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, Creating the App Widget Layout, https://developer.android.com/guide/topics/appwidgets#CreatingLayout

AppWidgetProviderInfo xml

Specify initial layout size

The initial layout size, the size of the widget when the user first drops it onto a home screen, is determined by the `minWidth` and `minHeight` properties of the AppWidgetProviderInfo xml, the device resolution, and how many cells offered by the home screen (which can vary according to user settings). The rule: the widget will be stretched to occupy the minimum number of cells, horizontally and vertically, specified by `minWidth` and `minHeight`.

(Google 2020, "App Widget Design Guidelines", https://developer.android.com/guide/practices/ui_guidelines/widget_design), Standard Widget Anatomy, https://developer.android.com/guide/practices/ui_guidelines/widget_design#anatomy

Calculating the `minWidth` and `minHeight` can be done roughly or exactly.

A rough calculation of `minWidth` and `minHeight` can be done as follows.

# of Cells (Columns or Rows)	**Available Size (dp) (minWidth or minHeight)**
1	40dp
2	110dp

# of Cells (Columns or Rows)	Available Size (dp) (minWidth or minHeight)
3	180dp
4	250dp
…	…
n	70 × n − 30

(Google 2020, "App Widget Design Guidelines", https://developer.android.com/guide/practices/ui_guidelines/widget_design), Standard Widget Anatomy, https://developer.android.com/guide/practices/ui_guidelines/widget_design#anatomy

An exact calculation can be done as follows [TODO]

AppWidgetProviderInfo xml.

Create the App Widget Provider info File. E.g. /res/xml/my_widget_provider_info.xml

Specify the initial layout size.

```
<!-- res/xml/my_widget_provider_info.xml -->
<?xml version="1.0" encoding="utf-8"?>
<!--  updatePeriodMillis: Updates requested with updatePeriodMillis will not be delivered
      more than once every 30 minutes.
      1800000 milliseconds = 30 minutes
      (1000 milliseconds * 60 Seconds * 60 Minutes) * 30
      = 1 minute in milliseconds * 30 minutes -->
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:initialLayout="@layout/my_widget"
    android:minHeight="40dp"
    android:minWidth="250dp"
    android:resizeMode="horizontal|vertical"
    android:updatePeriodMillis="1800000"
    android:widgetCategory="home_screen"
    />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, Adding the AppWidgetProviderInfo Metadata, https://developer.android.com/guide/topics/appwidgets#MetaData

AppWidgetProvider class

Extend AppWidgetProvider (which in turn extends BroadcastReceiver) and override at least onUpdate().

```
public class MyWidget extends AppWidgetProvider {

    private static String getRandomMessage(Boolean loremOverride) {
        ...
    }

    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
                         int[] appWidgetIds) {
        super.onUpdate(context, appWidgetManager, appWidgetIds);

        String message = String.format("%s: %s", getNowString(), getRandomMessage(true));
        // Loop for every App Widget that belongs to this provider.
        // That is a user might have multiple instances of the same widget.
        for (int appWidgetID : appWidgetIds) {
            RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
                                                      R.layout.my_widget);
            remoteViews.setTextViewText(R.id.textView_output, message);
```

```
            appWidgetManager.updateAppWidget(appWidgetID, remoteViews);

        }
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, Using the AppWidgetProvider Class, https://developer.android.com/guide/topics/appwidgets#AppWidgetProvider

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\MyWidget.java

onUpdate() is called to update the App Widget at intervals defined by the updatePeriodMillis attribute in the AppWidgetProviderInfo xml.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, Using the AppWidgetProvider Class, https://developer.android.com/guide/topics/appwidgets#AppWidgetProvider

AndroidManifest.xml declaration

Declare the AppWidgetProvider class and AppWidgetProviderInfo xml in AndroidManifest.xml like this ...

```
<receiver android:name=".MyWidget">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>
    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/my_widget_provider_info" />
</receiver>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, Declaring an App Widget in the Manifest, https://developer.android.com/guide/topics/appwidgets/#Manifest

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\AndroidManifest.xml

Obtaing and setting a preview image

To obtain an App Widget preview image:

- In an Emulator running Android 3.0 (API 11) or later (it might be better to use a later emulator running a high resolution device) run the app "Widget Preview" (in some emulators you might have to scroll to the second screen in the app draw to see "Widget Previw"). Click "Take a Snapshot"
- In Android Studio > Tools > Android > Android Device Monitor...
- Pull storage/sdcard/download/My_Widget_ori_portrait.png to your local desktop file system.
- Rename the file "my_widget_preview.png".
- Move to your project's res/drawable folder.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, Setting a Preview Image, https://developer.android.com/guide/topics/appwidgets/#preview

Set a preview image in res/xml/my_widget_provider_info.xml (works for Android 3.0, API 11 and higher) ...

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:initialLayout="@layout/my_widget"
    android:minHeight="40dp"
    android:minWidth="250dp"
    android:previewImage="@drawable/my_widget_preview"
    android:resizeMode="horizontal|vertical"
    android:updatePeriodMillis="1800000"
    android:widgetCategory="home_screen"
    />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, Setting a Preview Image, https://developer.android.com/guide/topics/appwidgets/#preview

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\xml\my_widget_provider_info.xml

Optionals

Use a 9 Patch drawable as a Backround

See, first, Nine-Patch Drawable

To Use a 9 Patch drawable as a backround for your App widget do this:

- When creating your 9 patches don't build in different margins for different API versions. We handle different margins as above: Margins and padding
- Copy your 9 patches for each resolution into the relevant folders. Res/drawable-mdpi, res/drawable-hdpi, etc.
- Optionally adjust the 9 patch border for your desired stretching and text containing behaviour by double clicking each 9 patch.
- Set background like this

```
<!-- res/layout/my_widget.xml (The Widget Layout file) -->
<?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:layout_margin="@dimen/widget_margin"
    android:padding="@dimen/widget_padding"
    android:background="@drawable/appwidget_dark_bg"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView_output"
...
```

Enabling on the lockscreen

App Widgets can be enabled on the lockscreen from Android 4.2 (API 17).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, https://developer.android.com/guide/topics/appwidgets/

Procedure:

- In res\xml\my_widget_provider_info.xml designate it is enabled on the lockscreen with "keyguard".
- In res\xml\my_widget_provider_info.xml provide and initial sizing (which may be different from the sizing for regular widget hosts)

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:configure="au.com.softmake.mysecondapp.widgets.MyWidgetConfigurationActivity"
    android:widgetCategory="home screen|keyguard"
    android:initialLayout="@layout/my_widget"
    android:initialKeyguardLayout="@layout/my_widget_lockscreen"
...
    />
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, https://developer.android.com/guide/topics/appwidgets/

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\xml\my_widget_provider_info.xml

- Create the layout\my_widget_lockscreen.xml layout (which will be based on your regular my_widget.xml with some small variations).
- Return the right layout in code (in the AppWidgetProvider Class).

```
public class MyWidget extends AppWidgetProvider {
...
/**
 * Usage Example:
 * <pre>
 *  int layoutReference = 0;
 *  try {
 *     layoutReference = getHomeOrLockScreenLayout(appWidgetManager,
 *                                     appWidgetId,
 *                                     R.layout.my_widget,
 *                                     R.layout.my_widget_lockscreen);
 *  } catch (Exception e) {
 *          e.printStackTrace();
 *  }
 *  RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
 *                                             layoutReference);
 * </pre>
 * @param appWidgetManager
 * @param appWidgetId
 * @param homeScreenLayout E.g. R.layout.my_widget
 * @param lockScreenLayout E.g. R.layout.my_widget_lockscreen
 * @return An int reference to the home screen or lock screen layout.
 * @throws Exception
 */
private static int getHomeOrLockScreenLayout(AppWidgetManager appWidgetManager,
                                             int appWidgetId, int homeScreenLayout,
                                             int lockScreenLayout) throws Exception {

    int result = -1;
    if (Build.VERSION.SDK_INT >= 17) {
        Bundle optionsBundle = appWidgetManager.getAppWidgetOptions(appWidgetId);

        int category = optionsBundle.getInt(AppWidgetManager
                                                .OPTION_APPWIDGET_HOST_CATEGORY,
                                            -1);

        switch (category) {
            case AppWidgetProviderInfo.WIDGET_CATEGORY_HOME_SCREEN:
                result = homeScreenLayout;
                break;
```

```
            case AppWidgetProviderInfo.WIDGET_CATEGORY_KEYGUARD:
                result = lockScreenLayout;
                break;
            default:
                throw new Exception("Unexpected switch case");
        }
    } else {
        result = homeScreenLayout;
    }

    return result;
}

public static void updateAppWidget(Context context, AppWidgetManager appWidgetManager,
                                   int appWidgetId, String message) {
    int layoutReference = 0;
    try {
        layoutReference = getHomeOrLockScreenLayout(appWidgetManager,
                                                    appWidgetId,
                                                    R.layout.my_widget,
                                                    R.layout.my_widget_lockscreen);
    } catch (Exception e) {
        e.printStackTrace();
    }
    RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
                                              layoutReference);
...
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, https://developer.android.com/guide/topics/appwidgets/

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\widgets\MyWidget.java

Sizing notes about widgets on the locksreen:

- minWidth, minHeight, minResizeWidth, and minResizeHeight are ingored (but you still will need them as they are used on home screens).
- The width always fills the space provided.

For widget height you can only mark it verticially resizable (recommended) or not (e.g. "android:resizeMode="vertical")

If vertically resizeable:

- In portrait on phones height is "small".
- Otherwise fills available height.
- If not vertically resizeable then height will always be "small":
- In portrait on phones, "small" means the space remaining apart form the unlock UI.
- Otherwise (tablets and landscape on phones), "small" is set on a device basis.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, https://developer.android.com/guide/topics/appwidgets/

Providing an update button

To provide an update button (the following supercedes my Stackoverflow entry):

- Create a custom action string.
- Create getPendingSelfIntent()
- Create a method, updateAppWidget(Context context, AppWidgetManager appWidgetManager, int appWidgetId, String message) which:
- Works on AppWidget instances;
- Registers a click handler with the update button, using getPendingSelfIntent() and your custom action constant.
- Takes any custom information to update as a parameter; and
- Is public and static so that in can be optionally called by an external Activity like WidgetConfigurationActivity.
- In onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) call updateAppWidget for each instance, with your custom update values.
- Create onUpdate(Context context) a general technique for calling the overridden onUpdate method, requiring only the context parameter.
- In onReceive() fire onUpdate(Context context) when you receive the custom action.

```
public class MyWidget extends AppWidgetProvider {

    // Update Button 1. Create a custom action String.
    private static final String ACTION_UPDATE_CLICK = "au.com.softmake" +
            ".mysecondapp" + ".action.UPDATE_CLICK";

    private static int mCount = 0;

    private static String getMessage() {
        return String.valueOf(mCount++);
    }

    // Update Button 2. Create getPendingSelfIntent().
    private static PendingIntent getPendingSelfIntent(Context context, String action) {
        // An explicit intent directed at the current class (the "self").
        Intent intent = new Intent(context, MyWidget.class);
        intent.setAction(action);
        return PendingIntent.getBroadcast(context, 0, intent, 0);
    }

    /**
     * Update Button 3. Create a method, updateAppWidget(Context context,
     * AppWidgetManager appWidgetManager, int appWidgetId, String message) which:
     * <ol>
     * <li>Works on AppWidget instances;</li>
     * <li>Registers a click handler with the update button,
     * using getPendingSelfIntent()
     * and your custom action constant.</li>
     * <li>Takes any custom information to update as a parameter;</li>
     * <li>Is public and static so that in can be optionally called by an external
     * Activity like WidgetConfigurationActivity.</li>
     * </ol>
     *
     * @param message the message to display on the widget
     */
    public static void updateAppWidget(Context context, AppWidgetManager appWidgetManager,
                                       int appWidgetId, String message) {
        RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
                                                  R.layout.my_widget);

        remoteViews.setTextViewText(R.id.textView_output, message);
        remoteViews.setOnClickPendingIntent(R.id.button_update,
                                            getPendingSelfIntent(context,
                                                                 ACTION_UPDATE_CLICK));

        appWidgetManager.updateAppWidget(appWidgetId, remoteViews);

    }
```

```
    /**
     * Update Button 4. In onUpdate(Context context,
     * AppWidgetManager appWidgetManager,
     * int[] appWidgetIds) call updateAppWidget for each instance,
     * with your custom update values.
     */
    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
                         int[] appWidgetIds) {
        super.onUpdate(context, appWidgetManager, appWidgetIds);

        String message = getMessage();
        //        String message = ZaraData.getMessage(context);

        // Loop for every App Widget instance that belongs to this provider. Noting,
        // that is, a user might have multiple instances of the same widget on
        // their home
        // screen.
        for (int appWidgetID : appWidgetIds) {
            updateAppWidget(context, appWidgetManager, appWidgetID, message);
        }
    }

    /**
     * Update Button 5. Create onUpdate(Context context) a general technique for
     * calling
     * the
     * onUpdate method, requiring only the context parameter.
     *
     * @see <a href="http://android-er.blogspot.com
     * .au/2010/10/update-widget-in-onreceive-method.html">
     * Android-er > 2010-10-19 > Update Widget in onReceive() method</a>
     */
    public void onUpdate(Context context) {
        AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(context);

        // Uses getClass().getName() rather than MyWidget.class.getName() for
        // portability into any App Widget Provider Class
        ComponentName thisAppWidgetComponentName =
                new ComponentName(context.getPackageName(), getClass().getName ());
        int[] appWidgetIds = appWidgetManager.getAppWidgetIds(thisAppWidgetComponentName);
        onUpdate(context, appWidgetManager, appWidgetIds);
    }

    /**
     * Update Button 6. In onReceive() fire onUpdate(Context context) when you
     * receive
     * the
     */
    @Override
    public void onReceive(Context context, Intent intent) {
        super.onReceive(context, intent);

        if (ACTION_UPDATE_CLICK.equals(intent.getAction())) {
            onUpdate(context);
        }
    }
}
```

(Stackoverflow 2018, "Stackoverflow", https://stackoverflow.com/), http://stackoverflow.com/questions/14798073/button-click-event-for-android-widget, Search for "John Bentley"

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\MyWidget.java

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, Using the AppWidgetProvider Class, https://developer.android.com/guide/topics/appwidgets#AppWidgetProvider

App Widget Configuration Activity

The procedure for creating a widget configuration activity:

- Ensure your app widget provider class has a public static updateAppWidget. This will allow us to update the widget from the configuration activity. See Providing an update button.
- Create the app widget configuration activity class.

```
public class MyWidgetConfigurationActivity extends ActionBarActivity {
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\widgets\MyWidgetConfigurationActivity.java

- In AndroidManifest.xml declare your app widget configuration activity, with an APPWIDGET_CONFIGURE intent filter.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="au.com.softmake.mysecondapp" >
<application>
    ....
    <activity
        android:name=".widgets.MyWidgetConfigurationActivity"
        android:label="@string/title_activity_my_widget_configuration_activitiy">
        <intent-filter>
            <action android:name="android.appwidget.action.APPWIDGET_CONFIGURE" />
        </intent-filter>
    </activity>
</application>
```

You don't need any parent activity (if the App Widget Configuration isn't accessible from the settings menu).

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\AndroidManifest.xml

- In your res/xml/app_widget_provide_info.xml and a reference to your configuration activity, using "android:configure".

```
<!-- \res\xml\my_widget_provider_info.xml -->
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:configure="au.com.softmake.mysecondapp.widgets.MyWidgetConfigurationActivity"
    ...
    />
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\xml\my_widget_provider_info.xml

- In your app widget configuration activity class, onCreate().
- setResult(RESULT_CANCELED) so that the host will cancel widget placement if the user presses the back button.
- Get widget id from the intent that launched the activity.
- If they gave us an intent without the widget id, just bail.

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

        // This will cause the widget host to cancel out of the widget placement if they
        // press the back button.
        setResult(RESULT_CANCELED);

        setContentView(R.layout.activity_my_widget_configuration);
```

```
        // Get widget id from the intent that launched the activity.
        Intent intent = getIntent();
        Bundle extras = intent.getExtras();
        if (extras != null) {
            mAppWidgetId = extras.getInt(AppWidgetManager.EXTRA_APPWIDGET_ID,
                                         AppWidgetManager.INVALID_APPWIDGET_ID);
        }

        // If they gave us an intent without the widget id, just bail.
        if (mAppWidgetId == AppWidgetManager.INVALID_APPWIDGET_ID) {
            finish();
        }
    }
```

- In your app widget configuration activity class, onSaveClick().
- Save preferences.
- Do App widget configuration.
- Update the App Widget
- Create a return intent with result; set result.
- finish();

```
    public void saveOnClick(View view) {

        // TODO: Save preferences

        // Do App Widget Configuration
        String message = "Message set by configuration";

        // Update the App Widget
        AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(this);
        MyWidget.updateAppWidget(this, appWidgetManager, mAppWidgetId, message);

        // Create a return intent with result; set result.
        Intent resultIntent = new Intent();
        resultIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, mAppWidgetId);
        setResult(RESULT_OK, resultIntent);

        finish();
    }
```

- In your app widget configuration activity class, onCancelClick(), finish()

```
    public void cancelOnClick(View view) {
        finish();
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, App Widgets, Build an App Widget, Creating an App Widget Configuration Activity, https://developer.android.com/guide/topics/appwidgets/#Configuring

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\widgets\MyWidgetConfigurationActivity.java

Changing the update interval in Code

This is convoluted but can be done via an Alarm Manager and a Service. This is the advantage of: setting an interval less than 30 minutes (not recommended); letting the user configure the interval; and making the update intervals exact.

See http://www.parallelrealities.co.uk/2011/09/using-alarmmanager-for-updating-android.html

Generally it is probably simpler just to set the widget update interval in xml.

App Widgets backed by collections

Permissions

System Permissions

Basic Architecture

Applications statically declare the permissions they require, and the Android system prompts the user for consent at the time the application is installed.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Permissions, https://developer.android.com/guide/topics/permissions/overview

Review the permissions your application requires from Android Device > Settings > Apps > [Click on your app] ....

Application Signing

All APKs (`.apk` files) must be signed with a certificate whose private key is held by their developer. This certificate identifies the author of the application. The certificate does **not** need to be signed by a certificate authority; it is perfectly allowable, and typical, for Android applications to use self-signed certificates.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Permissions, https://developer.android.com/guide/topics/permissions/overview

Share data

Share security between applications that you create by:

- Signing the two apps with the same signature; and
- Assign the same id with `sharedUserId` in `AndroidManifest.xml`

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Permissions, https://developer.android.com/guide/topics/permissions/overview

To make data stored by an application accessible to all apps create a new file with `getSharedPreferences(String, int)`, `openFileOutput(String, int)`, or `openOrCreateDatabase(String, int)`, `SQLiteDatabase.CursorFactory)`, and can use the `MODE_WORLD_READABLE` and/or `MODE_WORLD_WRITEABLE`

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Permissions, https://developer.android.com/guide/topics/permissions/overview

Using Android Permissions

In almost all cases a permission failure will be printed to the system log.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Permissions, https://developer.android.com/guide/topics/permissions/overview

However, in a normal user situation (such as when the app is installed from Google Play Store), an app cannot be installed if the user does not grant the app each of the requested permissions.

So you generally don't need to worry about runtime failures caused by missing permissions because the mere fact that the app is installed at all means that your app has been granted its desired permissions.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Permissions, https://developer.android.com/guide/topics/permissions/overview

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), https://developer.android.com/reference/android/Manifest.permission

Custom Permssions

Define your permission in AndroidManifest.xml as follows

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.me.app.myapp" >
    <permission android:name="com.me.app.myapp.permission.DEADLY_ACTIVITY"
        android:label="@string/permlab_deadlyActivity"
        android:description="@string/permdesc_deadlyActivity"
        android:permissionGroup="android.permission-group.COST_MONEY"
        android:protectionLevel="dangerous" />
    ...
</manifest>
```

permissionGroup is optional but useful. protectionLevel is required.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Permissions, https://developer.android.com/guide/topics/permissions/overview

Set your custom permission as required, in the AndroidManifest.xml as follows.

```
<uses-permission android:name="au.com.softmake.permission.CUSTOM_PERMISSION" />
```

When defining custom permissions you should use `label` and `decription` as these are displayed to the user.

```
    <string name="permlab_callPhone">directly call phone numbers</string>
    <string name="permdesc_callPhone">Allows the application to call
        phone numbers without your intervention. Malicious applications may
        cause unexpected calls on your phone bill. Note that this does not
        allow the application to call emergency numbers.</string>
```

The label should be short, a few words describing the key piece of functionality the permission is protecting. The description should be a couple sentences describing what the permission allows a holder to do. Our convention for the description is two sentences, the first describing the permission, the second warning the user of what bad things can happen if an application is granted the permission.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Permissions, https://developer.android.com/guide/topics/permissions/overview

Enforce permissions

Enforce permissions for the whole application with `<uses-permission>` in AndroidManifest.xml.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.app.myapp" >
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    ...
</manifest>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Permissions, https://developer.android.com/guide/topics/permissions/overview

Enforce permission at the component level with the `android:permission` attribute, in the AndroidManifest.xml

```
<activity
    android:name="au.com.softmake.myfirstapp.DisplayMessageActivity"
    android:label="@string/title_activity_display_message"
    android:parentActivityName="au.com.softmake.myfirstapp.MainActivity"
    android:permission="au.com.softmake.permission.CUSTOM_PERMISSION" >
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="au.com.softmake.myfirstapp.MainActivity" />
</activity>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Permissions, https://developer.android.com/guide/topics/permissions/overview

For custom permissions you need to enforce permissions using one of the two above techniques, even though you've already declared the custom permission in AndroidManifest.xml with `<permission ...>`

```
<manifest . . . >
    <!-- Declare (create) custom permission -->
    <permission android:name="com.example.project.DEBIT_ACCT" . . . />

    <!-- Enforce custom permission, application level -->
    <uses-permission android:name="com.example.project.DEBIT_ACCT" />
    . . .
    <application . . .>

        <!-- Enforce custom permission, component level -->
        <activity android:name="com.example.project.FreneticActivity"
                  android:permission="com.example.project.DEBIT_ACCT"
                  . . . >
            . . .
        </activity>
    </application>
</manifest>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Permissions, https://developer.android.com/guide/topics/permissions/overview

URI Permissions

When an activity is accessing a content provider the receving activity can set `Intent.FLAG_GRANT_READ_URI_PERMISSION` and/or `Intent.FLAG_GRANT_WRITE_URI_PERMISSION`. To support this content providers need to set `android:grantUriPermissions` or `<grant-uri-permissions>` in the AndroidManifest.xml

```
[*fix]
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Permissions, https://developer.android.com/guide/topics/permissions/overview

Intents and Intent Filters

Intent Fundamentals

Intents activate components. Intents optionally carry data to the component. The activation of a component is done:

- Explicitly, activating a specific component, or
- Implicitly, activating a type of component.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App basics, App fundamentals, https://developer.android.com/guide/components/fundamentals

Every Activity is invoked by an Intent, regardless of how the user navigated there.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App basics, Build your first app, Start another activity, Respond to the send button, https://developer.android.com/training/basics/firstapp/starting-activity#ReceiveIntent

Activation can be done from Java as follows:

App Component	Explicit Activation By
Activity	passing an Intent to `startActivity()` or `startActivityForResult()`
Service	passing an Intent to `startService()`. Or you can bind to the service by passing an Intent to `bindService()`.
ContentProvider (deal with ContentResolver)	by calling `query()` on a `ContentResolver`.
Broadcast Receiver	passing an Intent to methods like `sendBroadcast()`, `sendOrderedBroadcast()`, or `sendStickyBroadcast()`.

```
public void sendMessage(View view) {
  ....

          // Activate an app component with the intent, in this case an Activity
          startActivity(intent);
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App basics, App fundamentals, App components, Activitaing components, https://developer.android.com/guide/components/fundamentals#ActivatingComponents

Content providers are handled by Content resolvers, rather than intents.

Intent Parts

Intent Parts Overview

Intents have the following parts:

- Component name
- Action
- Data
- Category
- Extras
- Flags

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Building an intent, https://developer.android.com/guide/components/intents-filters#Building

The first four represent the defining characteristics of the intent and are used to resolve which implicit intent should start.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Building an intent, https://developer.android.com/guide/components/intents-filters#Building

Component name

Component name: if provided the intent is explicit, otherwise implicit.

Always start a service explicitly by supplying the component name to the intent.

Actions

Actions can be standard, as defined in the Intent class or elsewhere in the Android framework (Settings for example).

```
ACTION_VIEW
Use this action in an intent with startActivity() when you have some information that an
activity can show to the user, such as a photo to view in a gallery app, or an address to
view in a map app.

ACTION_SEND
Also known as the "share" intent, you should use this in an intent with startActivity() when
you have some data that the user can share through another app, such as an email app or
social sharing app.
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Building an intent, https://developer.android.com/guide/components/intents-filters#Building

Actions can be custom. In which case you must define an action string constant which includes your applications package name as a prefix.

```
static final String ACTION_TIMETRAVEL = "com.example.action.TIMETRAVEL";
```

Action Java methods: `setComponent()`, `setClass()`, `setClassName()`, or with the Intent constructor.

Data

"Data" is referenced by a URI object and/or MIME type. It is generally important to specify the MIME type in addition to the URI.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Building an intent, https://developer.android.com/guide/components/intents-filters#Building

Data Java Methods: To set the Data URI only use `setData()`. To set the Data MIME type only use `setType()`. Always use `setDataAndType()` when setting both (the other functions blank their companion).

Category

There can be multiple (or no) categories for an intent. See the Intent class for standard categories

```
CATEGORY_BROWSABLE
The target activity allows itself to be started by a web browser to display data referenced
by a link—such as an image or an e-mail message.

CATEGORY_LAUNCHER
The activity is the initial activity of a task and is listed in the system's application
launcher.
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Building an intent, https://developer.android.com/guide/components/intents-filters#Building

Java Category Method: `addCategory()`.

Extras

Extras are key-value pairs that you pack into, and extract from, an intent.

You can use standard Extra keys. See the Intent class for standard Extras.

```
With action ACTION_SEND you could specify the Extra key-value pairs

EXTRA_EMAIL. The "to" field
EXTRA_SUBJECT.
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Building an intent, https://developer.android.com/guide/components/intents-filters#Building

You can use custom Extra keys. To do so define your Extra key constant.

```
static final String EXTRA_GIGAWATTS = "com.example.EXTRA_GIGAWATTS";
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Building an intent, https://developer.android.com/guide/components/intents-filters#Building

Extra Java Methods: `putExtra()` can be used with key-value pairs; or use `putExtras()` with `Bundle` objects.

Flags

Flags provide metadata about the intent. See setFlags() in the Intent class.

Intent Creation

Build an Intent in Code

Build an (explicit) Intent (and optionally pack some data) as follows

```
/**
 * Called from activity_main.xml
```

```
 */
public void sendMessage(View view) {
           // Create intent by passing current context and target class.
           Intent intent = new Intent(this, DisplayMessageActivity.class);

           // Pack the intent with some data
           EditText editText = (EditText) findViewById(R.id.edit_message);
           String message = editText.getText().toString();
           intent.putExtra(EXTRA_MESSAGE, message);
           ....
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App basics, Build your first app, Start another activity, Build an intent, https://developer.android.com/training/basics/firstapp/starting-activity#BuildIntent

Without a component name, the intent is implicit and the system decides which component should receive the intent based on the other intent information (such as the action, data, and category).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Building an intent, https://developer.android.com/guide/components/intents-filters#Building

An Intent can carry a collection of various data types as key-value pairs called extras. The putExtra() method takes the key name in the first parameter and the value in the second parameter.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App basics, Build your first app, Start another activity, Build an intent, https://developer.android.com/training/basics/firstapp/starting-activity#BuildIntent

Example intent creation

Explicit intent.

```
/**
 * Called from activity_main.xml
 *
 * Demonstrates an explicit intent
 */
public void sendMessage(View view) {
           // Create intent by passing current context and target class.
           Intent intent = new Intent(this, DisplayMessageActivity.class);

           // Pack the intent with some data
           EditText editText = (EditText) findViewById(R.id.edit_message);
           String message = editText.getText().toString();
           intent.putExtra(EXTRA_MESSAGE, message);

           // Activate an app component with the intent, in this case an Activity
           startActivity(intent);
}
```

Explicit Intent, one line example.

```
public void goSettings(View view) {
    // Explicit intent
    startActivity(new Intent(this, SettingsActivity.class));
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\MainActivity.java

Implicit intent.

```
/**
 * Called from activity_main.xml
 *
 * Demonstrates an implicit intent
 */
public void shareMessage(View view) {
          Intent intent = new Intent();

          // Pack the intent with some data
          EditText editText = (EditText) findViewById(R.id.edit_message);
          String message = editText.getText().toString();
          intent.putExtra(Intent.EXTRA_TEXT, message);

          // Set other parts of the intent
          intent.setAction(Intent.ACTION_SEND);
          intent.setType(HTTP.PLAIN_TEXT_TYPE);

          // Verify that the intent will resolve to an activity
          if (intent.resolveActivity(getPackageManager()) != null) {
              startActivity(intent);
          }
}
```

Verify Intent is Safe to Start

Always verify that an implicit intent will resolve to be filtered for at least one intent before starting the Activity. Use `au.com.softmake.standardapplibrary.android.Intents.isIntentSafe(...)`

```
// Verify that the intent will resolve to an activity
if (au.com.softmake.standardapplibrary.android.Intents.isIntentSafe(mapIntent,
                                                                    this)) {
    startActivity(mapIntent);
} else {
    Toast.makeText(getApplicationContext(),
                   "You have no application to handle mapping.",
                   Toast.LENGTH_SHORT).show();
}

// au.com.softmake.standardapplibrary.android.Intents.isIntentSafe ...
public class Intents {
    public static boolean isIntentSafe(Intent intent, Context context) {
        return intent.resolveActivity(context.getPackageManager()) != null;
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\standardapplibrary\src\main\java\au\com\softmake\standardapplibrary\android\Intents.java

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Overview, Building an intent, Example implicity intent, https://developer.android.com/guide/components/intents-filters#ExampleSend

Alternative verification technique.

```
PackageManager packageManager = getPackageManager();
List<ResolveInfo> activities = packageManager.queryIntentActivities(intent, 0);
boolean isIntentSafe = activities.size() > 0;
// isIntentSafe = true if at least one App can handle your intent

if (isIntentSafe) {
    startActivity(intent);
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Interact with other apps, Sending the user to another app, Start an activity with the intent, https://developer.android.com/training/basics/intents/sending#StartActivity

Forcing a Chooser

When an implicit intent can be handled by multiple components or apps you have two design choices: allows the user to set a default; or force the user every time to choose between the multiple options. To force the user to choose every time, wrap your intent in a chooser intent.

```
/**
 * Demonstrates forcing an app chooser for implicit intents.
 *
 * Called from activity_main.xml
 */
public void shareMessageForceAppChooser(View view) {
          Intent targetIntent = new Intent();

          // Pack the targetIntent with some data
          EditText editText = (EditText) findViewById(R.id.edit_message);
          CharSequence message = editText.getText().toString();
          targetIntent.putExtra(Intent.EXTRA_TEXT, message);

          // Set other parts of the targetIntent
          targetIntent.setAction(Intent.ACTION_SEND);
          targetIntent.setType(HTTP.PLAIN_TEXT_TYPE);

          String title = getResources().getString(R.string.title_app_chooser);
          Intent chooserIntent = Intent.createChooser(targetIntent, title);

          // Verify that the intent will resolve to an activity before starting
          // the activity. You must use the targetIntent to test for resolution,
          // not the chooserIntent, otherwise strange behaviour might result.
          if (targetIntent.resolveActivity(getPackageManager()) != null) {
                    // Use the chooserIntent, not the targetIntent
                    startActivity(chooserIntent);
          }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Overview, Building an intent, Forcing an app chooser, https://developer.android.com/guide/components/intents-filters#ForceChooser

Use PendingIntents when you want a foregin app to execute the intent as if executed in your app's process. The three main cases: from your notification; from your App widget; and from the Alarm.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Overview, Using a pending intent, https://developer.android.com/guide/components/intents-filters#PendingIntent

Intent Receipt

Register component to receive implicit intents

Overview

Define the implicit intents a component can respond to in the manifest with intent filters.

```
<manifest ... >
    ...
    <application ... >
        <activity android:name="com.example.project.ComposeEmailActivity">
            <intent-filter>
                <action android:name="android.intent.action.SEND" />
                <data android:type="*/*" />
                <category android:name="android.intent.category.DEFAULT" />
```

```
            </intent-filter>
        </activity>
    </application>
</manifest>
```

The way the system identifies the components that can respond to an intent is by comparing the intent received to the intent filters provided in the manifest file applications.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App Basics, App fundamentals, The manifest file, Declaring component capabilities, https://developer.android.com/guide/components/fundamentals#DeclaringComponentCapabilities

Register a component to recieve implicit intents in AndroidManifest.xml using one or more of action, data, and category elements.

```
<activity android:name="ShareActivity">
    <!-- This activity handles "SEND" actions with text data -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
    <!-- This activity also handles "SEND" and "SEND_MULTIPLE" with media data -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <action android:name="android.intent.action.SEND_MULTIPLE"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="application/vnd.google.panorama360+jpg"/>
        <data android:mimeType="image/*"/>
        <data android:mimeType="video/*"/>
    </intent-filter>
</activity>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Overview, Receiving an implicit intent, https://developer.android.com/guide/components/intents-filters#Receiving

If you do **not** declare any intent filters for an activity, then it can be started only with an explicit intent.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Overview, https://developer.android.com/guide/components/intents-filters

Always use an explicit intent when starting a service.

Using an implicit intent to start a service is a security hole.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Overview, https://developer.android.com/guide/components/intents-filters

For all acitivites that you wish to receive implicit intents you must declare them in the AndroidManifest.xml. For broadcast receivers, however, you can register them dynamically with `registerReceiver()` (and you can use `unregisterReceiver()`).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Overview, Receiving an implicit intent, https://developer.android.com/guide/components/intents-filters#Receiving

<data> declares the type of data accepted, using one or more attributes that specify various aspects of the data URI (scheme, host, port, path, etc.) and MIME type.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Overview, Receiving an implicit intent, https://developer.android.com/guide/components/intents-filters#Receiving

In order to receive implicit intents, you must include the CATEGORY_DEFAULT category in the intent filter. The methods `startActivity()` and `startActivityForResult()` treat all intents as if they declared the CATEGORY_DEFAULT category.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Overview, Receiving an implicit intent, https://developer.android.com/guide/components/intents-filters#Receiving

Implicit Intent Resolution

Implicit intents are resolved using the action, data, and category elements in the following way.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Overview, Intent resolution, https://developer.android.com/guide/components/intents-filters#Resolution

Action: zero or more actions allowed; one action match means the whole intent matches; If the xml filter contains no actions, it fails. If the Intent does not specify an action, it will pass (so long as the filter contains at least one action).

Category: zero or more categories allowed; all categories in the calling intent must match for the whole intent to match; if more categories are be declared, in xml, than the calling intent includes, then the whole intent matches; a calling intent with no categories always matches; Always include "android.intent.category.DEFAULT" in the xml.

Data: zero or more data are allowed; each <data> specifys a URI and/or a MIME type;

The Data URI has parts (scheme, host, port, path)

```
<scheme>://<host>:<port>/<path>
content://com.example.project:200/folder/subfolder/etc

scheme: content
host: com.example.project
port: 200
path: folder/subfolder/etc
```

Programmatically discover available intents

To programmatically discover available intents you can use PackManager.query....() and PackageManager.resolve....()

```
queryIntentActivities()
queryIntentServices()
queryBroadcastReceivers()
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Overview, Intent resolution, Intenting matching, https://developer.android.com/guide/components/intents-filters#imatch

Receive the Intent

Receive the intent with `getIntent()` in the `OnCreate()` method. Extract the data from the **extra** by using `getStringExtra` with the relevant key.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           setContentView(R.layout.activity_display_message);
           // Show the Up button in the action bar.
           setupActionBar();

           // Get the message from the intent
           Intent intent = getIntent();
           String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

           // Display results
           TextView textView = (TextView) findViewById(R.id.display_message_textview);
           textView.setText(message);
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App basics, Build your first app, Start another activity, https://developer.android.com/training/basics/firstapp/starting-activity

For an example of receiving and handling an intent that uses branching according to Action and Data (E.g. using a mime type), see ...

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing simple data, Receiving simple data from other apps, https://developer.android.com/training/sharing/receive

Returning Results

Request a Result from an Activity (with an Intent)

In your sending Activity: Create an Intent, in this example an implicit Intent, then use `startActivityForResult()`, passing the intent.

```
// In ActionBarActivity

static final int REQUEST_SELECT_CONTACT = 1;

public void getgetContactFieldValue(View view) {
           Intent intent = new Intent(Intent.ACTION_PICK);
           intent.setType(ContactsContract.Contacts.CONTENT_TYPE);
           if (intent.resolveActivity(getPackageManager()) != null) {
                      startActivityForResult(intent, REQUEST_SELECT_CONTACT);
           }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Common intents, Contacts/People App, https://developer.android.com/guide/components/intents-commonl#Contacts

Return a Result

To return a result you return either an intent or a code.

Return a result to an Activity (from an Activity) with setResult(). You should then use finish() to close and destroy your Activity. The following example returns an Intent.

```
// In Activity which returns a result
Intent result = new Intent("com.example.RESULT_ACTION", Uri.parse("content://result_uri"));
setResult(Activity.RESULT_OK, result);
```

```
finish();
```

There is no need to check with the current Activity was started with startActivity() or startActivityForResult(). The calling activity will ignore a returned result if it used startActivity() to call your returning Activity.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Interact with other apps, Allowing other apps to start your activity, Return a result, https://developer.android.com/training/basics/intents/filters#ReturnResult

If you only need to return a result code and not an intent then do this ...

```
// Result code must be greater than 0.
setResult(RESULT_COLOR_RED);
finish()
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Interact with other apps, Allowing other apps to start your activity, Return a result, https://developer.android.com/training/basics/intents/filters#ReturnResult

Handle a Result from an Activity

In your sending Activity: Receive the Intent that returns from an external Activity using `onActivityResult()`.

```
// Also in ActionBarActivity

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent) {

          if (requestCode == REQUEST_SELECT_CONTACT && resultCode == RESULT_OK) {
                    Uri contactUri = intent.getData();

                    String contactDataColumn = CommonDataKinds.Phone.DISPLAY_NAME;

                    String[] projection = new String[]{contactDataColumn};
                    String phoneNumber = "";
                    Cursor cursor = null;

                    try {
                              cursor = getContentResolver().query(contactUri,
projection, null, null, null);
                    // If field not available in selected contact then an exception is
thrown
                    // but is not trappable (this is strange).
                    // We therefore continue execution with a "finally" clause an test
cursor for null.
                    } finally {
                              if (cursor != null && cursor.moveToFirst()) {
                                        int columnIndex =
cursor.getColumnIndex(contactDataColumn);
                                        phoneNumber = cursor.getString(columnIndex);
                              } else {

                   String text = "No such field in contact to return";
                                        Toast toast =
Toast.makeText(getApplicationContext(), text,
                                  Toast.LENGTH_SHORT);
                                        toast.show();

              }
                    }

                    EditText editText = (EditText) findViewById(R.id.edit_message);
                    editText.setText(phoneNumber);
          }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Intents and intent filters, Common intents, Contacts/People App, https://developer.android.com/guide/components/intents-commonl#Contacts

Common Intents

Intro

See (Google API Guides, 2013. Develop > API Guides) http://developer.android.com/guide/components/intents-common.html for using Intents to work with ...

- Camera
- Contacts/People App
- Email
- File Storage
- Maps
- Music or Video
- Phone
- Settings
- Text Messaging
- Web Browser

Maps

Map Example

```
public void viewMap(View view) {
    // Uri based on search string
    Uri locationUri = Uri.parse(
            "geo:0,0?q=Ahuriri+Conservation+Park,+Otago,+New+Zealand");

    // Uri locationUri = Uri.parse("geo:37.422219,-122.08364?z=14"); // z param is
    // zoom level

    Intent mapIntent = new Intent(Intent.ACTION_VIEW, locationUri);

    // Verify that the intent will resolve to an activity
    if (mapIntent.resolveActivity(getPackageManager()) != null) {
        startActivity(mapIntent);
    } else {
        Toast.makeText(getApplicationContext(),
                       "You have no application to handle mapping.",
                       Toast.LENGTH_SHORT).show();
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Interact with other apps, Sending the user to another app, https://developer.android.com/training/basics/intents/sending

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\MoreIntentsActivity.java

Send

Send/Share some text with an implicit Intent using ACTION_SEND

```
public void goSendText(View view) {
    // Implicit intent.
    Intent sendIntent = new Intent();
    sendIntent.putExtra(Intent.EXTRA_TEXT, "Text to share");
```

```
    sendIntent.setAction(Intent.ACTION_SEND);
    sendIntent.setType("text/plain");

    if (au.com.softmake.standardapplibrary.android.Intents.isIntentSafe(sendIntent,
                                                                        this)) {
       startActivity(sendIntent);
    } else {
        Toast.makeText(getApplicationContext(), "No apps can receive your text",
                       Toast.LENGTH_SHORT).show();
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing simple data, Sending simple data to other apps, https://developer.android.com/training/sharing/send

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\SendingSimpleDataActivity.java

Todo. Send a Binary

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing simple data, Sending simple data to other apps, https://developer.android.com/training/sharing/send

Todo. Send Multiple Pieces of Content

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing simple data, Sending simple data to other apps, https://developer.android.com/training/sharing/send

Activities

Establish activities in the navigation hierarchy

In AndroidManifet.xml, when declaring an activity, reference its parent activity as in the example below. Eclipse will generate this automatically for you.

```
<activity
    android:name="au.com.softmake.myfirstapp.DisplayMessageActivity"
    android:label="@string/title_activity_display_message"
    android:parentActivityName="au.com.softmake.myfirstapp.MainActivity" >
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="au.com.softmake.myfirstapp.MainActivity" />
</activity>
```

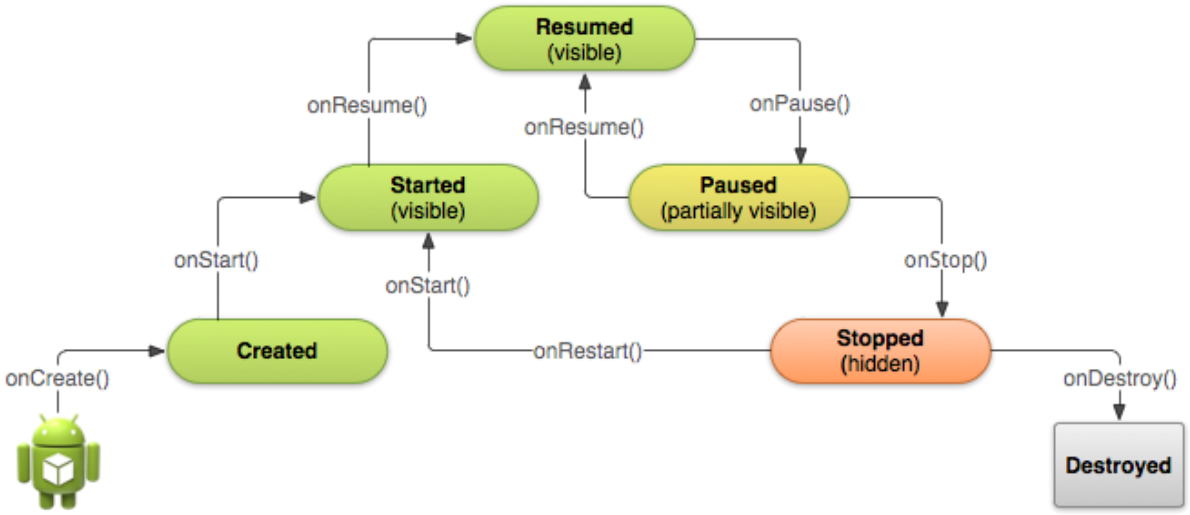
The android:parentActivityName attribute declares the name of this activity's parent activity within the app's logical hierarchy. The system uses this value to implement default navigation behaviors, such as Up navigation on Android 4.1 (API level 16) and higher. You can provide the same navigation behaviors for older versions of Android by using the Support Library and adding the <meta-data> element as shown here.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App basics, Build your first app, Start another actvitiy, Create the second activity, https://developer.android.com/training/basics/firstapp/starting-activity#CreateActivity

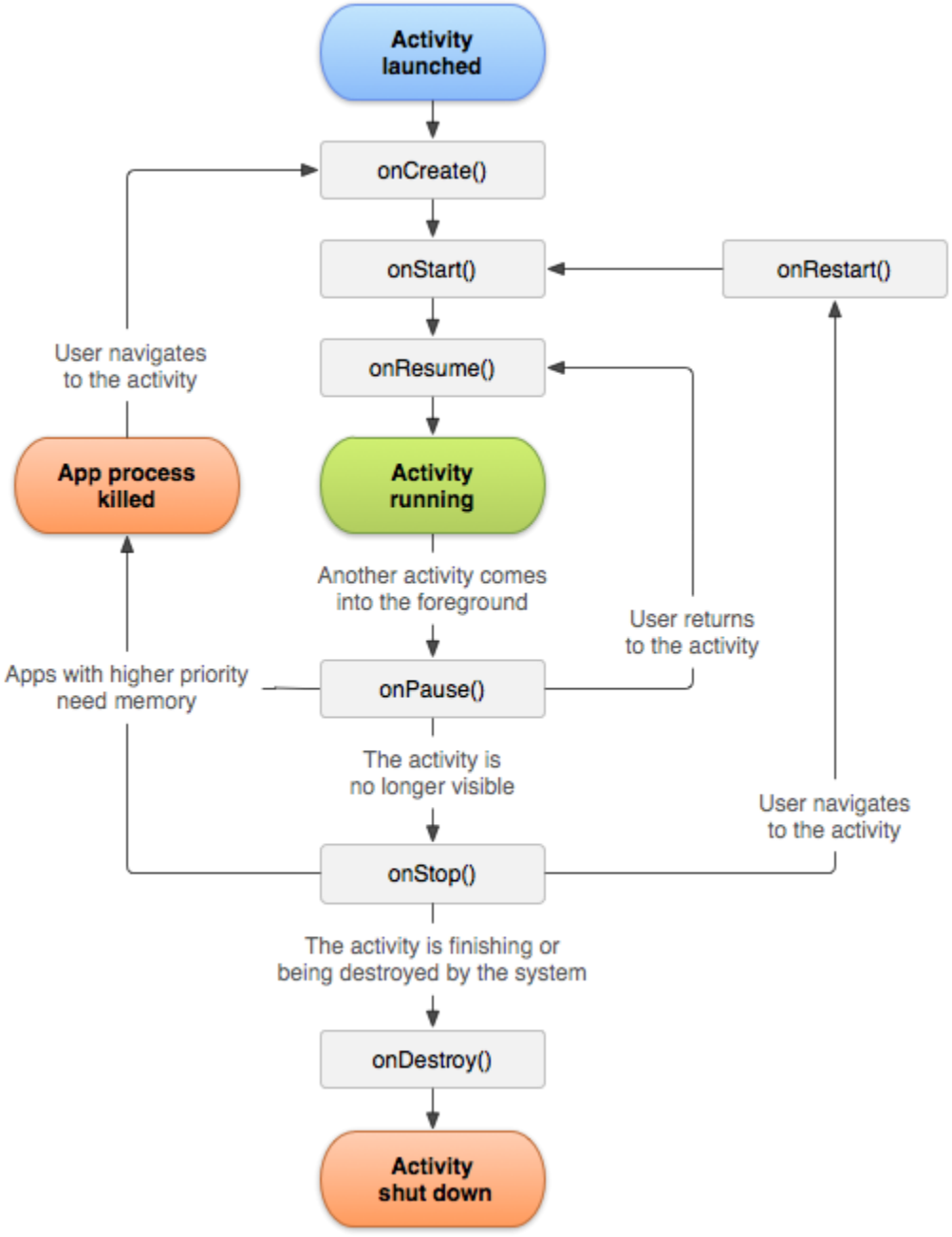
Activity states and events

The three main states (states which are static) of an activity:

- Resumed/Running. In the foreground and has focus.
- Paused. Still visible but another activity is in the foreground and has focus. Still alive: in memory; maintains state, and attached to window manager.
- Stopped. Completely obscured by another activity. Still alive: In memory; maintains state, but **not** attached to window manager.



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, The activity lifecycle, https://developer.android.com/guide/components/activities/activity-lifecycle#lifecycle-states



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, The activity lifecycle, Lifecycle callbacks, https://developer.android.com/guide/components/activities/activity-lifecycle#lc

Table 3 Lifecycle activities

Entire	Visible	Foreground	Method	Description	Next
			`onCreate()`	Called when the activity is first created. This is where you should do all of your normal static set up — create views, bind data to lists, and so on. This method is passed a Bundle object containing the activity's previous state, if that state was captured (see Saving Activity State, later).	`onStart()`
			`onRestart()`	Called after the activity has been stopped, just prior to it being started again.	`onStart()`

			`onStart()`	Called just before the activity becomes visible to the user.
				
				Followed by onResume() if the activity comes to the foreground, or onStop() if it becomes hidden.
					`onResume()`
					`or`
					`onStop()`

			`onResume()`	Called just before the activity starts interacting with the user. At this point the activity is at the top of the activity stack, with user input going to it. Always followed by onPause().	`onPause()`

			`onPause()`	Called when the system is about to start resuming another activity. This method is typically used to commit unsaved changes to persistent data, stop animations and other things that may be consuming CPU, and so on. It should do whatever it does very quickly, because the next activity will not be resumed until it returns.
				
				Followed either by onResume() if the activity returns back to the front, or by onStop() if it becomes invisible to the user.
					`onResume()`
					`or`
					`onStop()`

			`onStop()`	Called when the activity is no longer visible to the user. This may happen because it is being destroyed, or because another activity (either an existing one or a new one) has been resumed and is covering it.
				
				Followed either by onRestart() if the activity is coming back to interact with the user, or by onDestroy() if this activity is going away.
					`onRestart()`
					`or`
					`onDestroy()`

			`onDestroy()`	Called before the activity is destroyed. This is the final call that the activity will receive. It could be called either because the activity is finishing (someone called finish() on it), or because the system is temporarily destroying this instance of the activity to save space. You can distinguish between these two scenarios with the isFinishing() method.	`nothing`

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, The activity lifecycle, Lifecycle callbacks, https://developer.android.com/guide/components/activities/activity-lifecycle#lc

Coding guidance for using Activity events (using the callbacks).

OnCreate. Reference the activity layout with `setContentView()` in the `onCreate` event. Use Eclipse to generate this for you: File > New > Other ... > Android > Android Activity.

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

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App Basics, Build your first app, Start another activity, Create the second activity, https://developer.android.com/training/basics/firstapp/starting-activity#CreateActivity

OnCreate. Start other activities. See Intent Creation.

OnCreate. Receive Intents and startActivityForResult. See Create and receive an intent that visits an external app.

OnPause.

- Stop animations or other ongoing actions that could consume CPU.
- Commit unsaved changes if users expect such changes to be permanently saved when they leave (such as a draft email).
- Release system resources, such as broadcast receivers, handles to sensors (like GPS), or any resources that may affect battery life while your activity is paused and the user does not need them.
- Avoid performing CPU-intensive work during onPause(), such as writing to a database, because it can slow the visible transition to the next activity.

```
@Override
public void onPause() {
    super.onPause();  // Always call the superclass method first

    // Release the Camera because we don't need it when paused
    // and other activities might need to use it.
    if (mCamera != null) {
        mCamera.release()
        mCamera = null;
    }
}
```

You should use onPause() to write crucial persistent data (such as user edits) to storage. However, you should be selective about what information must be retained during onPause(), because any blocking procedures in this method block the transition to the next activity and slow the user experience.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, The activity lifecycle, https://developer.android.com/guide/components/activities/activity-lifecycle

onPause(). You should use onPause() to write crucial persistent data (such as user edits) to storage.

This is the last method that's guaranteed to be called before the process **can** be killed (although an activity can be killed in exceptional circumstances before this).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, The activity lifecycle, Lifecycle callbacks, https://developer.android.com/guide/components/activities/activity-lifecycle#lc

onResume(). When your user resumes activity from a paused sate. Initialise componets that you release during onPause().

```
@Override
public void onResume() {
    super.onResume();  // Always call the superclass method first

    // Get the Camera instance as the activity achieves full user focus
    if (mCamera == null) {
        initializeCamera(); // Local method to handle camera init
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, The activity lifecycle, Lifecycle callbacks, https://developer.android.com/guide/components/activities/activity-lifecycle#lc

OnStop. Perform heavy-load shutdown operations.

```
@Override
protected void onStop() {
    super.onStop();  // Always call the superclass method first

    // Save the note's current draft, because the activity is stopping
    // and we want to be sure the current note progress isn't lost.
    ContentValues values = new ContentValues();
    values.put(NotePad.Notes.COLUMN_NAME_NOTE, getCurrentNoteText());
    values.put(NotePad.Notes.COLUMN_NAME_TITLE, getCurrentNoteTitle());

    getContentResolver().update(
            mUri,    // The URI for the note to update.
            values,  // The map of column names and new values to apply to them.
            null,    // No SELECT criteria are used.
            null     // No WHERE columns are used.
            );
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, The activity lifecycle, Lifecycle callbacks, https://developer.android.com/guide/components/activities/activity-lifecycle#lc

For many activities you will not have to implement onStop(), onRestart, or even onStart().

Many activities are simple that way.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, The activity lifecycle, https://developer.android.com/guide/components/activities/activity-lifecycle

onStart() and onRestart(). Generally use onStart() for any (re)initialisation of objects destroyed in onStop() as these objects will need initialising on start anyway. That is, you'll really need to use onRestart().

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, The activity lifecycle, Lifecycle callbacks, https://developer.android.com/guide/components/activities/activity-lifecycle#lc

OnDestory. Release all remaining resources, if any, in onDestory().

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, The activity lifescycle, Lifecycle callbacks, https://developer.android.com/guide/components/activities/activity-lifecycle#lc

OnDestroy. Most apps don't need to implement this. Your activity should perform most cleanup during `onPause()` and `onStop()`. The exception includes any background threads you started in `onCreate()`.

```
@Override
public void onDestroy() {
    super.onDestroy();  // Always call the superclass

    // Stop method tracing that the activity started during onCreate()
    android.os.Debug.stopMethodTracing();
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, The activity lifescycle, Lifecycle callbacks, https://developer.android.com/guide/components/activities/activity-lifecycle#lc

Shutting down an activity

Rarely needed. Shut down current activity (?) with `finish()` and an acivity you have previously started with `finishActivity()`.

In most cases, you should not explicitly finish an activity using these methods. As discussed in the following section about the activity lifecycle, the Android system manages the life of an activity for you, so you do not need to finish your own activities. Calling these methods could adversely affect the expected user experience and should only be used when you absolutely do not want the user to return to this instance of the activity.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, The activity lifescycle, Lifecycle callbacks, https://developer.android.com/guide/components/activities/activity-lifecycle#lc

Fragments

Frements Intro

Fragments are designed to support applications running either on mobiles or tablets. You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

Fragments are supported from Anroid 3.0 (API level 11).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

When you add a fragment as a part of your activity layout, it lives in a ViewGroup inside the activity's view hierarchy.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

Fragments do not necessarily have a UI.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

You should design each fragment as a modular and reusable activity component.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

Procedure Template

The following procedure reproduces the code automatically generated in Android Studio when you right click on a Java folder and go: New > Activity > Blank Activity with Fragment. This creates a basic fragment for use within an activity.

Activity Layout File.

```
<!-- \res\layout\activity_with_fragment.xml -->
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="au.com.softmake.mysecondapp.ActivityWithFragment"
    tools:ignore="MergeRootFrame" />
```

Fragment Layout File

```
<!-- res\layout\fragment_placeholder.xml -->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="au.com.softmake.mysecondapp.ActivityWithFragment$PlaceholderFragment">

    <TextView
        android:text="@string/hello_world"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

Activity class containing fragment class

```
// ActivityWithFragment.java

...
import android.support.v4.app.Fragment;
import android.support.v7.app.ActionBarActivity;
...

public class ActivityWithFragment extends ActionBarActivity {

    /**
     * A placeholder fragment containing a simple view.
     */
    public static class PlaceholderFragment extends Fragment {

        public PlaceholderFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            View rootView = inflater.inflate(R.layout.fragment_placeholder,
                                             container,
                                             false);
            return rootView;
        }
    }

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

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

    ...
}
```

AndroidManifest.xml activity declaration

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

...

    <application>
        <activity
            android:name=".ActivityWithFragment"
            android:label="@string/title_activity_activity_with_fragment"
            android:parentActivityName=".MainActivity">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="au.com.softmake.mysecondapp.MainActivity" />
        </activity>
    </application>
</manifest>
```

Procedure Overview

Create a Fragement UI

Create a fragment UI. You may need to provide different fragment UIs depending on whether in portrait or landscape.

```
<!-- Portrait res/layout/example_fragment.xml, i.e. portrait layout -->

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">
    <fragment class="com.example.android.apis.app.FragmentLayout$TitlesFragment"
            android:id="@+id/titles"
            android:layout_width="match_parent" android:layout_height="match_parent" />
</FrameLayout>


<!-- Landscape es/layout-land/example_fragment.xml -->

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent" android:layout_height="match_parent">

    <fragment class="com.example.android.apis.app.FragmentLayout$TitlesFragment"
            android:id="@+id/titles" android:layout_weight="1"
            android:layout_width="0px" android:layout_height="match_parent" />

    <FrameLayout android:id="@+id/details" android:layout_weight="1"
            android:layout width="0px" android:layout height="match parent"
            android:background="?android:attr/detailsElementBackground" />

</LinearLayout>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

Create a Fragment Class

Create a subclass of Fragment (or an existing subclass of it). You will usually implement at least onCreate(), OnCreateView(), and OnPause().

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Create, https://developer.android.com/guide/fragments/create

Consider creating a Frament from one of the specialised standard subclasses, if the need suits: DialogFragment; ListFragment; PreferenceFragment.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Create, https://developer.android.com/guide/fragments/create

Add the fragment UI to the FramentClass. Fragment onCreateView(). The system calls this when it's time for the fragment to draw its user interface for the first time. To draw a UI for your fragment, you must return a View from this method that is the root of your fragment's layout. You can return null if the fragment does not provide a UI.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Create, https://developer.android.com/guide/fragments/create

```
public static class ExampleFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        // Inflate the layout for this fragment using a layout defined in
        // res/layout/example_fragment.xml

        // The container parameter passed to onCreateView() is the parent
        // ViewGroup (from the activity's layout) in which your fragment
        // layout will be inserted
        return inflater.inflate(R.layout.example_fragment, container, false);
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Create, https://developer.android.com/guide/fragments/create

Add a fragment to an activity

There are three ways to add a fragment to an activity:

Declare the fragment inside the activity layout xml.

In the activity programmatically add the fragment to an existing ViewGroup.

In the activity programmatically add the fragment without a UI.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Create, Add a fragment to an activity, https://developer.android.com/guide/fragments/create#add

Declare the fragment inside the activity layout xml. Reference the class name of the fragment with android:name.

```
<!-- layout for an activity with two fragments -->

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

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.news.ArticleListFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
    <fragment android:name="com.example.news.ArticleReaderFragment"
            android:id="@+id/viewer"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
</LinearLayout>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Create, Add a fragment to an activity, https://developer.android.com/guide/fragments/create#add

Always ad an android:id (or android:tag) to the fragment so state can be restored by default.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Create, Add a fragment to an activity, https://developer.android.com/guide/fragments/create#add

In the activity programmatically add the fragment to an existing ViewGroup.

```
FragmentManager fragmentManager = getFragmentManager()
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

ExampleFragment fragment = new ExampleFragment();

// R.id.fragment_container is a view group.
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Create, Add a fragment to an activity, https://developer.android.com/guide/fragments/create#add

In the activity programmatically add the fragment without a UI. To add a fragment without a UI, add the fragment from the activity using add(Fragment, String) (supplying a unique string "tag" for the fragment, rather than a view ID).

This adds the fragment, but, because it's not associated with a view in the activity layout, it does not receive a call to onCreateView(). So you don't need to implement that method.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Create, Add a fragment to an activity, https://developer.android.com/guide/fragments/create#add

Every fragment must have an empty constructor to help with saving state.

```
public static class PlaceholderFragment extends Fragment {


    public PlaceholderFragment() {
    }
```

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), Fragment, https://developer.android.com/reference/android/app/Fragment

Within the fragment use findViewById or findViewByTag to access UI elements

```
public static class PlaceholderFragment extends Fragment {
```

```
    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        View rootView = inflater.inflate(R.layout.fragment_radio_frequency,
                                         container,
                                         false);
        TextView outputTextView = (TextView) rootView.findViewById(R.id.textView_output);
        outputTextView.setText("cools");
        return rootView;
    }
```

C:\Users\John\Documents\Sda\Code\Android\Apps\AndroidSystemInfo\app\src\main\java\au\com\softmake\androidsysteminfo\RadioFrequencyActivity.javas

Fragment manager and fragment transactions

The FragmentManager has useful methods.

```
findFragmentById()
findFragmentByTag()
popBackStack()
addOnBackStackChangedListener()

FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Fragment manager, https://developer.android.com/guide/fragments/fragmentmanager

Fragment transactions allow you, from within your activity, to add, remove, and replace fragments. Fragment transactions also allow you to manipulate the backstack (for fragments) and respond to user interaction.

```
// Create new fragment and transaction
Fragment newFragment = new ExampleFragment();
FragmentTransaction transaction = getFragmentManager().beginTransaction();

// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack
transaction.replace(R.id.fragment_container, newFragment);
transaction.addToBackStack(null);

// Commit the transaction
transaction.commit();
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Fragment transactions, https://developer.android.com/guide/fragments/transactions

With fragment transactions commit() actually effects all the changes you specified. Well actually it schedules it to run with the activities UI thread is able to do so.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Fragment transactions, https://developer.android.com/guide/fragments/transactions

In an activity you must commit() (the fragment transaction) before the activity state saving.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Fragment transactions, https://developer.android.com/guide/fragments/transactions

Handling UI onclick events

See Event Listener Best Practice

Fragment Lifecycle

Fragment Lifecycle

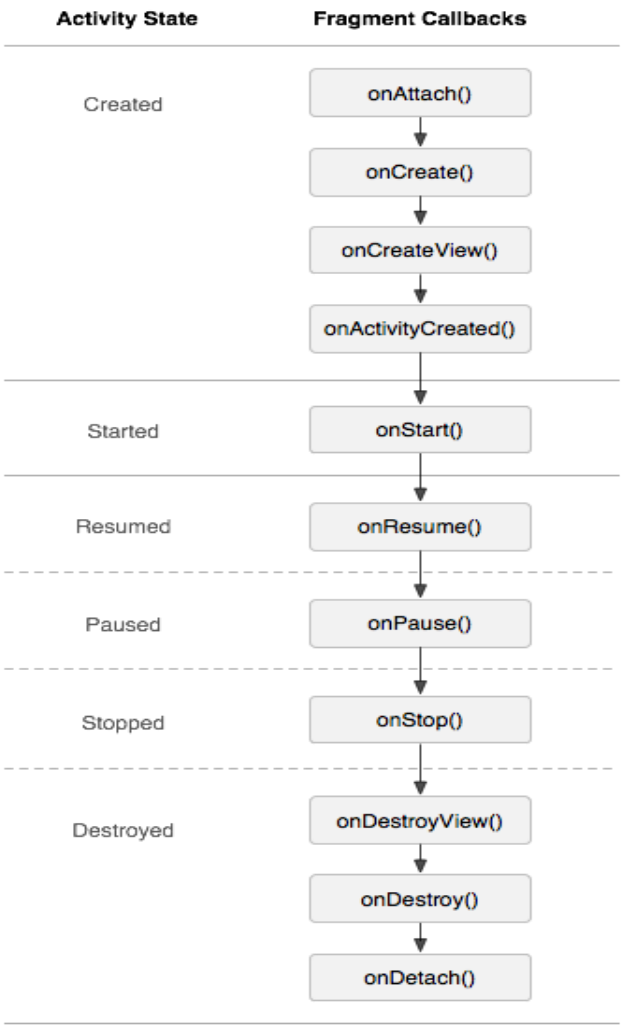


Figure 2 Fragment Lifecycle

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Fragment lifecycle, https://developer.android.com/guide/fragments/lifecycle

Like an activity a fragment can exist in three states:

Resumed. Visible in running activity.

Paused. Another activity foregrounded. Activity in which this fragment lives is still visible (foreground activity is partially transparent or doesn't cover the entire screen).

Stopped. Not visible. Host activity stopped or fragment removed from activity but still on the backstack. A stopped fragment is alive (state and member information retained in the system).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Fragment lifecycle, https://developer.android.com/guide/fragments/lifecycle

For backstack management a fragment behaves unlike an activity. Unlike for an activity, with a fragment you must explicitly call addToBackStack() during a transaction that removes the fragment.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Fragment lifecycle, https://developer.android.com/guide/fragments/lifecycle

Mostly, each event for an activity results in a similar event for a fragment. E.g. When an activity receives onPause(), each fragment in the activity receives onPause(). However fragments have a view extra events as follows:

onAttach(). When fragment has been associated with the activity. The activity is passed in here.

onCreateView(). To create the view hierarchy associated with the fragment.

onActivityCreated(). The activity's onCreate() has returned.

onDestroyView(). The view hierarchy associated with the fragment is being removed.

onDetach(). The fragment is being disassociated from the activity.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

Figure 2 Fragment Lifecycle shows the next available fragment event that occurs after an activity event. E.g. After the activity has received its onCreate() event, the fragment receives no more than the onActivityCreated() event.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

Only after the activity onResume() can you add and remove fragments to the activity.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

Communicating with the Activity

Basic references

From the fragment, access the containing activity instance.

```
View listView = getActivity().findViewById(R.id.list);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

From the activity, access the fragment instance.

```
// All on one line
ExampleFragment fragment = (ExampleFragment) ...
   getFragmentManager().findFragmentById(R.id.example_fragment);

ExampleFragment fragment = (ExampleFragment) ...
   getFragmentManager().findFragmentByTag(fragmentTag);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

From fragment to activity

When you want to communicate from the fragment to the activity: create event callbacks to the activity:

Fragment Class. Define an interface with a callback function which the Activity is to implement.

```
public static class FragmentA extends ListFragment {
    ...
    // Container Activity must implement this interface
    public interface OnArticleSelectedListener {
        public void onArticleSelected(Uri articleUri);
    }
    ...
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

Fragment Class. Force the activity to implement the callback function.

```
public static class FragmentA extends ListFragment {
    OnArticleSelectedListener mListener;
    ...
    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mListener = (OnArticleSelectedListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString() + " must implement
OnArticleSelectedListener");
        }
    }
    ...
}
```

Fragment Class. Raise the event by calling the Activity callback implementation.

```
public static class FragmentA extends ListFragment {
    OnArticleSelectedListener mListener;
    ...
    @Override
    public void onListItemClick(ListView l, View v, int position, long id) {
        // Append the clicked item's row ID with the content provider Uri
        Uri noteUri = ContentUris.withAppendedId(ArticleColumns.CONTENT_URI, id);
        // Send the event and Uri to the host activity
        mListener.onArticleSelected(noteUri);
    }
```

```
    ...
}
```

Activity Class. Respond to the event by implementing the callback function.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

From activity to fragment

To communicate from activity to fragment use setArguments(Bundle) and getArguments().

```
/**
* Activity
*/
private void showAddScheduleExerciseDialog(int scheduleID, int exerciseID) {

    Bundle bundle = new Bundle();

    // Create dialog if it doesn't exist
    if (addScheduleExerciseDialog == null) {
        addScheduleExerciseDialog = new AddScheduleExerciseDialog();

        bundle.putInt("scheduleID", scheduleID);
        bundle.putInt("exerciseID", exerciseID);
        addScheduleExerciseDialog.setArguments(bundle);
    }
    // Show dialog
    addScheduleExerciseDialog.show(getFragmentManager(),
                                   "AddScheduleExerciseDialogTag");
}

/**
 * Dialog (fragment)
 */
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    Bundle argumentBundle = getArguments();

    // You could use getArguments() directly put I presume it's more efficient to
    // retrieve the bundle object once.
    int scheduleID = argumentBundle.getInt("scheduleID");
```

"Default constructor. Every fragment must have an empty constructor, so it can be instantiated when restoring its activity's state. It is strongly recommended that subclasses do not have other constructors with parameters, since these constructors will not be called when the fragment is re-instantiated; instead, arguments can be supplied by the caller with setArguments(Bundle) and later retrieved by the Fragment with getArguments()."

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), Fragment, https://developer.android.com/reference/android/app/Fragment

Menus

From your fragment add menu items to the activity's options menu (on the action bar): During the frament's onCreate() call setHasOptionsMenu(); Implement (in the fragment) onCreateOptionsMenu(); respond to menu item selection by implement (in the fragment) onOptionsItemSelected().

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Working with the app bar, https://developer.android.com/guide/fragments/appbar

From your fragment add a context menu: call registerForContextMenu(); respond to menu opening by implementing onCreateContextMenu(); respond to item selected by implementing onContextItemsSelected().

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Working with the app bar, https://developer.android.com/guide/fragments/appbar

Example

Fragment example.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Fragments, Overview, https://developer.android.com/guide/fragments

Tasks and the Back Stack

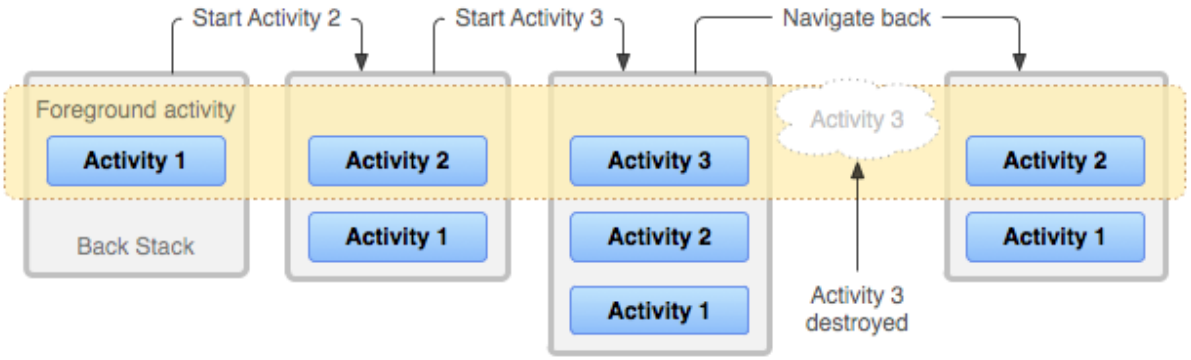
A task is a collection of activities grouped into a coherent job.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Tasks and back stack, https://developer.android.com/guide/components/activities/tasks-and-back-stack

The Home Screen is that starting place for most tasks. When the users touches the application icon on the home screen that application's task comes to the foreground. If no task exists for the application (the application has not been used recently), then a new task is crated and the main activity for that application opens as the root activity in the stack.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Tasks and back stack, https://developer.android.com/guide/components/activities/tasks-and-back-stack

Each new activity in a task adds an item to the back stack. When the user presses the **Back** button, the current activity is destroyed and the previous activity resumes.



(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Tasks and back stack, https://developer.android.com/guide/components/activities/tasks-and-back-stack

Each task maintains its own backstack. Two tasks, and corresponding backstacks, will occur if in the middle of a task the user navigates back to the Home Screen (by pressing the Home button).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Tasks and back stack, https://developer.android.com/guide/components/activities/tasks-and-back-stack

Basic task and back stack default operation:

- When Activity A starts Activity B, Activity A is stopped, but the system retains its state (such as scroll position and text entered into forms). If the user presses the **Back** button while in Activity B, Activity A resumes with its state restored.
- When the user leaves a task by pressing the **Home** button, the current activity is stopped and its task goes into the background. The system retains the state of every activity in the task. If the user later resumes the task by selecting the

launcher icon that began the task, the task comes to the foreground and resumes the activity at the top of the stack.

- If the user presses the **Back** button, the current activity is popped from the stack and destroyed. The previous activity in the stack is resumed. When an activity is destroyed, the system **does not** retain the activity's state.
- Activities can be instantiated multiple times, even from other tasks.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Tasks and back stack, https://developer.android.com/guide/components/activities/tasks-and-back-stack

You can alter the default operation through `<activity>` attributes in the manifest and/or through flags when calling `startActivity`(). But unless you had a good reason to do this you wouldn't want to change the default operation expected by the user.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Tasks and back stack, https://developer.android.com/guide/components/activities/tasks-and-back-stack

Services

Intro

A Service is an application component that performs long-running background operations and does not have a user interface.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, https://developer.android.com/guide/components/services

A Service might handle network transactions, play music, perform file I/O, or interact with a content provider.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, https://developer.android.com/guide/components/services

There are two kinds of Service:

- Started: started when an application component (e.g. an Activity) calls `startService()`.
- Can run indefinitely (even with the starting app component has been destroyed);
- Usually performs a single operation;
- Usually does not return a result;
- A Started Service should stop itself.
- Example: download or upload a file over the network.

- Bound: bound when an application component calls `bindService()`.
- Offers a client-server interface that allows client components to interact with it (even across processes "interprocess communication (IPC)";
- Lasts as long as there is another application component bound to it;
- Multiple components can bind to a service.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, https://developer.android.com/guide/components/services

A Service can be Started and Bound, so long as the service implements the corresponding supporting call back methods, `onStartCommand()` and `onBind()` respectively.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, https://developer.android.com/guide/components/services

A Service is started or bound with Intents.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, https://developer.android.com/guide/components/services

Common Coding Considerations

Create a Service by creating a subclass of Service; or by subclassing one of the existing subclasses of Service.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, The basics, https://developer.android.com/guide/components/services#Basics

By default a Service runs in the same thread and process as the hosting process. If the service is going to do intensive work you should create a new thread within the service, to do the work.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, https://developer.android.com/guide/components/services

If you need to do work outside the main thread only while the user is interacting with the application you should probably just create a new thread and not a service. Consider using AsyncTask or HandlerThread, instead of the traditional Thread class.

```
For example, if you want to play some music, but only while your activity is running, you might create a thread in onCreate(), start running it in onStart(), then stop it in onStop().
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, https://developer.android.com/guide/components/services

When creating your Service you need to override some key callback methods, depending on whether you are Starting or Binding a Service:

- `onCreate()`
- Called when service is first created.
- Runs before `onStartCommand()` or `onBind()`.
- If service already running, the method is not called.
- `onStartCommand()`
- Called when client code calls `startService()`.
- You must stop the service at some point with `stopSelf()` or `stopService()`, if the service is subclassed from Service.
- `onBind()`
- Called when client code calls `bindService()`.
- You must return an `IBinder` interface to the client code.
- You must always implement this method but return null if you don't want to allow binding.
- `onDestroy()`
- Perform any clean up here of: threads, registered listenters, receivers, etc.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, The basics, https://developer.android.com/guide/components/services#Basics

Running time of the Service:

- Started Service (subclassed from Serivce): runs indefinitely until it stops itself, with `stopSelf()`, or client code stops it, with `stopService()`.
- Started Service (subclassed from IntentService): Stops the service after all start requests have been handled, so you never have to call stopSelf().
- Bound Service: runs until all client code is unbound from the service.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, The basics, https://developer.android.com/guide/components/services#Basics

The Android system can stop and destroy your service at any time. This can occur when memory is low and the system needs to recover resources. The Android system stopping and destruction of your service is more or less likely (from more likely to less likely) as

follows: regular background Started service; Bound service which is bound to an activity that has current focus; and if the service is declared to run in the foreground.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, The basics, https://developer.android.com/guide/components/services#Basics

If the Android System stops and destroys your service it will restart it as soon as it can (although you can override this behaviour with a return value from `onStartCommand()`). You must therefore gracefully handle stops and destroys with the concomitant restart.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, The basics, https://developer.android.com/guide/components/services#Basics

Services, like other app components, must be declared in AndroidManifest.xml

```
<manifest ... >
  ...
  <application ... >
      <service android:name="com.example.ExampleService" />
      ...
  </application>
</manifest>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, The basics, https://developer.android.com/guide/components/services#Basics

Security settings:

- Always use an explicit intent when starting or binding your Service.
- Additionaly, you can ensure your Service is only available to your app with an `android:exported="false"` attribute for the `<service ... />` element.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, The basics, https://developer.android.com/guide/components/services#Basics

User feedback is often achieved through toast notifications or status bar notifications.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, Sending notifications to the user, https://developer.android.com/guide/components/services#Notifications

Started Services

The Two Options

There are two ways to create a Started Service, by subclassing:

- The Service class.
- The base class for all Services.
- If you use this class directly you ought create a new thread in the Service so the main thread is not blocked.
- More complex but can perform multi-threading (processing simultaneously).
- The IntentService class.
- This is a subclass of Service.
- Handles all requests one at a time.
- Use this class unless you need to execute multi-threading (processing simultaneously).
- Simpler class.

Started Service, by Service class

Started Service, by Service Class example.

```
public class HelloService extends Service {

	private Looper mServiceLooper;
	private ServiceHandler mServiceHandler;

	private final class ServiceHandler extends Handler {
		public ServiceHandler(Looper looper) {
			super(looper);
		}

		@Override
		public void handleMessage(Message message) {
			// Normally work is done, but just sleep for 5 seconds in
this
			// example
			long endTime = System.currentTimeMillis() + 5 * 1000;
			while (System.currentTimeMillis() < endTime) {
				synchronized (this) {
					try {
						wait(endTime -
System.currentTimeMillis());
					} catch (Exception e) {

					}
				}
			}

			// Stop the service using an id, so we don't stop the
service in the
			// middle
			// of another job
			stopSelf(message.arg1);
		}
	}

	@Override
	public void onCreate() {
		// We start up a new thread so as to not block the main thread.
		// We also make it background priority so our work will not disrupt
our
		// UI.
		HandlerThread thread = new HandlerThread("ServiceStartArguments",
				Process.THREAD_PRIORITY_BACKGROUND);
		thread.start();

		// Get the HandlerThead's Looper and use it for our Handler
		mServiceLooper = thread.getLooper();
		mServiceHandler = new ServiceHandler(mServiceLooper);
	}

	@Override
```

```
	public int onStartCommand(Intent intent, int flags, int startId) {
			String text = String.format("Service starting %d", startId);
			Toast.makeText(this, text, Toast.LENGTH_SHORT).show();

			// We use a startID to keep track of jobs.
			Message message = mServiceHandler.obtainMessage();
			message.arg1 = startId;
			mServiceHandler.sendMessage(message);

			// If we get killed, after returning from here, restart.
			return START_STICKY;
	}

	@Override
	public IBinder onBind(Intent intent) {
			// We don't allow binding (the service is a "Started" service), so we
			// must
			// return null.
			return null;
	}

	@Override
	public void onDestroy() {
			Toast.makeText(this, "Stopping Service", Toast.LENGTH_SHORT).show();
	}
}

// Client Code (from an activity)
public void goHelloService(View view) {
	// An explicit intent
	Intent intent = new Intent(this, HelloService.class);
	startService(intent);
}
```

This example is does not (quite) handle multiple requests simultaneously but could be changed to do so {*fix: and so do this}

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, Creating a started service, Extending the Service class, https://developer.android.com/guide/components/services#ExtendingService

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\HelloService.java

Started Service, By IntentService (More common)

Started Service, By IntentService example.

```
public class HelloIntentService extends IntentService {

	/**
	 * This calls the IntentService(String name) constructor with a name for the
	 * worker thread.
	 */
	public HelloIntentService() {
			super("HelloIntentService");
	}

	/**
	 * The IntentService calls this method from the default worker thread with
	 * the intent that started the service. When this method returns,
	 * IntentService stops the service, as appropriate.
	 */
	@Override
	protected void onHandleIntent(Intent intent) {
			// Normally work is done, but just sleep for 5 seconds in this
			// example
			long endTime = System.currentTimeMillis() + 5 * 1000;
			while (System.currentTimeMillis() < endTime) {
					synchronized (this) {
							try {
```

```
                                                        wait(endTime -
System.currentTimeMillis());
                                                } catch (Exception e) {

                                                }
                                        }
                                }
                                // For an IntentService the service is stopped after all start
requests
                                // have been handled, so you never have to call stopSelf().

                }
}

// Client Code
public void goHelloIntentService(View view) {
                // An explicit intent
                Intent intent = new Intent(this, HelloIntentService.class);
                startService(intent);
}

// Manifest
...
        <service
            android:name="au.com.softmake.myfirstapp.HelloIntentService"
            android:exported="false" />
    </application>

</manifest>
```

Note:

The service is stopped after all requests have been handles, so you don't need to call "stopSelf()".

Provides default implementations of onBind(), onStartCommand.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, Creating a started service, Extending the Service class, https://developer.android.com/guide/components/services#ExtendingService

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\HelloIntentService.java

The default implementation of onStartCommand(), for an IntentService, creates a worker thread that executes all intents separately from your application's main thread.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, Creating a started service, Extending the Service class, https://developer.android.com/guide/components/services#ExtendingService

For IntentService Subclasses if you override default callback methods ensure you pass execution to the base class methods

```
public class HelloIntentService extends IntentService {

  ...

          @Override
          public int onStartCommand(Intent intent, int flags, int startId) {
                      String text = String.format("Service starting %d", startId);
                      Toast.makeText(this, text, Toast.LENGTH_SHORT).show();
                      return super.onStartCommand(intent, flags, startId);
          }


          @Override
          public void onDestroy() {
                      Toast.makeText(this, "Stopping Service", Toast.LENGTH_SHORT).show();
```

```
                        super.onDestroy();
            }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, Creating a started service, Extending the Service class, https://developer.android.com/guide/components/services#ExtendingService

Foreground Service

A foreground service is a service registered as an "ongoing" status bar notification. By doing so the service resists being killed when the system is low on resources. Such a service is suitable when the user is, or wishes to be, actively aware of the service.

For example a music player playing a track ought use a foreground service and perhaps offer a chance for the user to open the music player app from the status bar notification.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Foreground services, https://developer.android.com/guide/components/foreground-services

To create a foreground service you:

- Create a service.
- Create a notification within the service.
- Within `onStartCommand` call `startForeground`(), passing it your notification.
- Optionally (but recommended) when doing work with your service update the notification and the progress indicator to provide feedback for the user.
- Call your service (from another activity) with ().

```
// Create a service
public class HelloIntentService extends IntentService {

            // An Id allows you to update the notification later.
            int mNotificationID = 66;

            // A module level variable facilitates updating
            NotificationCompat.Builder mBuilder = null;
...


// Create a notification within the service.
public class HelloIntentService extends IntentService {
 ...
            private Notification getCustomNotification() {
                        mBuilder = new NotificationCompat.Builder(this);
                        mBuilder.setSmallIcon(R.drawable.ic_action_flash_on);
                        mBuilder.setContentTitle("My Foreground service");
                        mBuilder.setContentText("My Foreground service at work");
                        // mBuilder.setContentInfo("12");
                        mBuilder.setNumber(14); // or use setContentInfo
                        // Dismiss the notification when the user presses it.
                        mBuilder.setAutoCancel(true);

                        // Create an explicit intent for the target activity when the
                        // notification is pressed.
                        Intent targetIntent = new Intent(this, MainActivity.class);

                        // Create a stack mBuilder object in order to:
                        // 1. Create an artificial back stack for the target activity. This
                        // ensures that navigating backwards from the target activity leads
out
                        // of your application to the home screen.
                        // 2. Get a pending intent object.
                        TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
                        // Add to the back stack the target class, not the target intent.
```

```
                        stackBuilder.addParentStack(MainActivity.class);
                        // Add the target intent.
                        stackBuilder.addNextIntent(targetIntent);
                        // Retrieve a pendingIntent Object
                        PendingIntent pendingIntent =
                                                stackBuilder.getPendingIntent(0,

            PendingIntent.FLAG_UPDATE_CURRENT);

                        // Set the pending intent for a user action, here pressing on the
                        // notification.
                        mBuilder.setContentIntent(pendingIntent);
                        return mBuilder.build();
            }
...

// Within onStartCommand call startForeground(), passing it your notification.
public class HelloIntentService extends IntentService {
 ...
            @Override
            public int onStartCommand(Intent intent, int flags, int startId) {
                        // String text = String.format("Service starting %d", startId);
                        // Toast.makeText(this, text, Toast.LENGTH_SHORT).show();

                        startForeground(mNotificationID, getCustomNotification());

                        return super.onStartCommand(intent, flags, startId);
            }
...

// Optionally (but recommended) when doing work with your service update the notification
and the progress indicator to provide feedback for the user.
public class HelloIntentService extends IntentService {
 ...
            /**
             * The IntentService calls this method from the default worker thread with
             * the intent that started the service. When this method returns,
             * IntentService stops the service, as appropriate.
             */
            @Override
            protected void onHandleIntent(Intent intent) {

                        String contentText = "";

                        if (mBuilder != null) {

                                    NotificationManager notificationManager =
                                                (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

                                    for (int i = 0; i <= 100; i += 20) {
                                                // Normally work is done, but just sleep for 5
seconds in this
                                                // example
                                                sleep(1);

                                                contentText = String.format("Notification
updated ... %d%%", i);
                                                mBuilder.setContentText(contentText);
                                                // Determinate Progress Indicator
                                                mBuilder.setProgress(100, i, false);
                                                // Indeterminate Progress Indicator
                                                // mBuilder.setProgress(0, 0, true);
                                                notificationManager.notify(mNotificationID,
mBuilder.build());

                                    } // for (int i ...

                        } else {
                                    Toast.makeText(getApplicationContext(),
                                                            "Send a notification before
updating it",
                                                            Toast.LENGTH_SHORT).show();
                        }

                        // Stops the service after all start requests have been handled,
                        // so you never have to call stopSelf().
```

```
        }

// Call your service (from another activity).
public class MainActivity extends Activity {
...
        public void goStartForegroundService(View view) {
                // An explicit intent
                Intent intent = new Intent(this, HelloIntentService.class);
                startService(intent);
        }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Foreground services, https://developer.android.com/guide/components/foreground-services

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\HelloIntentService.java

Service Lifecycle callbacks

All the lifecycle callbacks for a service include (depending on whether it is a Started Service, Bound Service, or Both) ...

```
public class ExampleService extends Service {
    int mStartMode;       // indicates how to behave if the service is killed
    IBinder mBinder;      // interface for clients that bind
    boolean mAllowRebind; // indicates whether onRebind should be used

    @Override
    public void onCreate() {
        // The service is being created
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // The service is starting, due to a call to startService()
        return mStartMode;
    }
    @Override
    public IBinder onBind(Intent intent) {
        // A client is binding to the service with bindService()
        return mBinder;
    }
    @Override
    public boolean onUnbind(Intent intent) {
        // All clients have unbound with unbindService()
        return mAllowRebind;
    }
    @Override
    public void onRebind(Intent intent) {
        // A client is binding to the service with bindService(),
        // after onUnbind() has already been called
    }
    @Override
    public void onDestroy() {
        // The service is no longer used and is being destroyed
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Overview, Managing the lifecycle of a service, https://developer.android.com/guide/components/services#Lifecycle

Bound Services

Intro

A bound service typically lives only as long as it is serving a client application component.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Bound servcies, https://developer.android.com/guide/components/bound-services

Bound Service by extending a binder class

Creating a bound service by extending a binder class entails:

- From your Service (Server):
- Inherit a `Service` class.
- Define an `IBinder`() interface, for example by extending a `Binder` class within the `Service` class. This can define the working methods your client can call. You have three options when doing this. Make the Binder class:
- Contain public methods the client can call, or
- Return the current service instance, which has public methods the client can call.
- Returns an instance of another class hosted by the service with public methods the client can call.
- Create an instance of the Binder class (as an `IBinder`) for clients to use.
- Implement the `onBind()` method of the `Service` class, to return the Binder class (as an `IBinder`) to the client.
- Implement some public classes that does work.
- Declare your service in the manifest. See Common Coding Considerations .

```
// Inherit a Service class.
public class BoundServiceDemo extends Service {

          // Define an IBinder() interface, done here by extending a Binder class
          // within the Service class.
          public class LocalBinder extends Binder {
                    BoundServiceDemo getService() {
                              return BoundServiceDemo.this;
                    }
          }

          // Create an instance of the Binder class (as an IBinder) for clients to
          // use. Here this is done by returning the current service instance, which
          // has public methods the client can call.
          private final IBinder mBinder = new LocalBinder();

          // Implement the onBind() method of the Service class, to return the Binder
          // class (as an IBinder) to the client.
          @Override
          public IBinder onBind(Intent intent) {
                    au.com.softmake.standardapplibrary.Log.i("onBind for
BoundServiceDemo",
                                                  this);
                    return mBinder;
          }

          // Implement some public classes that does work.
          private final Random mGenerator = new Random();

          /** Method for clients */
          public int getRandomNumber() {
                    return mGenerator.nextInt(100);
          }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Bound servcies, Creating a bound service, https://developer.android.com/guide/components/bound-services#Creating

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\BoundServiceDemo.java

From your Client:

- Implement a `ServiceConnection` in order to receive the Binder class (as an `IBinder`), returned to the client by the system.
- Bind to the Service class by calling `bindService()`, passing the `ServiceConenction`.
- Initiate, from the client, some work to be done. If the work might hang, then you should request the service to do work in a separate thread.
- Ubind from the Service.

```
public class BoundServiceClientActivity extends Activity {

	BoundServiceDemo mService;
	boolean mBound = false;

	// Implement a ServiceConnection in order to receive the Binder class (as
	// an IBinder), returned to the client by the system.
	private class MyServiceConnection implements ServiceConnection {

		@Override
		public void onServiceConnected(ComponentName className, IBinder
service) {
			au.com.softmake.standardapplibrary.Log.i(
								"onServiceConnected Start",
getApplicationContext());
			// We've bound to LocalService, cast the IBinder and get
			// LocalService instance
			LocalBinder binder = (LocalBinder) service;
			mService = binder.getService();
			mBound = true;
		}

		@Override
		public void onServiceDisconnected(ComponentName arg0) {
			mBound = false;
		}
	};

	private MyServiceConnection mConnection = new MyServiceConnection();

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

	@Override
	protected void onStart() {
		super.onStart();
		boolean succesfullBind = false;
		// Bind to the Service class by calling bindService(), passing the
		// ServiceConenction.
		Intent intent = new Intent(this, BoundServiceDemo.class);
		succesfullBind =
					bindService(intent, mConnection,
Context.BIND_AUTO_CREATE);
		String message =
					String.format("onStart End. Success? %b",
succesfullBind);
		au.com.softmake.standardapplibrary.Log.i(message, this);
	}

	/** Called from a button click in the activity UI */
	public void doWork(View view) {
		au.com.softmake.standardapplibrary.Log.i("doWork Begin", this);
		if (mBound) {
			// Call a method from the LocalService.
```

```
                                    // However, if this call were something that might hang,
then this
                                    // request should occur in a separate thread to avoid
slowing down
                                    // the activity performance.
                                    int number = mService.getRandomNumber();
                                    Toast.makeText(this, "number: " + number,
Toast.LENGTH_SHORT)
                                                                        .show();
                        }
            }

            @Override
            protected void onStop() {
                        super.onStop();
                        // Unbind from the Service
                        if (mBound) {
                                    unbindService(mConnection);
                                    mBound = false;
                        }
            }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Bound servcies, Creating a bound service, https://developer.android.com/guide/components/bound-services#Creating

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\BoundServiceDemo.java

To define an `IBinder`() interface (as in step 2 above for the Service(Server)). There are three ways to do this:

- Within your `Service` class extend the Binder class. Return the `IBinder` interface from `onBind`(). Do this if your Client and Service will run in the same application and in the same process.
- Within your `Service` create an `IBinder` interface with a `Messenger`. The service also must define a `Handler` that handles messages from the client. The client can also define a `Messenger` to receive messages from the Service. Do this if you need simple Interprocess Communication (IPC). This technique handles Messages in a single thread, so you don't have to design your service to be thread safe.
- Use the Android Interface Definition Language (AIDL). Do this if you need Interprocess Communication (IPC) that can multithread. Your service will need to be designed to handle multi-threading.

Bound Service by using a Messenger

Creating a bound service by using a Messenger entails:

From your Service (Server):

- Inherit a `Service` class.
- Within your `Service` class define messages using constants.
- Within your `Service` class extend a `Handler` class as an IncomingHandler for the Service, and override the `handleMessage()` method to receive messages from your client.
- Instantiate a `Messenger` from the `IncomingHandler()`.
- Within the Service's onBind() return a `IBinder` from the `Messanger`, for the clients to use.
- Declare your service in the manifest. See Common Coding Considerations .

```
// Inherit a Service class.
public class BoundServiceMessengerDemo extends Service {

	// Within your Service class define messages using constants.
	static final int MESSAGE_SAY_HELLO = 1;

	// Within your Service class extend a Handler as an IncomingHandler for
	// the Service, and override the handleMessage() method to receive messages
	// from your client.
	class IncomingHandler extends Handler {
			@Override
			public void handleMessage(Message msg) {
				switch (msg.what) {
				case MESSAGE_SAY_HELLO:
					Toast.makeText(getApplicationContext(),
"hello!",

	Toast.LENGTH_SHORT).show();
					break;
				default:
					super.handleMessage(msg);
				} // switch
			} // handleMessage
	} // class IncomingHandler

	// Instantiate a Messenger from the IncomingHandler().
	final Messenger mMessenger = new Messenger(new IncomingHandler());

	// Within the Service's onBind() return a IBinder from the Messanger, for
	// the clients to use.
	@Override
	public IBinder onBind(Intent intent) {
			// Within the Service's onBind() return a IBinder from the Messanger,
			// for the clients to use.
			Toast.makeText(getApplicationContext(), "binding!",
					Toast.LENGTH_SHORT).show();
			return mMessenger.getBinder();
	}

	@Override
	public boolean onUnbind(Intent intent) {
			Toast.makeText(getApplicationContext(), "unbinding!",
					Toast.LENGTH_SHORT).show();
			return super.onUnbind(intent);
	}
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Bound servcies, Creating a bound service, Using a Messenger, https://developer.android.com/guide/components/bound-services#Messenger

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\BoundServiceMessengerDemo.java

From your Client:

- Implement a `ServiceConnection` in order to receive the Binder class (as an `IBinder`), returned to the client by the system.
- When receiving the Binder class, in onServiceConnected, use this to instantiate a `Messenger`.
- Bind to the Service class by calling `bindService()`, passing the `ServiceConenction`.
- Initiate, from the client, some work to be done. To do so create a message, using a message defined in your `Service` class and then use the `Messenger` instance to send this message. If the work might hang, then you should request the service to do work in a separate thread.

Unbind from the Service.

```
public class BoundServiceMessengerClientActivity extends Activity {

    Messenger mMessenger = null;
    boolean mBound = false;

    // Implement a ServiceConnection in order to receive the Binder class (as an
    // IBinder), returned to the client by the system.
    // Here we implement the ServiceConnection as an anonymous class.
    private ServiceConnection mConnection = new ServiceConnection() {

        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            // When receiving the Binder class, in onServiceConnected, use this
            // to instantiate a Messenger.
            mMessenger = new Messenger(service);
            mBound = true;
        }

        @Override
        public void onServiceDisconnected(ComponentName name) {
            mMessenger = null;
            mBound = false;
        }
    };

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

    @Override
    protected void onStart() {
        super.onStart();
        // Bind to the Service class by calling bindService(), passing the
        // ServiceConnection.
        Intent intent = new Intent(this, BoundServiceMessengerDemo.class);
        bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
    }

    @Override
    protected void onStop() {
        super.onStop();
        // Unbind from the Service.
        if (mBound) {
            unbindService(mConnection);
            mBound = false;
        }
    }

    public void doWork(View view) {
        if (!mBound) return;

        // Initiate, from the client, some work to be done. To do so create a
        // message, using a message defined in your Service class and then use
        // the Messenger instance to send this message. If the work might hang,
        // then you should request the service to do work in a separate thread.
```

```
        Message message = Message.obtain(
                null, BoundServiceMessengerDemo.MESSAGE_SAY_HELLO);
        try {
            mMessenger.send(message);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Bound servcies, Creating a bound service, Using a Messenger, https://developer.android.com/guide/components/bound-services#Messenger

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\BoundServiceMessengerDemo.java

Notice that this example does not show how the service can respond to the client. If you want the service to respond, then you need to also create a Messenger in the client. For details see the documentation. [TODO: implement such an example]

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Bound servcies, Creating a bound service, Using a Messenger, https://developer.android.com/guide/components/bound-services#Messenger

Binding Process between client and server

More about the binding process between an client and server (when using a bound service), as exemplified above, can be found in the documentation under "Binding to a Service".

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Bound servcies, Creating a bound service, Binding to a service, https://developer.android.com/guide/components/bound-services#Binding

Services with AIDL

Creating services with Android Interface Definition Language (AIDL) will be necessary for interprocess communication, if you wish to:

- Make your service available to other applications; or
- Handle multithreading in your service.
- Otherwise, if you want to do interprocess communication but do not need to handle multithreading, then use a Bound Service with messengers.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Services, Android Interface Definition Language (AIDL), https://developer.android.com/guide/components/aidl

Data

The options for reading and writing data include:

- Key-Value sets;
- Files;
- SQLite Databases;
- Network operations to store data on your own web-services.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

Key-Value Sets (SharedPreferences API)

SharedPreferences Basics

To read and write sets of key-value pairs to a file use the `SharedPreferences` API.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save key-value data, https://developer.android.com/training/data-storage/shared-preferences

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\SharedPreferencesDemo.java

The `SharePreferences` API is different from the `Preference` API. The `Preference` API helps you build the UI for your app settings, although in turn, the `Preference` API uses the `SharedPreferences` API to save settings.

Instiate a SharedPreferences object.

```
SharedPreferences sharedPreferences = this.getPreferences(Context.MODE_PRIVATE);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save key-value data, https://developer.android.com/training/data-storage/shared-preferences

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\SharedPreferencesDemo.java

Use a SharedPreferences.Editor to make writes.

```
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putString("editText01", editText01.getText().toString());
editor.putString("editText02", editText02.getText().toString());
editor.commit();
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save key-value data, https://developer.android.com/training/data-storage/shared-preferences

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\SharedPreferencesDemo.java

SharedPreferences reads

```
SharedPreferences sharedPreferences = this.getPreferences(Context.MODE_PRIVATE);
String defaultValue = "Data not yet saved";
// defaultValue is returned if the SharedPreference does not exist.
```

```
String string01 = sharedPreferences.getString("editText01", defaultValue);
String string02 = sharedPreferences.getString("editText02", defaultValue);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save key-value data, https://developer.android.com/training/data-storage/shared-preferences

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\SharedPreferencesDemo.java

SharedPreferences Variations

To instantiate shared preferences from a fragment use getActivity()

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save key-value data, https://developer.android.com/training/data-storage/shared-preferences

To make a SharedPreferences accessible to the world, rather than your own app, use MODE_WORLD_READABLE or MODE_WORLD_WRITEABLE when instantiating your SharedPreference object.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save key-value data, https://developer.android.com/training/data-storage/shared-preferences

If you need multiple SharedPreferences files you can identify then by name and do the following (using getSharedPreferences() rather than getPreferences()) ...

```
<!-- In res/values/strings.xml -->
<string name="preference_file_key_01">com.example.myapp.PREFERENCE_FILE_KEY_01</string>
<string name="preference_file_key_02">com.example.myapp.PREFERENCE_FILE_KEY_02</string>

// In code
Context context = getActivity(); // When context is a fragment
SharedPreferences sharedPreferences02 = context.getSharedPreferences(
        getString(R.string.preference_file_key_02), Context.MODE_PRIVATE);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save key-value data, https://developer.android.com/training/data-storage/shared-preferences

Files (app external)

Files Intro

Directory types.

Nominal Storage Location	Type	Code	Notes
Internal	Regular	getFilesDir()	• It's always available. • Files saved here are accessible by only your app by default. • When the user uninstalls your app, the system removes all your app's files from internal storage.
	Temporary	getCacheDir()	• Be sure to delete each file once it is no longer needed and implement a reasonable size limit for the amount

			of memory you use at any given time, such as 1MB
External	Public	getExternalStoragePublicDirectory()	• It's not always available, because the user can mount the external storage as USB storage and in some cases remove it from the device. • It's world-readable, so files saved here may be read outside of your control. • When the user uninstalls your app, the system **doesn't** remove your app's files from here
	Private	getExternalFilesDir()	• It's not always available, because the user can mount the external storage as USB storage and in some cases remove it from the device. • It's world-readable, so files saved here may be read outside of your control. • When the user uninstalls your app, the system **does** remove your app's files from here.
	Temporary	getExternalCacheDir()	•

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

Directory locations within the Android file system.

- getFilesDir(): /data/data/com.example.myfirstapp/files/filename.ext
- getCacheDir(): /data/data/com.example.myfirstapp/cache/filename-DDDDDDDDDD.tmp
- getExternalStoragePublicDirectory(): E.g. /storage/sdcard0/Pictures/MyStuff/
- getExternalFilesDir(): E.g. /storage/sdcard0/Android/data/au.com.softmake.myfirstapp/files/Pictures/MyStuff/

The nominal storage location designations, "internal" and "external", are a historical legacy. Originally these terms referenced permanent and removable storage, respectively. However, some devices these days might partition permanent storage into "internal" and "external" parts.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

Best practice: internal storage to prevent the user and other apps from directly accessing your files; external storage for files not needing access restrictions to the user or other apps.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

Files suitable for storing on external storage include:

- Public files, files that should be available to the user and other apps even after the generating app has been uninstalled; (E.g. photos captured by your app)
- Private files, files that although are available to other apps are of no security consequence to be so, and that take up a lot of room (these files are deleted when the user uninstalls the app). (E.g. temporary media files).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

Permissions

For reading and writing to internal storage you don't need permissions (your app automatically has it).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, Permissions and access to external storage, https://developer.android.com/training/data-storage#permissions

For reading and writing to external storage set READ_EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE in AndroidManifest.xml.

```
<manifest ...>

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
...
```

Currently you don't need permission to read external storage but this will change in the future.

Also, if you explicitly grant write permssions, then read permissions will be implicitly granted. However, it is good practice to be explicit.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, Permissions and access to external storage, https://developer.android.com/training/data-storage#permissions

Query free space

Check how much free space is remaining on the partition on which the file object lives.

```
          File file = File.createTempFile(fileName, null, this.getCacheDir());
  ...

          // getFreeSpace() returns bytes.
          long freeSpaceInMB =  file.getFreeSpace() / (1024 * 1024);
          String message = String.format("File %s saved. Free space: %d MB", fileName,
freeSpaceInMB );
```

You can also use getTotalSpace(). If you don't know how much space the file writing will take ahead of time, you can just attempt the write and catch an IOException instead.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp\src\au\com\softmake\myfirstapp MainActivity.java

Saving to storage, common considerations.

When saving to an internal drive the stream you must use is a FileOutputStream. When saving to an external drive the stream you must not use a FileOutputStream, instead use a FileWriter.

```
// Internal
          FileOutputStream fileOutputStream = null;
          PrintWriter outputStream = null;
          File file = File.createTempFile(fileName, null, this.getCacheDir());

          try {
                    // A fileOutputStream is necessary for internal storage.
                    fileOutputStream =
                                                openFileOutput(file.getName(),
Context.MODE_PRIVATE);
                    outputStream = new PrintWriter(fileOutputStream);

// External
  FileWriter fileWriter = null;
          PrintWriter outputStream = null;
          File file =
                                    new File(getExternalAlbumStorageDir("MyStuff", true),
fileName);

          try {
                    // A FileOutputStream doesn't work with external storage,
                    // use a FileWriter instead.
                    fileWriter = new FileWriter(file);
                    outputStream = new PrintWriter(fileWriter);
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp\src\au\com\softmake\myfirstapp MainActivity.java

Save to Internal Storage

Use Root Explorer (Google Play App) to examine files (file explorers with normal permissions will not be able to access your private files).

There are two methods to write to the internal directory returned by getFilesDir(): create a FileOutputStream (using openFileOutput); and by calling getFilesDir() directly.

Write to the internal directory (which would be returned by getFilesDir()) by creating a FileOutputStream (using openFileOutput).

```
public void goSaveFileInternal(View view) throws IOException {
    String fileName = "JohnFile6.txt";
    String output =
            "Now is the time for all good men www...\n"
                    + "To catch a pale of water\n" + "And then some";

    FileOutputStream fileOutputStream = null;
    PrintWriter outputStream = null;
    File file = new File(this.getFilesDir(), fileName);
    // File file = File.createTempFile(fileName, null, this.getCacheDir());

    try {
        // A fileOutputStream is necessary for internal storage.
        fileOutputStream =
                openFileOutput(file.getName(), Context.MODE_PRIVATE);
        outputStream = new PrintWriter(fileOutputStream);
        outputStream.println(output);
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (outputStream != null) {
            outputStream.close();
        }
```

```
    }

    // getFreeSpace() returns bytes.
    long freeSpaceInMB = file.getFreeSpace() / (1024 * 1024);
    String message =
            String.format("File %s saved. Free space: %d MB", fileName,
                    freeSpaceInMB);
    Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT)
            .show();
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp\src\au\com\softmake\myfirstapp MainActivity.java

Write to internal directory by calling getFilesDir() directly.

```
          ...
  String fileName = "JohnFile2.txt";
  File file = new File(this.getFilesDir(), fileName);

          try {
                      // Wrap the FileOutputStream
                      outputStream = new PrintWriter(openFileOutput(file.getName(),
Context.MODE_PRIVATE));
  ...
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\soft make\myfirstapp\MainActivity.java

Write to internal temporary cache files by calling getCacheDir() directly.

```
  String fileName = "JohnFile2.txt";
  File file = File.createTempFile(fileName, null, this.getCacheDir());

          try {
                      // Wrap the FileOutputStream
                      outputStream = new PrintWriter(openFileOutput(file.getName(),
Context.MODE_PRIVATE));
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\soft make\myfirstapp\MainActivity.java

Take responsibility for deleting temporary cache files you create. The system might delete your cache files if the system begins to run low on storage.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

Save to Internal Storage, custom sub directory

Save to a internal storage with a custom sub directory as follows.

```
private void saveFileInternal(String fileName, String output) throws IOException {
    String subDirName = "coolFilesDir";
```

```
    FileOutputStream fileOutputStream = null;
    PrintWriter outputStream = null;
    String path = this.getFilesDir().getAbsolutePath() +  "/" + subDirName;
    File dirFile = new File(path);
    File fileFile = new File(dirFile, fileName);
    // File file = File.createTempFile(fileName, null, this.getCacheDir());

    try {
        if (!dirFile.exists()) {
            // Creates the directory named by this file,
            // creating missing parent directories if necessary.
            dirFile.mkdirs();
        }
        //        fileOutputStream =
        //                openFileOutput(fileFile.getName(), Context.MODE_PRIVATE);

        fileOutputStream = new FileOutputStream(fileFile);
        outputStream = new PrintWriter(fileOutputStream);
        outputStream.println(output);
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (outputStream != null) {
            outputStream.close();
        }
    }

    // getFreeSpace() returns bytes.
    long freeSpaceInMB = fileFile.getFreeSpace() / (1024 * 1024);
    String message = String.format("File %s saved. Free space: %d MB\n",
                                   fileFile.getAbsolutePath(), freeSpaceInMB);
    Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).show();
    TextView outputTextView = (TextView) getActivity().findViewById(
        R.id.textView_output);
    outputTextView.setText(outputTextView.getText() + message);
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\sharefiles\ShareFilesDemoActivity.javas

Save to External Storage

First check that the external storage is available to be written to or read from.

```
public static boolean isExternalStorageWritableAndReadable() {
          String state = Environment.getExternalStorageState();
          return Environment.MEDIA_MOUNTED.equals(state);
}

public static boolean isExternalStorageReadable() {
          String state = Environment.getExternalStorageState();
          return (Environment.MEDIA_MOUNTED.equals(state) ||
                                Environment.MEDIA_MOUNTED_READ_ONLY.equals(state));
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

C:\Users\John\Documents\Sda\Code\Android\Libraries\StandardAppLibrary\src\au\com\softmake\standardapplibrary\Files.java

Save to external storage, the public or private space, as follows ...

```
private File getExternalAlbumStorageDir(String albumName,
                      boolean isPublicDir) {

          File file = null;
          String standardDirectory = Environment.DIRECTORY_PICTURES;
          if (isPublicDir) {
```

```
                        // External Public Directory
                        file =
                                                new File(
                                                                        Environment

            .getExternalStoragePublicDirectory(standardDirectory),
                                                                        albumName);
            } else {
                        // External Private Directory
                        file =
                                                new
File(this.getExternalFilesDir(standardDirectory),
                                                                        albumName);
            }

            String message = "";
            // Returns false if the directory already exists. Use isDirectory for
            // checking instead.
            file.mkdirs();
            if ((file.isDirectory())) {
                        message =
                                                String.format("Directory %s created or already
exists.",

            file.getAbsolutePath());
            } else {
                        message =
                                                String.format(
                                                                        "Directory %s not
created and does not exist.",

            file.getAbsolutePath());
            }

            Log.i(message, this);
            Toast.makeText(getApplicationContext(), message, Toast.LENGTH_LONG)
                                    .show();

            return file;
}

public void goSaveFileExternal(View view) throws IOException {
            String fileName = "JohnFil8.txt";
            String output = "Now is the time for all good men...";

            FileWriter fileWriter = null;
            PrintWriter outputStream = null;
            File file =
                                    new File(getExternalAlbumStorageDir("MyStuff", false),
fileName);

            try {
                        // A FileOutputStream doesn't work with external storage,
                        // use a FileWriter instead.
                        fileWriter = new FileWriter(file);
                        outputStream = new PrintWriter(fileWriter);
                        outputStream.println(output);
            } catch (IOException e) {
                        e.printStackTrace();
            } finally {
                        if (outputStream != null) {
                                    outputStream.close();
                        }
            }

            // getFreeSpace() returns bytes.
            long freeSpaceInMB = file.getFreeSpace() / (1024 * 1024);
            String message =
                                    String.format("File %s saved. Free space: %d MB",
fileName,
                                                                        freeSpaceInMB);
            Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT)
                                    .show();
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\MainActivity.java

Saving to the real, removable SD card

"External Storage" on many devices might reference a partition on non removable storage rather, that is, than an SD card. For devices that have such a partition and a real, removable, SD card the former type of external storage is known as "primrary", the later, "secondary". Only after KitKat 4.4 can you formally access the secondary external device, on a device that also has a non removable primary external partition. To loop through all the kinds of external storage do the following ...

```
private String getRemovableExternalStorageDir(String dirName) {
          File[] directories =

          ContextCompat.getExternalFilesDirs(getApplicationContext(),
                                                        null);
          String message = "";
          String line = "";
          for (File dir : directories) {
                    line = dir.getAbsolutePath();
                    au.com.softmake.standardapplibrary.Log.i(line, this);
                    message += line;
          }
          Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT)
                              .show();
          return "todo";
}
```

On android versions 4.3 and below the loop will only return the primary external storage.

(Google API Guides, 2013. Develop > API Guides) https://developer.android.com/guide/topics/data/data-storage.html#filesExternal
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\MainActivity.java

Read a file

To read a file you could use a FileReader wrapped by a BufferedReader for line operations.

```
public void goReadFile(View view) throws IOException {
          String fileName = "JohnFile6.txt";

          BufferedReader inputStream = null;
          File file = new File(this.getFilesDir(), fileName);
          String message = "";

          try {

                    inputStream = new BufferedReader(new FileReader(file));
                    String line = "";
                    while ((line = inputStream.readLine()) != null) {
                      message += line + "\n";
                    }

          } catch (IOException e) {
                    e.printStackTrace();
          } finally {
                    if (inputStream != null) {
                              inputStream.close();
                    }
          }
```

```
            au.com.softmake.standardapplibrary.Log.i(message, this);;
            Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT)
                                .show();
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\MainActivity.java

Delete a file

Regularly delete files saved in cache directories and any other files you no longer need.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

Delete a file as follows

```
public void goDeleteFile(View view) {
            String fileName = "JohnFile6.txt";

            File file = new File(this.getFilesDir(), fileName);
            file.delete();

            // Also myContext.deleteFile(fileName);
            String message = String.format("File %s deleted.", fileName);
            Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT)
                                .show();
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Storage overview, https://developer.android.com/training/data-storage

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\MainActivity.java

Databases

Setup Environment for SQLite Access

Todo: https://developer.android.com/training/data-storage/room/sqlite-room-migration

The location of a database created from your app is:
/data/data/com.example.myfirstapp/databases/DatabaseName.db

- You need a third party SQLite Manager for viewing any SQLite databases you create from your app.
- From your device: Use a Google Play SQLite App. E.g. SQLite Debugger
- From your PC:
- Install SQLite Database Browser or Firefox SQLite Manager Plugin
- Open Eclipse and your emulator.
- Goto the DDMS perspective.
- Devices View > Click on your Emulator (or Device)
- File Explorer View > /data/data/com.example.myfirstapp/databases/DatabaseName.db
- Click the Pull a file from the device button and save the database file on your computer.
- Open the file in your SQLite Manager of choice.

(Theah 2011, "How to View the SQLite Database on Your Android Emulator", http://blog.kwyps.com/2011/07/how-to-view-sqlite-database-on-your.html) http://blog.kwyps.com/2011/07/how-to-view-sqlite-database-on-your.html

You can pull your database from an emulator or device to a location on your PC via the command line. This will be useful when debugging your database.

```
C:\Users\John>adb devices
List of devices attached
192.168.1.6:5555        device
emulator-5554   device

C:\Users\John>adb -s emulator-5554 pull /data/data/au.com.softmake.myfirstapp/da
tabases/Northwind.db C:\Users\John\Documents\zTemp
```

(Google 2020, "Android Developers > Android Studio > User Guide > Android Debug Bridge (Adb)", https://developer.android.com/studio/command-line/adb) http://developer.android.com/tools/help/adb.html#shellcommands

You can also use shell commands to examin a database.

```
C:\Users\John>adb -e shell
root@generic_x86:/ # sqlite3 /data/data/au.com.softmake.myfirstapp/databases/Nor
thwind.db
tmake.myfirstapp/databases/Northwind.db                                    <
SQLite version 3.7.11 2012-03-20 11:35:50
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .dump
.dump
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE android_metadata (locale TEXT);
INSERT INTO "android_metadata" VALUES('es');
CREATE TABLE Customers (_id INTEGER PRIMARY KEY AUTOINCREMENT, Company TEXT,Last
Name TEXT,FirstName TEXT,EmailAddress TEXT,JobTitle TEXT );
INSERT INTO Customers VALUES(1,'Cool Co','Blow','Joe','JoeBlow@email.com','Boss'
);
INSERT INTO Customers VALUES(2,'Cool Co','Blow','Joe','JoeBlow@email.com','Boss'
);
INSERT INTO Customers VALUES(3,'Cool Co','Blow','Joe','JoeBlow@email.com','Boss'
);
INSERT INTO Customers VALUES(4,'Cool Co','Blow','Joe','JoeBlow@email.com','Boss'
);
INSERT INTO Customers VALUES(5,'Hot Co','brazilla','filla','JoeBlow@email.com','
Boss');
INSERT INTO Customers VALUES(6,'Cool Co','Blow','Joe','JoeBlow@email.com','Boss'
);
DELETE FROM sqlite_sequence;
INSERT INTO "sqlite_sequence" VALUES('Customers',6);
COMMIT;
```

```
sqlite>
```

(Google 2020, "Android Developers > Android Studio > User Guide > Android Debug Bridge (Adb)", https://developer.android.com/studio/command-line/adb) http://developer.android.com/tools/help/adb.html#sqlite

Define a Database Schema within a Contract Class and DBHelper

One design pattern is to define your database schema with a "Contract" class and "DBHelper" class.

A Contract class is used to define the table and column names.

```
public final class NorthwindContract {

	// To prevent someone from accidentally instantiating the contract class,
	// give it an empty constructor.
	public NorthwindContract() {
	}

	// Static nested classes do not have access to other members of the
	// enclosing class.
	// Abstract classes cannot be instantiated (but they can be extended).
	// By implementing the BaseColumns interface, your inner class can inherit a
	// primary key field called _ID that some Android classes such as cursor
	// adaptors will expect it to have..
	public static abstract class Customers implements BaseColumns {
			public static final String TABLE_NAME = "Customers";
			public static final String COLUMN_NAME_COMPANY = "Company";
			public static final String COLUMN_NAME_LAST_NAME = "LastName";
			public static final String COLUMN_NAME_FIRST_NAME = "FirstName";
			public static final String COLUMN_NAME_EMAIL_ADDRESS = "EmailAddress";
			public static final String COLUMN_NAME_JOB_TITLE = "JobTitle";
	}
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save data in a local database, https://developer.android.com/training/data-storage/room

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\db\NorthwindContract.java

A DBHelper class is used to define table column datatypes and other column attributes. The DBHelper class also handles: database creation, upgrading/downgrading (when you change the database schema), and returns a database object for use.

```
public class NorthwindDBHelper extends SQLiteOpenHelper {

	// Basics
	public static final int DATABASE_VERSION = 3;
	public static final String DATABASE_NAME = "Northwind.db";

	// Database Definition Commands
	private static final String TEXT_TYPE = " TEXT";
	private static final String COMMA_SEP = ",";
	private static final String SQL_CREATE_ENTRIES = "CREATE TABLE " +
				Customers.TABLE_NAME + " (" + Customers._ID +
				" INTEGER PRIMARY KEY AUTOINCREMENT, " +
				Customers.COLUMN_NAME_COMPANY + TEXT_TYPE + COMMA_SEP +
				Customers.COLUMN_NAME_LAST_NAME + TEXT_TYPE + COMMA_SEP +
				Customers.COLUMN_NAME_FIRST_NAME + TEXT_TYPE + COMMA_SEP +
				Customers.COLUMN_NAME_EMAIL_ADDRESS + TEXT_TYPE +
COMMA_SEP +
				Customers.COLUMN_NAME_JOB_TITLE + TEXT_TYPE + " )";
	private static final String SQL_DELETE_ENTRIES = "DROP TABLE IF EXISTS " +
				Customers.TABLE_NAME;

	public NorthwindDBHelper(Context context) {
			super(context, DATABASE_NAME, null, DATABASE_VERSION);
```

```
        }

        @Override
        public void onCreate(SQLiteDatabase db) {
                db.execSQL(SQL_CREATE_ENTRIES);
        }

        // You don't need to explicitly invoke this. Implicitly invoke this
        // merely by incrementing DATABASE_VERSION
        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
                // This database is only a cache for online data, so its upgrade
policy
                // is to simply to discard the data and start over
                db.execSQL(SQL_DELETE_ENTRIES);
                onCreate(db);
        }

        @Override
        public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
                onUpgrade(db, oldVersion, newVersion);
        }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save data in a local database, Overview, https://developer.android.com/training/data-storage/room

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save data in a local database, Save data using SQLite, https://developer.android.com/training/data-storage/sqlite

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\db\NorthwindDBHelper.java

To upgrade your database schema (after making changes to the schema in your contract class and/or DBHelper) you only need to increment DATABASE_VERSION (in DBHelper). This will implicitly invoke the onUpgrade() method in you're your DBHelper.

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\db\NorthwindDBHelper.java

To obtain a database object from client code you use your DBHelper to getReadableDatabase() or getWriteableDatabase().

```
SQLiteDatabase db = mDBHelper.getWritableDatabase();

// ... perform database operations
```

Database Operations

Insert

```
public void insert(View view) throws SQLException {
    // Access the database via the DBHelper
    SQLiteDatabase db = mDBHelper.getWritableDatabase();

    ContentValues contentValues = new ContentValues();

    // Customers._ID is AutoIncremented
    contentValues.put(Customers.COLUMN_NAME_COMPANY, "Cool Co");
    contentValues.put(Customers.COLUMN_NAME_FIRST_NAME, "Joe");
    contentValues.put(Customers.COLUMN_NAME_LAST_NAME, "Blow");
    contentValues.put(Customers.COLUMN_NAME_EMAIL_ADDRESS,
            "JoeBlow@email.com");
    contentValues.put(Customers.COLUMN_NAME_JOB_TITLE, "Boss");

    long newRowPrimaryKeyID = 0;
```

```
    try {
        newRowPrimaryKeyID =
                db.insertOrThrow(Customers.TABLE_NAME, null, contentValues);
    } finally {
        db.close();
    }

    String message =
            String.format(
                    "Write to %s successful. Inserted row with ID: %d",
                    NorthwindDBHelper.DATABASE_NAME, newRowPrimaryKeyID);
    Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT)
            .show();
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save data in a local database, Overview, https://developer.android.com/training/data-storage/room

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save data in a local database, Save data using SQLite, https://developer.android.com/training/data-storage/sqlite

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\DatabaseActivity.java

Read

```
public void read(View view) {
    SQLiteDatabase db = mDBHelper.getReadableDatabase();

    // A projection is a list of column names to return.
    String[] projection =
            { Customers._ID, Customers.COLUMN_NAME_COMPANY,
                    Customers.COLUMN_NAME_LAST_NAME,
                    Customers.COLUMN_NAME_FIRST_NAME,
                    Customers.COLUMN_NAME_EMAIL_ADDRESS,
                    Customers.COLUMN_NAME_JOB_TITLE };

    String selection = null;
    String selectionArgs[] = null;
    String groupBy = null;
    String having = null;
    String orderBy = Customers._ID + " DESC";

    Cursor cursor =
            db.query(Customers.TABLE_NAME, projection, selection,
                    selectionArgs, groupBy, having, orderBy);

    String output = "";
    if (cursor != null && cursor.moveToFirst()) {
        do {
            long ID =
                    cursor.getLong(cursor
                            .getColumnIndexOrThrow(Customers._ID));
            String company =
                    cursor.getString(cursor
                            .getColumnIndexOrThrow(Customers.COLUMN_NAME_COMPANY));
            String lastName =
                    cursor.getString(cursor
                            .getColumnIndexOrThrow(Customers.COLUMN_NAME_LAST_NAME));
            String firstName =
                    cursor.getString(cursor
                            .getColumnIndexOrThrow(Customers.COLUMN_NAME_FIRST_NAME));
            String emailAddress =
                    cursor.getString(cursor
                            .getColumnIndexOrThrow(Customers.COLUMN_NAME_EMAIL_ADDRESS));
            String jobTitle =
                    cursor.getString(cursor
                            .getColumnIndexOrThrow(Customers.COLUMN_NAME_JOB_TITLE));
            // Note any null values get show up in the string as "null"
            output +=
                    String.format("%d, %s, %s, %s, %s, %s \n", ID, company,
                            lastName, firstName, emailAddress, jobTitle);
        } while (cursor.moveToNext());
```

```
        TextView textView = (TextView) findViewById(R.id.textView_messages);
        textView.setText(output.trim());
    } else {
        Toast.makeText(getApplicationContext(),
                "Nothing returned by cursor", Toast.LENGTH_SHORT).show();
    }
    db.close();
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save data in a local database, Overview, https://developer.android.com/training/data-storage/room

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save data in a local database, Save data using SQLite, https://developer.android.com/training/data-storage/sqlite

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\DatabaseActivity.java

Delete

```
public void delete(View view) {
    SQLiteDatabase db = mDBHelper.getWritableDatabase();

    Long rowIDTOSelect = 3L;

    // The "WHERE" clause without "WHERE"
    String selection = Customers._ID + " LIKE ?";
    String[] selectionArgs = { String.valueOf(rowIDTOSelect) };

    int rowsAffected =
            db.delete(Customers.TABLE_NAME, selection, selectionArgs);
    db.close();

    String message =
            String.format(
                    "Delete operation on RowID: %d. rowsAffected: %d",
                    rowIDTOSelect, rowsAffected);
    Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT)
            .show();
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save data in a local database, Overview, https://developer.android.com/training/data-storage/room

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save data in a local database, Save data using SQLite, https://developer.android.com/training/data-storage/sqlite

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\DatabaseActivity.java

Update

```
public void update(View view) {
    SQLiteDatabase db = mDBHelper.getWritableDatabase();

    ContentValues contentValues = new ContentValues();

    contentValues.put(Customers.COLUMN_NAME_COMPANY, "Hot Co");
    contentValues.put(Customers.COLUMN_NAME_FIRST_NAME, "filla");
    contentValues.put(Customers.COLUMN_NAME_LAST_NAME, "brazilla");

    Long rowIDTOSelect = 5L;

    // The "WHERE" clause without "WHERE"
    String selection = Customers._ID + " LIKE ?";
    String[] selectionArgs = { String.valueOf(rowIDTOSelect) };

    int rowsAffected =
```

```
            db.update(Customers.TABLE_NAME, contentValues, selection,
                    selectionArgs);
    db.close();

    String message =
            String.format(
                    "Update operation on RowID: %d. rowsAffected: %d",
                    rowIDTOSelect, rowsAffected);
    Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT)
            .show();
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save data in a local database, Overview, https://developer.android.com/training/data-storage/room

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Save data in a local database, Save data using SQLite, https://developer.android.com/training/data-storage/sqlite

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\DatabaseActivity.java

Other

See also Design data storage.

Content Providers

Intro

A content provider manages access to data.

You may want to create content provider if:

- You want to allow another application to access your app's data; or
- You want to allow users to copy complex data from your app into other apps; or
- You want to provide custom search suggestions using the search framework.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, https://developer.android.com/guide/topics/providers/content-provider-creating

You **don't** need a provider to use an SQLite database if the use is entirely within your own application.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, https://developer.android.com/guide/topics/providers/content-provider-creating

A content provider exposes data as a table.

A content provider is accessed with a content resolver. A cursor is used to traverse the table.

(Google API Guides, 2013. Develop > API Guides) http://developer.android.com/guide/topics/providers/content-provider-basics.html

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, https://developer.android.com/guide/topics/providers/content-provider-basics

Consider using a Loader to access content provider data. Loaders call content resolver's indirectly. See Loaders. However, Since a CursorLoader passes the same arguments onto ContentResolver.query() learning about ContentResolver's is essential.

Content URI

A content URI identifies data in a provider. It either identifies:

- The whole provider (its "authority"),
- A table (a "path"), or
- A row.

```
UserDictionary Provider, Words table, Row with _ID "10" Content URI:

content://user_dictionary/words/10

Parts:
* Scheme "content://"
* Authority "user_dictionary"
* Path "words"
* Row ID "10"
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Overview, Content URIs, https://developer.android.com/guide/topics/providers/content-provider-basics#ContentURIs

Every data access method in a ContentProvider takes a content URI, thereby enabling data access.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Overview, Content URIs, https://developer.android.com/guide/topics/providers/content-provider-basics#ContentURIs

The authority name needs to be globally unique. Therefore the naming convention is com.example.<appname>.provider (one app usually has at most one ConentProvider, if any). That is, your app package name with ".provider"

```
// Example Authority
au.com.softmake.myfirstapp.provider
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Overview, Content URIs, https://developer.android.com/guide/topics/providers/content-provider-basics#ContentURIs

Custom provider table content URI examples

```
content://au.com.softmake.myfirstapp.provider/table1
content://au.com.softmake.myfirstapp.provider/table2
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Overview, Content URIs, https://developer.android.com/guide/topics/providers/content-provider-basics#ContentURIs

By convention, Content URIs that reference a row reference the primary key named "_ID" value of the row. A CursorAdapter requires one of the columns to be "_ID".

```
content://user_dictionary/words/10

// 10 references the row with _ID = 10.
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Overview, Content URIs, https://developer.android.com/guide/topics/providers/content-provider-basics#ContentURIs

In code, obtain Content URI references like this...

```
// Content URI for Words table of UserDictionary Provider
Uri tableContentUri = UserDictionary.Words.CONTENT_URI;

// Row Content URI
Uri rowConentUri = ContentUris.withAppendedId(UserDictionary.Words.CONTENT_URI,4);
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Overview, Content URIs, https://developer.android.com/guide/topics/providers/content-provider-basics#ContentURIs

For building and parsing Content URIs from strings use convenience methods in URI, URI.Builder, and ContentURIs classes.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Overview, Content URIs, https://developer.android.com/guide/topics/providers/content-provider-basics#ContentURIs

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Designing Content URIs, https://developer.android.com/guide/topics/providers/content-provider-creating#ContentURI

Content URI patterns

A `ContentProvider` needs to handle Content Uris. The `ContentProvider` will, moreover, recognize several diffent Content Uris directed at the `ContentProvider`. The `UriMatcher` convenience class facilitates this Content URI handling in a switch statement within the `query()` method of the `ContentProvider`. `UriMatcher`, through `addURI()`, therefore implements a Content URI pattern matching system with wildcards. Coding example ...

```
public class ExampleProvider extends ContentProvider {

    // Creates a UriMatcher object.
    private static UriMatcher sUriMatcher = new UriMatcher(0);

    /**
     * The calls to addURI() go here, for all of the content URI patterns that the
     * provider
     * should recognize. For this snippet, only the calls for table 3 are shown.
     */
    public ExampleProvider() {
         /*
         * Sets the integer value for multiple rows in table 3 to 1. Notice that no
         * wildcard is used
         * in the path
         */
        sUriMatcher.addURI("com.example.app.provider", "table3", 1);

        /*
         * Sets the code for a single row to 2. In this case, the "#" wildcard is
         * used. "content://com.example.app.provider/table3/3" matches, but
         * "content://com.example.app.provider/table3 doesn't.
         */
        sUriMatcher.addURI("com.example.app.provider", "table3/#", 2);
    }

    // Implements ContentProvider.query()
    public Cursor query(Uri uri, String[] projection, String selection,
                        String[] selectionArgs, String sortOrder) {
```

```
        /*
         * Choose the table to query and a sort order based on the code returned for
         * the incoming
         * URI. Here, too, only the statements for table 3 are shown.
         */
        switch (sUriMatcher.match(uri)) {
            // If the incoming URI was for all of table3
            case 1:
                if (TextUtils.isEmpty(sortOrder)) sortOrder = "_ID ASC";
                break;

            // If the incoming URI was for a single row
            case 2:
                /*
                 * Because this URI was for a single row, the _ID value part is
                 * present. Get the last path segment from the URI; this is the _ID value.
                 * Then, append the value to the WHERE clause for the query
                 */
                selection = selection + "_ID = " + uri.getLastPathSegment();
                break;

            default:
                // If the URI is not recognized, you should do some error handling here.
        }
        // call the code to actually do the query
        return null;
    }

    ...
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Designing Content URIs, https://developer.android.com/guide/topics/providers/content-provider-creating#ContentURI

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\ExampleProvider.javas

`UriMatcher` recognizes the following wildcards for content URI pattern matching:

*: Matches a string of any valid characters of any length.

#: Matches a string of numeric characters of any length.

```
// Imagine a ContentProvider that wishes to recognize the following content URIs
content://com.example.app.provider/table1           [A table called table1]
content://com.example.app.provider/table2/dataset1  [A table called dataset1]
content://com.example.app.provider/table2/dataset2  [A table called dataset2]
content://com.example.app.provider/table3           [A table called table3]
content://com.example.app.provider/table3/1         [A Row identified by 1 in table3]

// Example Matches
content://com.example.app.provider/*        [Matches any content URI in the provider]
content://com.example.app.provider/table2/* [Matches a content URI for the tables
    dataset1 and dataset2, but doesn't match the Content URI for table1 or table3]

content://com.example.app.provider/table3/# [Matches a content URI for single rows in
table3, such as  content://com.example.app.provider/table3/6 or the row identified by 6.
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Designing Content URIs, https://developer.android.com/guide/topics/providers/content-provider-creating#ContentURI

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\ExampleProvider.javas

Mime Types

ContentProviders return MIME types (TODO: why? What for?).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, MIME Type Reference, https://developer.android.com/guide/topics/providers/content-provider-basics#MIMETypeReference

Mime types always have the format **type/subtype**. Mime types can be standard or custom (aka "Vendor-specific Mime types").

```
// Standard
text/html

// Custom
vnd.android.cursor.item/phone_v2
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, MIME Type Reference, https://developer.android.com/guide/topics/providers/content-provider-basics#MIMETypeReference

Standard Mime Types Reference. See ...

(Internet Assigned Numbers Authority 2021, "Media Types", https://www.iana.org/assignments/media-types/media-types.xhtml) http://blog.kwyps.com/2011/07/how-to-view-sqlite-database-on-your.html

For custom mime types the **type** value is always:

vnd.android.cursor.**dir** for multiple rows.

vnd.android.cursor.**item** for a single row.

... and the **subtype** is provider specific.

```
// Contacts Provider telephone row
vnd.android.cursor.item/phone_v2

// In response to a Content URI content://com.example.trains/Line1 (Line1 being a table)
vnd.android.cursor.dir/vnd.example.line1

// In response to a Content URI content://com.example.trains/Line2/5 (Row 5 in table Line2)
vnd.android.cursor.item/vnd.example.line2
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, MIME Type Reference, https://developer.android.com/guide/topics/providers/content-provider-basics#MIMETypeReference

Most content providers define contract class constants for the MIME types they use.

```
// Contacts Provider Contract Class, ContactsContract.RawContacts, defines
public static final String CONTENT_TYPE = "vnd.android.cursor.dir/raw_contact";
public static final String CONTENT_ITEM_TYPE = "vnd.android.cursor.item/raw_contact";
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, MIME Type Reference, https://developer.android.com/guide/topics/providers/content-provider-basics#MIMETypeReference

Content Resolver Coding

Basic Procedure (to read data)

Code a Content Resolver as follows:

Note for this example ensure you have some words in your User Dictionary ...

Hardware device:

Settings > Language & Input > Personal Dictionary (or, if unavailable)

https://play.google.com/store/apps/details?id=com.usr.dict.mgr&hl=en

Emulator:

Settings > Language & Input > Personal Dictionary

To access a content provider you need the relevant use-permission in AndroidManifest.xml in your client application.

```
<uses-permission android:name="android.permission.READ_USER_DICTIONARY">
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Content provider permissions, https://developer.android.com/guide/topics/providers/content-provider-basics#Permissions

Get some data

```
public class ContentResolverDemoActivity extends Activity {
...
public void goRead(View view) {
...
// Get some data
Uri contentUri = UserDictionary.Words.CONTENT_URI;
// Aka "projection"
String[] sourceColumnNames = new String[]{
        UserDictionary.Words._ID,
        UserDictionary.Words.WORD,
        UserDictionary.Words.FREQUENCY
};
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Retrieving data from the provider, Constructing the query https://developer.android.com/guide/topics/providers/content-provider-basics#Query

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\ContentResolverDemoActivity.java

Filter and sort (optionally)

```
// Filter and Sort
// For use with wildcards we must use "LIKE" not " = ?"
String selectionClause = UserDictionary.Words.WORD + " LIKE ?";
String[] selectionArgs = new String[]{"z%"}; // Case insensitive search with wildcard
String sortOrder = UserDictionary.Words.WORD + " DESC";
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Retrieving data from the provider, Constructing the query https://developer.android.com/guide/topics/providers/content-provider-basics#Query

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\ContentResolverDemoActivity.java

Fire query to retrieve cursor.

```
try {
    cursor = getContentResolver().query(contentUri,
                                        sourceColumnNames,
                                        selectionClause,
                                        selectionArgs,
                                        sortOrder);

    // If sourceColumnID not available in selected record then an exception is
    // thrown but is not trappable (this is strange). We therefore
    // continue execution with a "finally" clause an test cursor for
    // null.
} finally {
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Retrieving data from the provider, Constructing the query *https://developer.android.com/guide/topics/providers/content-provider-basics#Query*

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\ContentResolverDemoActivity.java

Iterate over cursor

```
} finally {
    // If sourceColumnID not available in selected record then an exception is
    // thrown but is not trappable (this is strange). We therefore
    // continue execution with a "finally" clause an test cursor for
    // null.

    if (cursor == null) {
        // Some providers return null if an error occurs,
        // others throw an exception
        throw new ProviderCustomException(
                "Something went wrong with the content resolver query");

    } else if (cursor.getCount() < 1) {
        // If the Cursor is empty, the provider found no matches
        Toast.makeText(getApplicationContext(),
                       "No data returned to cursor",
                       Toast.LENGTH_SHORT).show();

    } else {
        // Data returned, do something with it
        while (cursor.moveToNext()) {
            for (String columnName : sourceColumnNames) {
                // Get the name and value of
                output += String.format(" %s: %s",
                                        columnName,
                                        cursor.getString(cursor.getColumnIndex(
                                                columnName))
                );
            }
            output += "\n";
        }
        mOutputTextView.setText(output);
    } // if ... else
} // Try ... finally
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Retrieving data from the provider, Constructing the query *https://developer.android.com/guide/topics/providers/content-provider-basics#Query*

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\ContentResolverDemoActivity.java

Aspects

Protect against SQL injection

Use selection and selectionArgs parameters work together to prevent sql injection (aka "protecting against malicious input").

```
// Don't do this
String SelectionClause =  UserDictionary.Words.WORD + " = " + mUserInput;
// This could result in "word = nothing; DROP TABLE *;"

// Do this
String selectionClause = UserDictionary.Words.WORD + " = ?";
String[] selectionArgs = {"fuck"};
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Retrieving data from the provider, Constructing the query https://developer.android.com/guide/topics/providers/content-provider-basics#Query

Display

Display results with an AdapterView subclass (ListView or GridView) and an Adapter. See Adapter Views.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Retrieving data from the provider, Displaying query results, https://developer.android.com/guide/topics/providers/content-provider-basics#DisplayResults

Provider Data Types

ContentProviders have upto the following datatypes:

- Intenger
- Long integer (long)
- Floating point
- Long floating point (double)
- Binary Large Object (BLOB), implements as 64KB byte aray.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Provider Data Types, https://developer.android.com/guide/topics/providers/content-provider-basics#DataTypes

Determine the datatype of a **cursor** with cursor.getType(), but the documentation for the ContentProvider should also list the datatype.

```
for (String columnName : sourceColumnNames) {
    columnIndex = cursor.getColumnIndex(
            columnName);
    output += String.format(" %s: '%s' [Type: %s]",
                            columnName,
                            cursor.getString(columnIndex),
                            cursor.getType(columnIndex)
    );
}

// Output
_id: '4' [Type: 1] apid: '0' [Type: 1] locale: 'en_US [Type: 3] ...
```

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages) https://developer.android.com/reference/android/database/Cursor#getType(int)

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\ContentResolverDemoActivity.java

When retrieving cursor values: Cursor implementations contain several "get" methods for retrieving different types of data from the object. For example, the previous snippet uses getString(). They also have a getType() method that returns a value indicating the data type of the column.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Retrieving data from the provider, Getting data from query results, https://developer.android.com/guide/topics/providers/content-provider-basics#GettingResults

Contract Classes

ContentProvider contract classes help clients work with features such as content URIs, column names, intent actions, mime types and others. ContentProviders should provide contract classess.

```
/**
UserDictionary Contract class
Content URI constant: UserDictionary.Words.CONTENT_URI.
Column Name constants:
    UserDictionary.Words._ID,
    UserDictionary.Words.WORD,
    UserDictionary.Words.LOCALE

Contacts Contract classes:
ContactsContract
ContactsContract.Intents.Insert

*/
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Retrieving data from the provider, Contract Classes, https://developer.android.com/guide/topics/providers/content-provider-basics#ContractClasses

Data Access Techniques

Insert, Update, Delete

Insert data into a content provider with a content resolver, like this ...

```
public void goInsert(View view) {

    Uri insertedRowUri = null;

    ContentValues newValues = new ContentValues();

    // No need for a  ID entry as this will issued a number automatically by the
    // provider.
    newValues.put(UserDictionary.Words.APP_ID, ".myfirstapp02");

    // Columns that do not have a value put in them acquire "null" by default (in
    // one experiment). In this example  UserDictionary.Words.LOCALE is given no
    // new value.
    newValues.put(UserDictionary.Words.WORD, "fuck");
    newValues.putNull(UserDictionary.Words.FREQUENCY);

    // Duplicate words are not checked for, so a duplicated word will go through
    // without error.
    insertedRowUri = getContentResolver().insert(UserDictionary.Words.CONTENT_URI,
                                                 newValues);
    String message = "";
    if (insertedRowUri != null) {
        // You could also use ContentUris.parseId(insertedRowUri) to return a long.
```

```
        message = insertedRowUri.toString();
    } else {
        message = "Nothing inserted";
    }

    Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).show();
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics; Inserting, updating, and deleting data; Inserting Data, https://developer.android.com/guide/topics/providers/content-provider-basics#Inserting

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\ContentResolverDemoActivity.java

Update ...

```
public void goUpdate(View view) {

    int rowsUpdated = 0;

    // Protect against SQL injection.
    String selectionClause = UserDictionary.Words.WORD + " LIKE ?";
    String[] selectionArgs = {"fuck"};

    ContentValues updateValues = new ContentValues();
    updateValues.put(UserDictionary.Words.LOCALE, "en AU");

    rowsUpdated = getContentResolver().update(UserDictionary.Words.CONTENT_URI,
                                              updateValues,
                                              selectionClause,
                                              selectionArgs);

    String message = String.format("%d rows updated", rowsUpdated);
    Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).show();
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics; Inserting, updating, and deleting data; Updating data, https://developer.android.com/guide/topics/providers/content-provider-basics#Updating

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\ContentResolverDemoActivity.java

Delete ...

```
public void goDelete(View view) {
    int rowsDeleted = 0;

    // Protect against SQL injection.
    String selectionClause = UserDictionary.Words.WORD + " LIKE ?";
    String[] selectionArgs = {"fuck"};


    rowsDeleted = getContentResolver().delete(UserDictionary.Words.CONTENT_URI,
                                              selectionClause,
                                              selectionArgs);

    String message = String.format("%d rows deleted", rowsDeleted);
    Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).show();
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics; Inserting, updating, and deleting data; Deleting data, https://developer.android.com/guide/topics/providers/content-provider-basics#Deleting

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\ContentResolverDemoActivity.javas

Asynchronous Loading

You should do queries in a separate thread one way to do this is it load results asynchronously with a Loader. See Loaders.

Batch Access

Batch access is useful for providing:

- Efficient inserts, updates, or deletes of large numbers of rows; or
- Atomic data changes, changes that succeed for fail as a whole.

```
// TODO
ContentProviderOperation
ContentResolver.applyBatch().
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Alternative forms of provider access, Batch access, https://developer.android.com/guide/topics/providers/content-provider-basics#Batch

Data Access with Intents

You can pass an intent to a provider:

- Display data (e.g. CalendarProvider displays a date) ;
- Change data (e.g CalendarProvider inserts a new event); and/or
- Get back data as a result. (e.g. ContactsProvider returns specific contact info).
- To do you'll need, and the provider must provide, (temporary) "Uri permissions" rather than more general permissions.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Content provider basics, Alternative forms of provider access, Data access via intents, https://developer.android.com/guide/topics/providers/content-provider-basics#Intents

Content Provider Creation

Procedure Summary

Design data storage. Data storage can be of two basic types:

- File data (E.g. photos, audio, videos).
- Structured data (E.g. database, array). It is common to use SQLite for database data storage.
- Implement a subclass of `ContentProvider`.
- Define the Provider's:
- Table and column names
- Authority string and Content URIs.
- Permissions needed for client apps.
- If intents will be handled, define intent actions, extras data, and flags.
- Expose all the above definitions in a separate contract class.
- Other optional pieces. E.g. Sample data; implementation of AbstractThreadedSyncAdapter to synchronize data between the provider and cloud-based data.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, https://developer.android.com/guide/topics/providers/content-provider-creating

Design data storage

You can store data however you like. For example: SQLite database; file storage; network I/O (with java.net.* and android.net.*); or network syncing (see Sample Sync Adapter).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Designing data storage, https://developer.android.com/guide/topics/providers/content-provider-creating#DataStorage

Tips:

- Table data should have a primary key. The primary key should also use BaseColumns._ID as ListViews require one column to have the name "_ID".
- Don't store large pieces of data (e.g. bitmaps) in a table. Store it as a file and point to it with your provider.
- Use BLOBS for: data that varies in size or structure (e.g. for JSON data); or to implement a schema-independent table.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Designing data storage, https://developer.android.com/guide/topics/providers/content-provider-creating#DataStorage

A schema-independent table has a primary key column, a MIME type column, and one or more generic columns as BLOBs. The meaning of the BLOB is specified by the MIME type. This allows you to store different row types in the same table. E.g. ContactsConstract.Data is a schema-independent table.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Designing data storage, https://developer.android.com/guide/topics/providers/content-provider-creating#DataStorage

Implementing a subclass of ContentProvider

ContentProvider, in the framework, is an abstract class. When you subclass it you must therefore implement all the abstract methods:

- onCreate()
- query()
- insert()
- update()
- delete()
- getType()

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Implementing the ContentProvider class, https://developer.android.com/guide/topics/providers/content-provider-creating#ContentProvider

Notes about implementing these abstract methods:

- Although you must implement them you can signal to a client using your Provider, through a ContentResolver, that your method does nothing. To do this just return the expected datatype. E.g. return 0 when insert() is called to prevent insertions.
- All of the methods except for onCreate() need to be thread safe as they can be called by multiple threads.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Implementing the ContentProvider class, https://developer.android.com/guide/topics/providers/content-provider-creating#ContentProvider

onCreate(). Avoid lengthy operations in onCreate. Only do fast initialization.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Implementing the ContentProvider class, Implementing the onCreate() method, https://developer.android.com/guide/topics/providers/content-provider-creating#OnCreate

onCreate(). For SQLite operations specifically only creath the DBHelper object in the Provider onCreate(). Don't access DBHelper.getWriteableDatabase() or DBHelper.getReadableDatabase() until peforming database manipulation or retrieval.

```
public class ExampleProvider extends ContentProvider {

...

    private NorthwindDBHelper mDBHelper = null;
    private SQLiteDatabase mDB = null;


    @Override
    public boolean onCreate() {
        mDBHelper = new NorthwindDBHelper(getContext());
        return true;
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        // NorthwindDBHelper.onCreate() is called at this point, thereby
        // heavy database creation operations are properly avoided in ExampleProvider
        // .onCreate()
        mDB = mDBHelper.getWritableDatabase();
        return null;
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Implementing the ContentProvider class, Implementing the onCreate() method, https://developer.android.com/guide/topics/providers/content-provider-creating#OnCreate

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyFirstApp02\app\src\main\java\au\com\softmake\myfirstapp02\ExampleProvider.java

Implementing ContentProvider Mime Types

See Mime Types

Your ContentProvider has two methods for returning MIME Types:

getType(). You must implement this for all providers.

getStreamType(). You must also(?) implement this if your providers offers files.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Implementing ContentProvider MIME Types, https://developer.android.com/guide/topics/providers/content-provider-creating#MIMETypes

getType() for tables. Return the MIME type described by the Content URI argument.

```
// For standard types of data (text, html, jpeg, etc) you should return the
// standard MIME types.
text/plain, text/html, image/jpeg

// For content URIS that point to a row or rows of table data always return
// a custom MIME format...

// In response to a Content URI content://com.example.trains/Line1 (Line1 being a table)
vnd.android.cursor.dir/vnd.example.line1

// In response to a Content URI content://com.example.trains/Line2/5 (Row 5 in table Line2)
vnd.android.cursor.item/vnd.example.line2
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Implementing ContentProvider MIME Types, https://developer.android.com/guide/topics/providers/content-provider-creating#MIMETypes

getStreamTypes() for files. Returns a string array of the MIME types your provider handles.

```
// For example, consider a provider that offers photo images as files in .jpg, .png, and
.gif format. If an application calls ContentResolver.getStreamTypes() with the filter string
image/*, then the ContentProvider.getStreamTypes() method should return;

{ "image/jpeg", "image/png", "image/gif"}

// If the client app is only interested in .jpg files, then it can call
ContentResolver.getStreamTypes() with the filter string *\/jpeg, and
ContentProvider.getStreamTypes() should return:

{"image/jpeg"}


// If your provider doesn't offer any of the MIME types requested in the filter string,
getStreamTypes() should return:

null
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Implementing ContentProvider MIME Types, https://developer.android.com/guide/topics/providers/content-provider-creating#MIMETypes

ContentProvider Contract Class

Create a ContentProvider Contract class, a class declared `public final`, in order to provide URIs, column names, MIME types, and other meta data. Provide a .jar file that consuming developers can compile into their application. All this aids the consuming developer with code completion, documentation, etc.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Implementing a contract class, https://developer.android.com/guide/topics/providers/content-provider-creating#ContractClass

Defining the required permissions

When creating a ContentProvider the foundational data access kinds will have there own permissions. E.g. SQLiteDatabases and files can be made public or private. The general strategy is to make those foundational data access kinds private to your application and control permissions at the ContentProvider level.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Implementing content provider permissions, https://developer.android.com/guide/topics/providers/content-provider-creating#Permissions

By default your ContentProvider allows read and write access to the wold (even if the foundational data is private). Therefore set permissions in the manifest.

```
// TODO: Make clear the various relationships between permission manifest entries.
<manifest ..>

    <uses-permission android:name="android.permission.READ_USER_DICTIONARY" />
    <uses-permission android:name="android.permission.WRITE_USER_DICTIONARY" />
    <uses-permission androi (Google API Guides, 2013. Develop > API
Guides)d:name="android.permission.READ_CONTACTS" />

    <permission android:name=""></permission>

    <application
        ...
        <provider android:authorities="au.com.softmake.myfirstapp02.exampleprovider"
                  android:name=".ExampleProvider"
                  android:readPermission=""></provider>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Implementing content provider permissions, https://developer.android.com/guide/topics/providers/content-provider-creating#Permissions

Set further ContentProvider Manifest info

The <provider> element.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, The <provider> element, https://developer.android.com/guide/topics/providers/content-provider-creating#ProviderElement

Data Access with Intents

Instead of having client code access your ContentProvider via a ContentResolver you can have clients call Activities in your application which server as front ends to your ContentProvider. See Data Access with Intents

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content Providers, Creating a content provider, Intents and data access, , https://developer.android.com/guide/topics/providers/content-provider-creating#Intents

Framework ContentProviders

Calendar and Contacts Providers

Calendar Provider

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User data & Identity, Calendar provider overview, https://developer.android.com/guide/topics/providers/calendar-provider

Contacts Provider

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, User data & Identity, Contacts provider, Overview, https://developer.android.com/guide/topics/providers/calendar-provider

File Provider

File Provider (for securely sharing a file with other apps).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing files, Overview, https://developer.android.com/training/secure-file-sharing/

Procedure for securely sharing a file with other apps, using the File Provider:

Server App

Save some files somewhere in a custom directory. See Files.

Also
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\ShareFilesDemoActivity.java > saveFileInternal()

FileProvider is a part of the v4 Support Library. Reference the v4 Support Library.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing files, Setting up file sharing, https://developer.android.com/training/secure-file-sharing/setup-sharing

Define a FileProvider in your AndroidManifest.xml. This specifies: the URI authority; and the name of the XML file that specifies the directories you can share.

```
...
        <provider
            android:name="android.support.v4.content.FileProvider"
            android:authorities="au.com.softmake.mysecondapp.fileprovider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/file_provider_paths" />
        </provider>
    </application>
</manifest>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing files, Setting up file sharing, Specify the FileProvider, https://developer.android.com/training/secure-file-sharing/setup-sharing#DefineProvider

Specify the shareable directories (for directory types see: Files Intro) at res/xml/file_provider_paths.xml (the file you specified in the previous step).

```
<?xml version="1.0" encoding="utf-8"?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <files-path name="myCoolFilesPath" path="coolFilesDir/"  />
    <cache-path name="myCoolCachePath" path="coolCacheDir/" />
    <external-path name="myCoolExternalPath" path="coolExternalDir/" />
</paths>s
```

E.g. coolsFilesPath/ is permissions for /data/data/com.example. mysecondapp/files/coolFilesDir/. The name attribute tells the FileProvider to add the path segment, myCoolFilesPath to content URIs for files in fils/coolFilesDir/.

This XML file is the only way to specify permissions for the directories you want to share, you can't do it programamaticaaly.

When the app generates content URIs for a file it does so by concatenating:
* The URI authority (e.g. au.com.softmake.mysecondapp.fileprovider);
* the path name (e.g. myCoolFilesPath) plus "/";
* the file name.

E.g. funky.txt will get a content URI of
content://au.com.softmake.mysecondapp.fileprovider/myCoolFilesPath/funky.txt

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing files, Setting up file sharing, Specify sharable directories, https://developer.android.com/training/secure-file-sharing/setup-sharing#DefineMetaData

Sharing a file: Provide a means for client apps to select a file. This can be done through a File Selection Activity. This is an activity that clients can start by: calling `startActivityForResult();` with an Intent containing the `ACTION_PICK` action.

Specify your file selection activity in the manifest, with an Intent that filters on mimetype.

```
<activity
    android:name="au.com.softmake.mysecondapp.sharefiles.FileSelectionActivity"
    android:label="@string/title_activity_file_selection"

android:parentActivityName="au.com.softmake.mysecondapp.sharefiles.ShareFilesDemoActivity">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="au.com.softmake.mysecondapp.sharefiles.ShareFilesDemoActivity" />

    <intent-filter>
        <action android:name="android.intent.action.PICK" />

        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.OPENABLE" />

        <data android:mimeType="text/plain" />
    </intent-filter>
</activity>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\AndroidManifest.xml

Display a list of files for the user to choose from

```
public class FileSelectionActivity extends ListActivity {
    // 'Regular' (as for getFilesDir()) as opposed to 'Temporary' (as for getCacheDir())
    private File mAppInternalRegularStorageRootDir = null;
    private File mAppInternalRegularStorageSubDir = null;

    File[] mAppSubDirFiles = null;
    // Filenames corresponding to mAppSubDirFiles
    SortedMap<Integer, String> mAppSubDirFilesMap = new TreeMap<Integer, String>();
```

```
    Intent mResultIntent = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Show the Up button in the action bar.
        setupActionBar();

        mResultIntent = new Intent("au.com.softmake.mysecondapp.ACTION_RETURN_FILE");

        // Get the custom file names.
        mAppInternalRegularStorageRootDir = getFilesDir();
        mAppInternalRegularStorageSubDir = new File(mAppInternalRegularStorageRootDir,
                                                    "coolFilesDir");
        mAppSubDirFiles = mAppInternalRegularStorageSubDir.listFiles();

        int i = 0;
        for (File file : mAppSubDirFiles) {
            mAppSubDirFilesMap.put(i++, file.getAbsolutePath());
        }
        String[] appSubDirFileNames = mAppSubDirFilesMap.values().toArray(new String[0]);

        // Set the activity's result to null to begin with.
        setResult(Activity.RESULT_CANCELED, null);

        // Create and Adapter from context, simple item layout, dataList
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                                                                android.R.layout
                                                                        .simple_list_item_1,
                                                                appSubDirFileNames);
        // Set the activity's list to the adapter.
        ListView listView = this.getListView();
        listView.setAdapter(adapter);
        ...
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing files, Sharing a file, Create a file selection Activity, https://developer.android.com/training/secure-file-sharing/share-file#CreateFileSelection

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\sharefiles\FileSelectionActivity.java

Respond to a file selection:

- Use the created file provider to get a content URI.
- Grant permissions for the file.
- Pass the Intent containing the content URI , mimetype, and permissions to setResult().

```
public class FileSelectionActivity extends ListActivity {
    // 'Regular' (as for getFilesDir()) as opposed to 'Temporary' (as for getCacheDir())
    private File mAppInternalRegularStorageRootDir = null;
    private File mAppInternalRegularStorageSubDir = null;

    File[] mAppSubDirFiles = null;
    // Filenames corresponding to mAppSubDirFiles
    SortedMap<Integer, String> mAppSubDirFilesMap = new TreeMap<Integer, String>();

    Intent mResultIntent = null;

    private AdapterView.OnItemClickListener mItemClickListener = new AdapterView
            .OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

            File requestFile = new File(mAppSubDirFilesMap.get(position));
            Uri fileUri = null;

            // Use the created file provider to get a content URI
            try {
```

```
                fileUri = FileProvider.getUriForFile(getApplicationContext(),
                                                     "au.com.softmake.mysecondapp" + "" +
                                                             ".fileprovider",
                                                     requestFile);
            } catch (IllegalArgumentException e) {
                Log.e("File Selector",
                      "The selected file can't be shared: " + requestFile.getName());
            }

            String activityResultString = "";
            int resultCode = 0;

            if (fileUri != null) {
                // Grant permissions for the file
                mResultIntent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);

                // Pass the Intent containing the content URI, mimetype,
                // and permissions to setResult().

                // Set Content URI and Mimetype
                mResultIntent.setDataAndType(fileUri,
                                             getContentResolver().getType(fileUri));

                resultCode = Activity.RESULT_OK;

            } else {
                mResultIntent.setDataAndType(null, "");
                resultCode = Activity.RESULT_CANCELED;

            }
            // Set the result using previously set resultCode
            setResult(resultCode, mResultIntent);

            // User feedback handling
            switch (resultCode){
                case Activity.RESULT_OK:
                    activityResultString = "Activity.RESULT_OK";
                    break;
                case Activity.RESULT_CANCELED:
                    activityResultString = "Activity.RESULT_CANCELED";
                    break;
                default:
                    try {
                        throw new Exception(
                                "Unexpected switch reached (ResultCode = " + resultCode);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
            }

            String text = String.format(Locale.getDefault(),
                                        "Item name: %s, position %d, result: %s",
                                        requestFile.getName(), position,
                                        activityResultString);
            Toast toast = Toast.makeText(getApplicationContext(), text,
                                         Toast.LENGTH_SHORT);
            toast.show();
        }
    };
```

Provide users with a means to return immediately to the client app once they have chosen a file. Do this by calling te activity's finish() method.

```
// Call finish() from an ActionBar Command

<!-- \res\menu\file_selection_activity_action_bar.xml -->
<?xml version="1.0" encoding="utf-8"?>
<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="au.com.softmake.mysecondapp.sharefiles.FileSelectionActivity">
    <item
        android:id="@+id/action_done"
        android:orderInCategory="100"
        android:title="@string/action_done"
```

```
        app:showAsAction="ifRoom" />
</menu>

public class FileSelectionActivity extends Activity {

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

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_done:
                Toast.makeText(getApplicationContext(), "Done",
                               Toast.LENGTH_SHORT).show();
                // Provides a way for users to return immediately to the client app
                // after they have chosen a file.
                finish();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
...
```

Client App

The client app can requests the file, then, upon receipt, dipslay file meta data and file contents. The steps are:

Set an Intent that will match the server intent.

```
public class ShareFileClientDemoActivity extends ActionBarActivity {

    private Intent mRequestFileIntent;

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

        // Set the Intention Action and mimetype so that the server
        // FileSelectionActivity will be matched.
        mRequestFileIntent = new Intent(Intent.ACTION_PICK);
        mRequestFileIntent.setType("text/plain");
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing files, Requesting a shared file, Send a request for the file, https://developer.android.com/training/secure-file-sharing/request-file#SendRequest

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\sharefiles\ShareFileClientDemoActivity.java

Start an activity for a result, using the previously defined intent (in effect this broadcasts an intent to the entire android system).

```
public class ShareFileClientDemoActivity extends ActionBarActivity {

...

    /**
     * Called from the action bar menu
     */
    protected void requestFile() {
        // 0 = request code will be returned in onActivityResult().
```

```
        startActivityForResult(mRequestFileIntent, 0);
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing files, Requesting a shared file, Send a request for the file, https://developer.android.com/training/secure-file-sharing/request-file#SendRequest

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\sharefiles\ShareFileClientDemoActivity.java

Get the file's content URI from the returned intent.

```
public class ShareFileClientDemoActivity extends ActionBarActivity {
...

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
                                    Intent returnIntent) {
        // Not sure if this needs to be here.
        // It might delay code unnecessarily. But the functionality works with or
        // without this line.
        super.onActivityResult(requestCode, resultCode, returnIntent);

        if (resultCode != RESULT_OK) {
            // Exit immediately
            return;
        } else {
            android.net.Uri returnUri = returnIntent.getData();

...
        }
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing files, Requesting a shared file, Access the requested file, https://developer.android.com/training/secure-file-sharing/request-file#OpenFile

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\sharefiles\ShareFileClientDemoActivity.java

Get file meta data. You could use some of this meta data to determine what you want to do with the file (e.g. not copy it because it is too large).

```
public class ShareFileClientDemoActivity extends ActionBarActivity {

...

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
                                    Intent returnIntent) {
        // Not sure if this needs to be here.
        // It might delay code unnecessarily. But the functionality works with or
        // without this line.
        super.onActivityResult(requestCode, resultCode, returnIntent);

        if (resultCode != RESULT_OK) {
            // Exit immediately
            return;
        } else {
            android.net.Uri returnUri = returnIntent.getData();

            String output = getFileMetaData(returnUri);
...
        }
    }

    private String getFileMetaData(android.net.Uri uri) {
        String result = "";
```

```
        String returnFileMimeType = getContentResolver().getType(uri);

        Cursor returnCursor = getContentResolver().query(uri, null, null, null,
                                                         null);

        int displayNameIndex = returnCursor.getColumnIndex(
                android.provider.OpenableColumns.DISPLAY_NAME);
        int sizeIndex = returnCursor.getColumnIndex(
                android.provider.OpenableColumns.SIZE);
        returnCursor.moveToFirst();
        String displayName = returnCursor.getString(displayNameIndex);
        Long size = returnCursor.getLong(sizeIndex);

        result = String.format("Display Name: %s\nSize: %d bytes\nMimeType: " +
                                        "%s\nURI: %s",
                                displayName,
                                size,
                                returnFileMimeType,
                                uri);
        return result;
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing files, Retrieving file information, https://developer.android.com/training/secure-file-sharing/retrieve-info

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\sharefiles\ShareFileClientDemoActivity.java

Get the file contents. The secret link between file contents and the return URI is ParcelFileDescriptor and FileDescriptor.

```
public class ShareFileClientDemoActivity extends ActionBarActivity {
...

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
                                    Intent returnIntent) {
        // Not sure if this needs to be here.
        // It might delay code unnecessarily. But the functionality works with or
        // without this line.
        super.onActivityResult(requestCode, resultCode, returnIntent);

        if (resultCode != RESULT_OK) {
            // Exit immediately
            return;
        } else {
            android.net.Uri returnUri = returnIntent.getData();

            String output = getFileMetaData(returnUri);
            output += getFileContents(returnUri);

            TextView outputTextView = (TextView) findViewById(R.id.textview_output);
            outputTextView.setText(output);
            Toast.makeText(getApplicationContext(), output, Toast.LENGTH_SHORT).show();
        }
    }
...

    private String getFileContents(android.net.Uri uri) {
        android.os.ParcelFileDescriptor inputParcelFileDescriptor;
        /* Try to open the file for read access. If the file isn't readable then it
        is not found. We raise an error and exit. */
        try {
            inputParcelFileDescriptor = getContentResolver().openFileDescriptor(
                    uri, "r");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
            Log.e("ShareFileClientDemoActivity", "File not found");
            return "File not found";
        }

        // Print file contents to stream
        // Get a Java File Descriptor
```

```
        java.io.FileDescriptor fileDescriptor = inputParcelFileDescriptor
                .getFileDescriptor();
        FileReader fileReader = new FileReader(fileDescriptor);
        BufferedReader inputStream = new BufferedReader(fileReader);

        StringBuilder sb = new StringBuilder();
        String line;
        sb.append("\n\nFile Contents:\n");
        try {
            while ((line = inputStream.readLine()) != null) {
                sb.append(line + "\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

        return sb.toString();
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing files, Requesting a shared file, Access the requested file, https://developer.android.com/training/secure-file-sharing/request-file#OpenFile

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\sharefiles\ShareFileClientDemoActivity.java

Storage Access Framework

The Storage Access Framework provides a single interface for users to browse and open files (documents, images, etc) from several cloud or local storage services. Made available from Android 4.4 (API Level 19). The Storage Access Framework uses a ContentProvider subclassed from DocumentsProvider.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Content providers, Open files using storage access framework, https://developer.android.com/guide/topics/providers/document-provider

Loaders (Deprecated)

Intro

Loaders:

- Are available to Activities and Fragments;
- Provide asynchronous loading of data;
- Monitor their data soruce and deliver new results when the content changes; and
- Automatically reconnect to the last loader's cursor when being recreated after a configuration change (they therefore don't need to re-query their data).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, https://developer.android.com/guide/components/loaders

A Loader gets data in one of three ways:

1. A CursorLoader to load data backed by a Content Provider.
2. A subclass of Loader; or
3. A subclass of AsyncTaskLoader (which is itself a subclass of Loader).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, Using Loaders in an Application, https://developer.android.com/guide/components/loaders#app

When using a CursorLoader to load data backed by a Content Provider. You'll often want to plug the loaded cursor into an Adapter. An Adpater maps a cursor to interface elements.

Loaders exist from version 3.0 (API level 11).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, https://developer.android.com/guide/components/loaders

Important classes and interfactes:

- LoaderManager
- LoaderManager.LoaderCallbacks
- Loader
- AsyncTaskLoader
- CursorLoader

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, https://developer.android.com/guide/components/loaders

Coding a Loader

Loader Basics

The `LoadManager` manages one or more `Loader` instances. There is only one LoaderManager per activity or fragment.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, Starting a Loader, https://developer.android.com/guide/components/loaders#starting

With an activities's `onCreate()` or a fragment's `onActivityCreate()` initialize a `Loader`.

```
public class LoaderDemoActivity extends ListActivity {

...

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

                    // Prepare the Loader. This either re-connects with an existing Loader
                    // or starts
                    // a new one.
                    getLoaderManager().initLoader(0, null, this);
...
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, Starting a Loader, https://developer.android.com/guide/components/loaders#starting

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\LoaderDemoActivity.java

Create and instantiate an empty class level Adapter. Instead of passing a cursor to the third parameter we pass null. The CursorLoader will subsequently pass the cursor to the Adapter in onLoadFinished()

```
SimpleCursorAdapter mAdapter;

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

          // Prepare the Loader. This either re-connects with an existing Loader
          // or starts
          // a new one.
          getLoaderManager().initLoader(0, null, this);

          String[] cursorColumnIDsToDisplay = new String[] {
                                    ContactsContract.Data._ID,
ContactsContract.Data.DISPLAY_NAME };
          int[] viewColumnIDs = new int[] { R.id._id, R.id.display_name };

          // This is an empty Adapter. Instead of passing a cursor to the third
          // parameter we pass null. The CursorLoader will pass the cursor to the
          // Adapter in onLoadFinished()
          mAdapter = new SimpleCursorAdapter(this,
                                    R.layout.list_item_cursor_file, null,
cursorColumnIDsToDisplay,
                                    viewColumnIDs, 0);
          this.setListAdapter(mAdapter);
..

}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, https://developer.android.com/guide/components/loaders

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\LoaderDemoActivity.java

LoaderManager Callbacks

Implement LoaderManager Callbacks. Be sure and set <Cursor> (it might be "<Object>").

```
public class LoaderDemoActivity extends ListActivity implements
                    LoaderManager.LoaderCallbacks<Cursor>
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, https://developer.android.com/guide/components/loaders

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\LoaderDemoActivity.java

LoaderManager Callback onCreateLoader: Create and return a CursorLoader that queries a ContentResolver.

```
// These are the Contacts rows that we will retrieve.
// Note we can retrieve more than we will display.
static final String[] CONTACTS_SUMMARY_PROJECTION = new String[] {
                    Contacts._ID, Contacts.DISPLAY_NAME, Contacts.CONTACT_STATUS,
                    Contacts.CONTACT_PRESENCE, Contacts.PHOTO_ID, Contacts.LOOKUP_KEY, };


@Override
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
          // This is called when a new Loader needs to be created. This
          // sample only has one Loader, so we don't care about the ID.
          Uri baseUri;

          baseUri = Contacts.CONTENT_URI;

          // Creating the CursorLoader, that loads the data.
          String select = "((" + Contacts.DISPLAY_NAME + " NOTNULL) AND ("
                                    + Contacts.HAS_PHONE_NUMBER + "=1) AND ("
                                    + Contacts.DISPLAY_NAME + " != '' ))";
```

```
            // The parameters of CursorLoader are passed as-is to
            // ContentResolver.query()
            return new CursorLoader(this, baseUri, CONTACTS_SUMMARY_PROJECTION,
                                    select, null, Contacts.DISPLAY_NAME + " COLLATE LOCALIZED
ASC");
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, https://developer.android.com/guide/components/loaders

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\LoaderDemoActivity.java

LoaderManager Callback onLoadFinished: Remove all use of the old data. Remove all use of the old data, but do not do it directly (e.g. do not `Cursor.close()` a `CursorLoader`), swap it out (e.g. use `CursorAdpater.swapCursor()`).

```
// This is the Adapter being used to display the list's data.
SimpleCursorAdapter mAdapter;
...

public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
    // Swap the new cursor in.  (The framework will take care of closing the
    // old cursor once we return.)
    mAdapter.swapCursor(data);
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, https://developer.android.com/guide/components/loaders

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\LoaderDemoActivity.java

LoaderManager Callback onLoaderReset: Remove your reference to old data.

```
// This is the Adapter being used to display the list's data.
SimpleCursorAdapter mAdapter;
...

public void onLoaderReset(Loader<Cursor> loader) {
    // This is called when the last Cursor provided to onLoadFinished()
    // above is about to be closed.  We need to make sure we are no
    // longer using it.
    mAdapter.swapCursor(null);
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, https://developer.android.com/guide/components/loaders

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\LoaderDemoActivity.java

Refresh data (Exemplified by implementing Search)

Setup the Search UI in the menu.

```
<!-- /res/menu/loader_demo.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="au.com.softmake.myfirstapp.LoaderDemoActivity" >

    <item
        android:id="@+id/action_search"
```

```
        android:icon="@drawable/ic_action_search"
        android:showAsAction="ifRoom"
        android:title="@string/action_search"/>
    <item
        android:id="@+id/action_settings"
        android:showAsAction="never"
        android:title="@string/action_settings"/>

</menu>

@Override
public boolean onCreateOptionsMenu(Menu menu) {

          // Inflate the menu; this adds items to the action bar if it is present.
          getMenuInflater().inflate(R.menu.loader_demo, menu);

          MenuItem searchMenuItem = menu.findItem(R.id.action_search);

          SearchView searchView = new SearchView(this);
          searchView.setOnQueryTextListener(this);
          searchMenuItem.setActionView(searchView);

          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();
          switch (id) {
          case R.id.action_search:
                    // Do nothing. We let the loader handle search
                    return true;
          case R.id.action_settings:
                    // Do something
                    return true;
          default:
                    return super.onOptionsItemSelected(item);
          }
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, https://developer.android.com/guide/components/loaders

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\LoaderDemoActivity.java

Add a class level filter string. Modify onCreateLoader to handle filtering.

```
// if non-null this is the user provided filter.
String mFilter;

@Override
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
          // This is called when a new Loader needs to be created. This
          // sample only has one Loader, so we don't care about the ID.
          Uri baseUri;

          if (mFilter != null) {
                    // Not null therefore use the user supplied filter
                    baseUri = Uri.withAppendedPath(Contacts.CONTENT_FILTER_URI,
Uri.encode(mFilter));
          } else {
                    // There is no filtering to do, return everything.
                    baseUri = Contacts.CONTENT_URI;
          }

          // Creating the CursorLoader, that loads the data.
          String select = "((" + Contacts.DISPLAY_NAME + " NOTNULL) AND ("
                              + Contacts.HAS_PHONE_NUMBER + "=1) AND ("
                              + Contacts.DISPLAY_NAME + " != '' ))";
```

```
            // The parameters of CursorLoader are passed as-is to
            // ContentResolver.query()
            return new CursorLoader(this, baseUri, CONTACTS_SUMMARY_PROJECTION,
                                    select, null, Contacts.DISPLAY_NAME + " COLLATE LOCALIZED
ASC");
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, https://developer.android.com/guide/components/loaders

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\LoaderDemoActivity.java

Implement Search Callbacks.

```
public class LoaderDemoActivity extends ListActivity implements
                    LoaderManager.LoaderCallbacks<Cursor>, OnQueryTextListener {
```

(Google Training, 2013. Develop > Training) http://developer.android.com/guide/components/loaders.html
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\LoaderDemoActivity.java

When you want to discard your old data and start over perform a `restartLoader()` which forces a new call to `onCreateLoader()`.

```
@Override
public boolean onQueryTextSubmit(String query) {
            // Ignore. All the requerying occurs on text change (for each letter) rather
            // than at the end.
            return false;
}

@Override
public boolean onQueryTextChange(String newText) {
            // Called when the action bar search text has changed.
            // Update the search filter and restart the loader to force a requery with new
filter.

            // Ensure an empty text clears any filtering, otherwise set new filter string
            mFilter = !TextUtils.isEmpty(newText) ? newText : null;

            // Forces the onCreateLoader to be called again.
            getLoaderManager().restartLoader(0, null, this);
            return false;
}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Activities, Loaders, https://developer.android.com/guide/components/loaders

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\LoaderDemoActivity.java

Full Example

For the full CursorLoader example with a SimpleCursorAdapter, ListView, OnItemClickListener, and Search see ...

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\MyFirstApp\src\au\com\softmake\myfirstapp\LoaderDemoActivity.java

Using a Loader with a subclass of AsyncTaskLoader

Unless you need to provide regular feedback to a user for a background operation (e.g. through a determinate progress bar) you should prefer using a Loader with AsyncTaskLoader, rather than an AsyncTask (see Threads).

An AsyncTaskLoader is simpler to code and handles runtime configuration changes (e.g. when the user rotates the screen) without doing any extra work.

Using a Loader with a subclass of AsyncTaskLoader, example.

```
public class LoadImageFromNetworkLoaderActivity extends Activity implements
        LoaderManager.LoaderCallbacks {

    private ImageView mPictureImageView;

    // An indeterminate progress bar
    private ProgressBar mLoadingProgressBar;
    private Loader mLoader;

    private static class LoadImageFromNetworkAsyncTaskLoader extends
            AsyncTaskLoader<Bitmap> {

        private String mUrl = "";

        LoadImageFromNetworkAsyncTaskLoader(Context context, String url) {
            super(context);
            mUrl = url;
        }

        @Override
        public Bitmap loadInBackground() {
            return au.com.softmake.standardapplibrary.android.Network
                    .loadImageFromNetwork(
                    mUrl);
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity load image from network loader);
        mPictureImageView = (ImageView) findViewById(R.id.imageView_picture);
        mLoadingProgressBar = (ProgressBar) findViewById(R.id.progressBar_loading);

        mLoader = this.getLoaderManager().initLoader(0, null, this);
    }


    @Override
    public Loader onCreateLoader(int id, Bundle args) {
        return new LoadImageFromNetworkAsyncTaskLoader(getApplicationContext(),
                                                       "http://i.imgur.com/SikTbWe.jpg");
    }

    @Override
    public void onLoadFinished(Loader loader, Object data) {
        mPictureImageView.setImageBitmap((Bitmap) data);
        mLoadingProgressBar.setVisibility(View.INVISIBLE);
    }


    @Override
    public void onLoaderReset(Loader loader) {
    }

    public void getPicture(View view) {
        mLoadingProgressBar.setVisibility(View.VISIBLE);
        // Start processing
        mLoader.forceLoad();
    }
```

```
    public void clearPicture(View view) {
        mPictureImageView.setImageBitmap(null);
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\LoadImageFromNetworkLoaderActivity.java

XML parsing

Don't use the Java Stax parser.

The Java Stax parser can be made to work on Android but only with hacks.

(docx4java 2012, "JAXB Can Be Made to Run on Android", https://www.docx4java.org/blog/2012/05/jaxb-can-be-made-to-run-on-android/) https://www.docx4java.org/blog/2012/05/jaxb-can-be-made-to-run-on-android/

Use XmlPullParser ....

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Connectivity, Performating network operations, Parse XML data, https://developer.android.com/training/basics/network-ops/xml

There are two XmlPullParsers that you can use:

- KXmlParser via XmlPullParserFactory.newPullParser().
- ExpatPullParser, via Xml.newPullParser().

We'll use the ExpatPullParser just because this is the parser the Android documentation (Google Developer Guides, 2018) exemplifies ...

1. Start with an XML source.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
    From http://api.worldbank.org/v2/countries on 2018-04-06, as found on
    https://datahelpdesk.worldbank.org/knowledgebase/articles/898590-api-country-
queries
-->
<wb:countries page="1" pages="7" per_page="50" total="304"
xmlns:wb="http://www.worldbank.org">
    <wb:country id="ABW">
        <wb:iso2Code>AW</wb:iso2Code>
        <wb:name>Aruba</wb:name>
        <wb:region id="LCN" iso2code="ZJ">Latin America &amp;amp; Caribbean </wb:region>
        <wb:adminregion id="" iso2code="" />
        <wb:incomeLevel id="HIC" iso2code="XD">High income</wb:incomeLevel>
        <wb:lendingType id="LNX" iso2code="XX">Not classified</wb:lendingType>
        <wb:capitalCity>Oranjestad</wb:capitalCity>
        <wb:longitude>-70.0167</wb:longitude>
        <wb:latitude>12.5167</wb:latitude>
    </wb:country>
    <wb:country id="AFG">
        <wb:iso2Code>AF</wb:iso2Code>
        <wb:name>Afghanistan</wb:name>
        <wb:region id="SAS" iso2code="8S">South Asia</wb:region>
        <wb:adminregion id="SAS" iso2code="8S">South Asia</wb:adminregion>
        <wb:incomeLevel id="LIC" iso2code="XM">Low income</wb:incomeLevel>
        <wb:lendingType id="IDX" iso2code="XI">IDA</wb:lendingType>
        <wb:capitalCity>Kabul</wb:capitalCity>
        <wb:longitude>69.1761</wb:longitude>
        <wb:latitude>34.5228</wb:latitude>
    </wb:country>
    <wb:country id="AFR">
        <wb:iso2Code>A9</wb:iso2Code>
```

```
        <wb:name>Africa</wb:name>
        <wb:region id="NA" iso2code="NA">Aggregates</wb:region>
        <wb:adminregion id="" iso2code="" />
        <wb:incomeLevel id="NA" iso2code="NA">Aggregates</wb:incomeLevel>
        <wb:lendingType id="" iso2code="">Aggregates</wb:lendingType>
        <wb:capitalCity />
        <wb:longitude />
        <wb:latitude />
    </wb:country>
    ...
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\raw\country_data_from_world_bank.xml

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Connectivity, Performating network operations, Parse XML data, Analyze the Feed, https://developer.android.com/training/basics/network-ops/xml#analyze

2. Create a target data structure.

```
public class Country {
    public String name = "";
    public String iso2Code = "";
    public String incomeLevelIso2Code = "";
    // Using floats just for kicks. In production we'd use doubles given that
    // Mobile CPU architectures are moving toward 64 bits and in that context
    // there's no speed benefit in using 32bit datatypes.
    public float longitude = 0f;
    public float latitude = 0f;
    public boolean isRegion = false;

    public Country(String name, String iso2Code, String incomeLevelIso2Code, float
longitude, float latitude) {
        this.name = name;
        this.iso2Code = iso2Code;
        this.incomeLevelIso2Code = incomeLevelIso2Code;
        this.longitude = longitude;
        this.latitude = latitude;
    }

    public Country(String name, String iso2Code, String incomeLevelIso2Code, float
longitude, float latitude, boolean isRegion) {
        this(name, iso2Code, incomeLevelIso2Code, longitude, latitude);
        this.isRegion = isRegion;
    }

    @Override
    public String toString() {
        String result = "";
        String format = "%s: %s%n";
        result += String.format(format, "name", this.name);
        result += String.format(format, "iso2Code", this.iso2Code);
        result += String.format(format, "incomeLevelIso2Code",
this.incomeLevelIso2Code);
        result += String.format(format, "longitude", this.longitude);
        result += String.format(format, "latitude", this.latitude);
        return result;
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\recyclerviewdemo\Country.java

3. Instantiate the Parser

```
// Depends on raw/country_data_from_world_bank.xml which was downloaded
//        From http://api.worldbank.org/v2/countries on 2018-04-06, as found on
//        https://datahelpdesk.worldbank.org/knowledgebase/articles/898590-api-
country-queries
public class CountryXmlImporter {
    public String getCountries(InputStream countriesXmlInputStream) throws IOException
{
        String result = "";
        BufferedReader bufferedReader = null;

        try {

            List<Country> countryList = parseXml(countriesXmlInputStream);
            result += String.format("Country List Length: %d%n", countryList.size());
            if (countryList != null) {
                for (ListIterator<Country> it = countryList.listIterator();
it.hasNext(); ) {
                    Country country = it.next();
```

```
                    result += String.format("%d: %s%n", it.previousIndex(),
country.toString());
                }
            } else {
                throw new Exception("countyList was null");
            }

            System.out.println(countryList);
        } catch (IOException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (bufferedReader != null) {
                bufferedReader.close();
            }
        }

        return result;
    }


    public List<Country> parseXml(InputStream inputStream) throws IOException {
        List<Country> countryList = null;
        XmlPullParser xmlPullParser = null;
        try {

            xmlPullParser = Xml.newPullParser();
            xmlPullParser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, true);
            xmlPullParser.setInput(inputStream, null);
            xmlPullParser.nextTag();
            countryList = readCountries(xmlPullParser);
        } catch (XmlPullParserException e) {
            e.printStackTrace();
        } finally {
            if (inputStream != null) inputStream.close();
        }
        return countryList;
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Connectivity, Performating network operations, Parse XML data, Instantiate the parser, https://developer.android.com/training/basics/network-ops/xml#instantiate

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\recyclerviewdemo\CountryXmlImporter.java

4. Having parsed a repeating node (e.g. a country) into your data structure (e.g. your Country class), something done in the next step, add that data structure instance to a collection (like a List<Country>). Optionally filter out repeating nodes that don't match some criteria (e.g. a country entry that is really a region like "Africa") before adding to your collection.

```
    private List<Country> readCountries(XmlPullParser xmlPullParser) throws
IOException, XmlPullParserException {
        List<Country> countriesList = new ArrayList<Country>();
        Country currentCountry = null;

        logXmlPullParserState(xmlPullParser);
        xmlPullParser.require(xmlPullParser.START_TAG, NAMESPACE, "countries");
        String tagName = "";
        while (xmlPullParser.next() != XmlPullParser.END_TAG) {
            if (xmlPullParser.getEventType() != XmlPullParser.START TAG) {
                continue;
            }
            tagName = xmlPullParser.getName();
            if (tagName.equals("country")) {
                currentCountry = readCountry(xmlPullParser);
                if (!currentCountry.isRegion) {
                    countriesList.add(currentCountry);
                }
            } else {
```

```
                skipEntireElement(xmlPullParser);
            }

        }
        return countriesList;
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Connectivity, Performating network operations, Parse XML data, Read the feed, https://developer.android.com/training/basics/network-ops/xml#read

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\recyclerviewdemo\CountryXmlImporter.java

5. Parse child elements and attributes of the repeating node (e.g a country) into your target data structure (e.g. your Country class).

```
    private Country readCountry(XmlPullParser xmlPullParser) throws IOException,
XmlPullParserException {

        String countryName = "";
        String iso2Code = "";
        String incomeLevelIso2Code = "";
        String region = "";
        boolean isRegion = false;
        float longitude = 0f;
        float latitude = 0f;

        String tagName = "";
        xmlPullParser.require(xmlPullParser.START_TAG, NAMESPACE, "country");
        try {

            while (xmlPullParser.next() != XmlPullParser.END_TAG) {
                if (xmlPullParser.getEventType() != XmlPullParser.START_TAG) continue;
                tagName = xmlPullParser.getName();
                switch (tagName) {
                    case "iso2Code":
                        iso2Code = readElementValueAsString(xmlPullParser, tagName);
                        break;
                    case "name":
                        countryName = readElementValueAsString(xmlPullParser,
tagName);
                        break;
                    case "incomeLevel":
                        // Fetching an attribute value
                        // Note the case format of "iso2code" in the xml source is
unconventional.
                        // That is,  it is not "iso2Code".
                        incomeLevelIso2Code =
readElementAttributeAsString(xmlPullParser, tagName, "iso2code");
                        break;
                    case "longitude":
                        longitude = readElementValueAsFloat(xmlPullParser, tagName);
                        break;
                    case "latitude":
                        latitude = readElementValueAsFloat(xmlPullParser, tagName);
                        break;
                    case "region":
                        // Mark the country as special in some way, where that the
country entry is
                        // a region e.g. "Africa" so that this special property can be
later used
                        // to skip adding this element to the list.
                        region = readElementValueAsString(xmlPullParser, tagName);
                        if (region.equals("Aggregates")) {
                            isRegion = true;
                        }
                        break;
                    default:
                        skipEntireElement(xmlPullParser);
                        //throw new Exception("Unexpected case in switch, no tagName
found for: " + tagName);
                        break;
```

```
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return new Country(countryName, iso2Code, incomeLevelIso2Code, longitude,
latitude, isRegion);
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Connectivity, Performating network operations, Parse XML data, Parse XML, https://developer.android.com/training/basics/network-ops/xml#parse

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\recyclerviewdemo\CountryXmlImporter.java

6. Parse individual elements and attributes.

```
    private String readElementValueAsString(XmlPullParser xmlPullParser, String
tagName) throws IOException, XmlPullParserException {
        String result = "";
        xmlPullParser.require(XmlPullParser.START_TAG, NAMESPACE, tagName);
        if (xmlPullParser.next() == XmlPullParser.TEXT) {
            result = xmlPullParser.getText();
            xmlPullParser.nextTag();
        }
        xmlPullParser.require(XmlPullParser.END_TAG, NAMESPACE, tagName);
        return result;
    }


    private float readElementValueAsFloat(XmlPullParser xmlPullParser, String tagName)
throws IOException, XmlPullParserException {
        float result = 0f;
        String valueAsString = "";
        xmlPullParser.require(XmlPullParser.START_TAG, NAMESPACE, tagName);
        if (xmlPullParser.next() == XmlPullParser.TEXT) {
            valueAsString = xmlPullParser.getText();
            if (!(valueAsString.equals("") || valueAsString == null)) {
                result = Float.parseFloat(valueAsString);
            }
            xmlPullParser.nextTag();
        }
        xmlPullParser.require(XmlPullParser.END_TAG, NAMESPACE, tagName);
        return result;
    }

    private String readElementAttributeAsString(XmlPullParser xmlPullParser, String
tagName, String attributeName) throws IOException, XmlPullParserException {
        String attributeValue = "";
        xmlPullParser.require(XmlPullParser.START_TAG, NAMESPACE, tagName);
        attributeValue = xmlPullParser.getAttributeValue(NAMESPACE, attributeName);
        xmlPullParser.next(); // Skip over element text;
        xmlPullParser.nextTag();
        xmlPullParser.require(XmlPullParser.END_TAG, NAMESPACE, tagName);
        return attributeValue;
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Connectivity, Performating network operations, Parse XML data, Parse XML, https://developer.android.com/training/basics/network-ops/xml#parse

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\recyclerviewdemo\CountryXmlImporter.java

7. Skip elements you don't care about.

```
    private void skipEntireElement(XmlPullParser xmlPullParser) throws
XmlPullParserException, IOException {
        if (xmlPullParser.getEventType() != XmlPullParser.START_TAG) {
            throw new IllegalStateException();
        }

        // To make sure that it stops at the correct END_TAG and not at the
        // first tag it encounters after the original START_TAG, it keeps track
        // of the nesting depth.
        int depth = 1;
        while (depth != 0) {
            switch (xmlPullParser.next()) {
                case XmlPullParser.END_TAG:
                    depth--;
                    break;
                case XmlPullParser.START_TAG:
                    depth++;
                    break;
            }
        }
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Connectivity, Performating network operations, Parse XML data, Skip tags you don't care about, https://developer.android.com/training/basics/network-ops/xml#skip

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\recyclerviewdemo\CountryXmlImporter.java

8. A handy logging function.

```
    private void logXmlPullParserState(XmlPullParser xmlPullParser) throws
XmlPullParserException {
        String message = "";
        message = String.format("xmlPullParser state: ... %n");
        message += String.format("  EventType: %s%n", xmlPullParser.getEventType());
        message += String.format("     Prefix: %s%n", xmlPullParser.getPrefix());
        message += String.format("   Tag name: %s%n", xmlPullParser.getName());
        Log.w(Globals.LOG_TAG, message);
    }
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\java\au\com\softmake\mythirdapp\recyclerviewdemo\CountryXmlImporter.java

Content Sharing

This section mostly references other parts of this document. It is based on:

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing files, Overview, https://developer.android.com/training/secure-file-sharing

Sharing Simple Data

Send simple data (text, images, etc) to other apps.

See Send (In Intents)

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing simple data, Sending simple data to other apps, https://developer.android.com/training/sharing/send

Add an Easy Share Action.

See Action Provider Procedure (ShareActionProvider example)

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing simple data, Sending simple data to other apps, https://developer.android.com/training/sharing/send

Sharing Files

Share files with other applications.

See File Provider

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, App data & files, Sharing files, Overview, https://developer.android.com/training/secure-file-sharing

Network

You can use the network to store and retrieve data on your own web-based services. To do network operations, use classes in the following packages: Java.net.*; and/or android.net.*

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core Topics, Connectivity, Performing network operations, https://developer.android.com/training/basics/network-ops

[TODO: Draft Section]

The options include:

- Use an asynchronous Loader. See Loaders
- Use a service.

  ... for example, a service might handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

  http://developer.android.com/guide/components/services.html#Foreground

- Use java.net.* and/or android.net.*

  (Google API Guides, 2013. Develop > API Guides) http://developer.android.com/guide/topics/data/data-storage.html#netw

- Use an AsyncTask.

  (Google Developer Guides, 2018), "Consume XML Data", https://developer.android.com/training/basics/network-ops/xml.html#consume

Sync Adapters: Syncing data locally and at a remote location:

> Add other optional pieces, such as sample data or an implementation of **AbstractThreadedSyncAdapter** that can synchronize data between the provider and cloud-based data.
>
> http://developer.android.com/guide/topics/providers/content-provider-creating.html

For working with network-based data, use classes in java.net and android.net. You can also synchronize network-based data to a local data store such as a database, and then offer the data as tables or files. The **Sample Sync Adapter** sample application demonstrates this type of synchronization.

http://developer.android.com/guide/topics/providers/content-provider-creating.html

See also Calendar Provider Sync Adapter. http://developer.android.com/guide/topics/providers/calendar-provider.html#sync-adapter

ContentProvider Synching: Add other optional pieces, such as sample data or an implementation of AbstractThreadedSyncAdapter that can synchronize data between the provider and cloud-based data.

http://developer.android.com/guide/topics/providers/content-provider-creating.html

Processes and Threads

Processes

By default all components (activities, services, receivers, providers) run in the same process and thread (the "main" or "UI" thread). Most applications should not change this.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Best practices, Performance, Processes and threads overview, https://developer.android.com/guide/components/processes-and-threads

However to make components start in a different process use the `android:process` attribute for a component, in `AndroidManifest.xml`.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Best practices, Performance, Processes and threads overview, https://developer.android.com/guide/components/processes-and-threads

The system shuts down processes when it needs more resources. This is prioritised (from most important and last killed) as follows:

- Foreground process
- Visible process
- Service process
- Background process
- Empty process

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Best practices, Performance, Processes and threads overview, https://developer.android.com/guide/components/processes-and-threads

For long running operations, especially those that may outlast an activity, best practice to to start a service for that operation, rather than create a worker thread.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Best practices, Performance, Processes and threads overview, https://developer.android.com/guide/components/processes-and-threads

Threads

Threads Intro

If an app performs intensive work in response to user interaction (network access or database queries) this will, unless you do something about it, risk blocking the main/UI thread. If this takes longing than 5 seconds than, in the worst case, an "application not responding" (ANR) dialog will appear. The user will be rightly fucked off.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Best practices, Performance, Processes and threads overview, Threads, https://developer.android.com/guide/components/processes-and-threads#Threads

The Android UI is *not* thread-safe.

Therefore there are two threading rules:

- Do not block the main/UI thread; and
- Do not access the Andriod UI toolkit (Views and Widgets) from outside the main/UI thread.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Best practices, Performance, Processes and threads overview, Threads, https://developer.android.com/guide/components/processes-and-threads#Threads

If you have operations that are not instantaneous then you should execute them in separate ("Background" or "Worker") threads that also allow you to not violate threading rule 2, either via:

- A Runnable.
- AsyncTask.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Best practices, Performance, Processes and threads overview, Threads, https://developer.android.com/guide/components/processes-and-threads#Threads

When the method you implement might be called from more than one thread you need that method to be thread-safe therefore you use:

- An in-process Service.
- A ContentProvider; or
- Interprocess Communication (IPC) using a Service (bindService).

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Best practices, Performance, Processes and threads overview, Threads, https://developer.android.com/guide/components/processes-and-threads#Threads

Runnable

To construct a Runnable that does not violate the two threading rules.

Call a new thread that receives a new runnable class implementing the run method.

Then access the main/UI thread safely with one of:

- `Activity.runOnUiThread(Runnable)`
- `View.post(Runnable)`
- `View.postDelayed(Runnable, long)`

```
public class ThreadsActivity extends ActionBarActivity {

    private ImageView mPictureImageView;

    /**
     * Requires in AndroidManifext.xml
     *  <uses-permission android:name="android.permission.INTERNET" />
     */
    private Bitmap loadImageFromNetwork(String url) {
        Bitmap bitmap = null;
        try {
            bitmap = BitmapFactory.decodeStream((InputStream) new URL(url).getContent());
        } catch (Exception e) {
            e.printStackTrace();
        }
        return bitmap;
    }

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_threads);

        mPictureImageView = (ImageView) findViewById(R.id.imageView_picture);
    }

    public void getPicture(View view) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                final Bitmap bitmap = loadImageFromNetwork("http://i.imgur." +
                                                           "com/NFR1Ydk.jpg?1");
                mPictureImageView.post(new Runnable() {
                    @Override
                    public void run() {
                        mPictureImageView.setImageBitmap(bitmap);
                    }
                });
            }
        }).start();
    }

}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Best practices, Performance, Processes and threads overview, Threads, Worker threads, https://developer.android.com/guide/components/processes-and-threads#WorkerThreads

AsyncTasks (Deprecated)

AsyncTask Basic Procedure

However Runnables get complex very quickly, and can't handle runtime configuration changes (like rotating the screen), so the recommendation is to use AsyncTasks instead.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Best practices, Performance, Processes and threads overview, Threads, Worker threads, https://developer.android.com/guide/components/processes-and-threads#WorkerThreads

AsyncTasks should ideally be used for short operations (a few seconds at the most.) If you need to keep threads running for long periods of time, it is highly recommended you use the various APIs provided by the `java.util.concurrent` pacakge such as `Executor`, `ThreadPoolExecutor` and `FutureTask`. Or use a service, loader, or content provider.

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), AsyncTask, https://developer.android.com/reference/android/os/AsyncTask

To use an AsyncTask:

- Create a nested class that extends `AsyncTask`.
- Specify the type of the parameters, the progress values, and the final value of the task, using generics.
- Implement `doInBackground()`, which runs a task in a background (separate) thread.
- Implment `onPostExecute()` which receives a result from doInBackground and runs in the UI thread (thereby not accessing it externally and not blocking it).
- Run the AsyncTask by calling `execute()`.

```
public class ThreadsActivity extends ActionBarActivity {
```

```
    private ImageView mPictureImageView;

    private class LoadImageFromNetworkAsyncTask extends AsyncTask<String, Void, Bitmap> {

        @Override
        protected Bitmap doInBackground(String... urls) {
            return loadImageFromNetwork(urls[0]);
        }

        @Override
        protected void onPostExecute(Bitmap bitmap) {
            mPictureImageView.setImageBitmap(bitmap);
        }
    }

    /**
     * Requires in AndroidManifext.xml
     *  <uses-permission android:name="android.permission.INTERNET" />
     */
    private Bitmap loadImageFromNetwork(String url) {
        Bitmap bitmap = null;
        try {
            bitmap = BitmapFactory.decodeStream((InputStream) new URL(url).getContent());
        } catch (Exception e) {
            e.printStackTrace();
        }
        return bitmap;
    }

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

        mPictureImageView = (ImageView) findViewById(R.id.imageView_picture);
    }

    public void getPicture(View view) {
        new LoadImageFromNetworkAsyncTask().execute("http://i.imgur.com/SikTbWe.jpg");
    }

}
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Best practices, Performance, Processes and threads overview, Threads, Worker threads, https://developer.android.com/guide/components/processes-and-threads#WorkerThreads

AsyncTask Notes:

- onPreExecute(), onPostExecute(), and onProgressUpdate() are all invoked on the UI thread.
- You can call publishProgress() at anytime in doInBackground() to execute onProgressUpdate() on the UI thread.
- You can cancel the task at any time, from any thread. `AsyncTask.canel(true).`

AsyncTask Full Procedure

In addition to the AsyncTask Basic Procedure above we need to handle runtime configuration changes (e.g. when the user rotates the screen, or places their device in a dock), otherwise data maybe lost (e.g. the loaded picture disappears). An AsyncTask can also provide progress updates which could update a progress bar. Our AsyncTask Full Procedure implements both of these features:

Start with AsyncTask Basic Procedure as above.

Add a nested class RetainedFragment that extends the Fragement class and doesn't have it's own UI. Add setRetainInstance(true) to the onCreate event of this Fragment. Provide procedures to set and get your data.

```
public class ThreadsActivity extends Activity {

    private ImageView mPictureImageView;
    private RetainedFragment mRetainedFragment = null;
    ...

    public static class RetainedFragment extends Fragment {

        private Bitmap mBitmap;

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

            // The key to making data survive runtime configuration changes.
            setRetainInstance(true);
        }

        public Bitmap getData() {
            return this.mBitmap;
        }

        public void setData(Bitmap bitmapToRetain) {
            this.mBitmap = bitmapToRetain;
        }
    }

    private class LoadImageFromNetworkAsyncTask extends AsyncTask<String, Integer,
            Bitmap> {
    ....
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App Basics, App resources, Handle configuration changes, Retaining an object during a configuration change, https://developer.android.com/guide/topics/resources/runtime-changes#RetainingAnObject

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\ThreadsActivity.java

In the outermost Activity class's onCreate() handle the RetainedFragment: Reference it if it already exists (in case the Activity is restarting); create and add it if it doesn't exist; Then, if it already existed, get data from the RetainedFragment and set your UI with that data.

```
public class ThreadsActivity extends Activity {

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

        final String retainedFragmentTag = "RetainedFragmentTag";

        mPictureImageView = (ImageView) findViewById(R.id.imageView_picture);
        mLoadingProgressBar = (ProgressBar) findViewById(R.id.progressBar_loading);

        // Find the RetainedFragment on Activity restarts
        FragmentManager fm = getFragmentManager();
        // The RetainedFragment has no UI so we must reference it with a tag.
        mRetainedFragment = (RetainedFragment) fm.findFragmentByTag(retainedFragmentTag);

        // if Retained Fragment doesn't exist create and add it.
        if (mRetainedFragment == null) {

            // Add the fragment
            mRetainedFragment = new RetainedFragment();
```

```
            fm.beginTransaction().add(mRetainedFragment, retainedFragmentTag).commit();

        // The Retained Fragment exists
        } else {

            mPictureImageView.setImageBitmap(mRetainedFragment.getData());
        }
    }
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App Basics, App resources, Handle configuration changes, Retaining an object during a configuration change, https://developer.android.com/guide/topics/resources/runtime-changes#RetainingAnObject

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\ThreadsActivity.java

Initiate the AsyncTask from the UI

```
        public void getPicture(View view) {
            mLoadingProgressBar.setVisibility(View.VISIBLE);
            new LoadImageFromNetworkAsyncTask().execute(
                    "http://i.imgur.com/SikTbWe.jpg");
        }
```

Add and code a determinate progress bar:

- Add a progress bar to the UI layout;
- Get a reference to it in the Activity oncreate();
- Make it visible and invisble at the start and end of the process;
- Define the progress to report to UI in onProgressUpdate.
- Change the AsyncTask 2nd Generic parameter from Void to a type that can handle progress updates (e.g. Integer).
- publishProgress at regular points in doInBackground().

AsyncTask Full Procedure, entire code

The entire code for AsyncTask Full Procedure:

Activity Layout.

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="au.com.softmake.mysecondapp.ThreadsActivity">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin">

        <ImageView
            android:id="@+id/imageView_picture"
            android:layout_width="300dp"
            android:layout_height="300dp"
            android:background="@android:color/black" />

        <Button
            android:id="@+id/button_get_picture"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true"
            android:layout_below="@id/imageView_picture"
            android:onClick="getPicture"
            android:text="Get Picture" />

        <Button
            android:id="@+id/button_clear_picture"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignBottom="@id/button_get_picture"
            android:layout_toEndOf="@id/button_get_picture"
            android:layout_toRightOf="@id/button_get_picture"
            android:onClick="clearPicture"
            android:text="Clear Picture" />

        <ProgressBar
            android:id="@+id/progressBar_loading"
            style="?android:attr/progressBarStyleHorizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/button_get_picture"
            android:progress="0"
            android:indeterminateOnly="false"
            android:visibility="invisible" />

    </RelativeLayout>
</ScrollView>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\res\layout\activity_threads.xml

The Activity with: subclassed AsyncTask inner class; subclassed RetainedFragment inner class that handles runtime configuration changes (e.g. when the user rotates the screen); and a determinate progress bar updating at regular intervals.

```
public class ThreadsActivity extends Activity {

    private ImageView mPictureImageView;
    private RetainedFragment mRetainedFragment = null;
    private ProgressBar mLoadingProgressBar;

    public static class RetainedFragment extends Fragment {

        private Bitmap mBitmap;

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

            // The key to making data survive runtime configuration changes.
            setRetainInstance(true);
        }

        public Bitmap getData() {
            return this.mBitmap;
        }

        public void setData(Bitmap bitmapToRetain) {
            this.mBitmap = bitmapToRetain;
        }
    }

    private class LoadImageFromNetworkAsyncTask extends AsyncTask<String, Integer,
            Bitmap> {

        @Override
        protected Bitmap doInBackground(String... urls) {
            // Simulate a burdensome load.
            int sleepSeconds = 4;
            for (int i = 1; i <= sleepSeconds; i++) {
                SystemClock.sleep(1000); // milliseconds
                publishProgress(i * 20); // Adjust for a scale to 100
```

```
            }

            return com.example.standardapplibrary.android.Network
                    .loadImageFromNetwork(urls[0]);
        }

        @Override
        protected void onProgressUpdate(Integer... progress) {
            mLoadingProgressBar.setProgress(progress[0]);
        }

        @Override
        protected void onPostExecute(Bitmap bitmap) {
            publishProgress(100);
            mRetainedFragment.setData(bitmap);
            mPictureImageView.setImageBitmap(bitmap);
            mLoadingProgressBar.setVisibility(View.INVISIBLE);
            publishProgress(0);
        }
    }


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

        final String retainedFragmentTag = "RetainedFragmentTag";

        mPictureImageView = (ImageView) findViewById(R.id.imageView_picture);
        mLoadingProgressBar = (ProgressBar) findViewById(R.id.progressBar_loading);

        // Find the RetainedFragment on Activity restarts
        FragmentManager fm = getFragmentManager();
        // The RetainedFragment has no UI so we must reference it with a tag.
        mRetainedFragment = (RetainedFragment) fm.findFragmentByTag(retainedFragmentTag);

        // if Retained Fragment doesn't exist create and add it.
        if (mRetainedFragment == null) {

            // Add the fragment
            mRetainedFragment = new RetainedFragment();
            fm.beginTransaction().add(mRetainedFragment, retainedFragmentTag).commit();

            // The Retained Fragment exists
        } else {

            mPictureImageView.setImageBitmap(mRetainedFragment.getData());
        }
    }

    public void getPicture(View view) {
        mLoadingProgressBar.setVisibility(View.VISIBLE);
        new LoadImageFromNetworkAsyncTask().execute("http://i.imgur.com/SikTbWe.jpg");
    }

    public void clearPicture(View view) {
        mRetainedFragment.setData(null);
        mPictureImageView.setImageBitmap(null);
    }
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\java\au\com\softmake\mysecondapp\ThreadsActivity.java

In our example the function that does the actual work is as follows

```
public static Bitmap loadImageFromNetwork(String url) {
    Bitmap bitmap = null;
    try {
        bitmap = BitmapFactory.decodeStream((InputStream) new URL(url).getContent());
    } catch (Exception e) {
        e.printStackTrace();
    }
```

```
    return bitmap;
}
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\standardapplibrary\src\main\java\au\com\softmake\standardapplibrary\android\Network.java

Add any permissions that your background task requires to the AndroidManifest.xml

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

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\AndroidManifest.xml

Add your Activity to AndroidManifest.xml

```
<manifest>
...
    <application>
        <activity
            android:name=".ThreadsActivity"
            android:label="@string/title_activity_threads"
            android:parentActivityName=".MainActivity">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="au.com.softmake.mysecondapp.MainActivity" />
        </activity>
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MySecondApp\app\src\main\AndroidManifest.xml

Parrallel (Concurrent) Execution

If you truly want parallel execution, you can invoke executeOnExecutor(java.util.concurrent.Executor, Object[]) with THREAD_POOL_EXECUTOR.

(Google, n.d., "Android API Reference > Android Platorm > Packages", https://developer.android.com/reference/packages), AsyncTask, https://developer.android.com/reference/android/os/AsyncTask

Todo: expand.

Asynchronous operations

The following section references the various ways you can effect an asynchronous operation. An asynchronous operation is an operation that performs its work in the background without blocking the execution of main, foreground, operations. This section is just an overview. It'll help you reference other parts of this document.

You can make Asynchronous calls in the following places:

- Loaders (using a CursorLoader, a subclass of Loader, or a subclass of AsyncTaskLoader);
- Services (Service started by IntentService)
- A runnable.
- AsyncTask or HandlerThread.
- AsyncQueryHandler for use with a ContentResolver.

  In practice, this should be done in an asynchronous thread instead of on the main thread. For more discussion, see Loaders. If you are not just reading data but modifying it, see AsyncQueryHandler.

  http://developer.android.com/guide/topics/providers/calendar-provider.html#query

When using a Loaders load data either by: using a CursorLoader, a subclass of Loader, or a subclass of AsyncTaskLoader.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Activities, Loaders, https://developer.android.com/guide/components/loaders

Unless you need to provide regular feedback to a user for a background operation (e.g. through a determinate progress bar) you should prefer using a Loader with AsyncTaskLoader, rather than an AsyncTask.

Both AsyncTaskLoader and AsyncTask can handle runtime configuration changes but AsyncTaskLoader is simpler to implement.

See Using a Loader with a subclass of AsyncTaskLoader; and Threads

Example of using an AyncTask to download XML from the internet and parse it.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Connectivity, Performing network operations, Parse XML data, Consume XML data, https://developer.android.com/training/basics/network-ops/xml#consume

App Install Location

In the manifest specify `android:installLocation` to make explicit your intentions. The values are: internalOnly (the default if omitted); auto (prefers internal but can be moved by user to external); preferExternal.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="au.com.softmake.myfirstapp"
    android:installLocation="internalOnly"
 ...
```

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, App data & files, Storage Overview, https://developer.android.com/training/data-storage

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), App Basics, App manifest files, <manifest>, android:installLocation, https://developer.android.com/guide/topics/manifest/manifest-element#install

Potentional Security Holes

If it's important that only your own app is able to start one of your components, set the exported attribute to "false" for that component. Filtering intents won't do it.

(Google 2021, "Android Developers > Developer Guides", https://developer.android.com/guide), Core topics, Intents and intent filters, Overview, Receiving an implicit intent, https://developer.android.com/guide/components/intents-filters#Receiving

References (Zotero)

Banes, Chris. 2014. "AppCompat V21 — Material Design for Pre-Lollipop Devices!". Android developers blog. https://android-developers.googleblog.com/2014/10/appcompat-v21-material-design-for-pre.html. 2014-10-22.

Bentley, John. 2013. "JavaReference-Framework.Docx". C:\Users\John\Documents\Sda\Info\Java\KB\Reference\JavaReference-Framework.docx. 2013.

———. 2019. "AndroidJavaGradle-CustomLibrarySetup.Docx". AndroidJavaGradle-CustomLibrarySetup.docx. 2019.

———. 2021. "Android And Java SDA". C:\Users\John\Documents\CustomData\AppDataUserSaved\Brainforest\DevAndComputer\Android\AndroidAndJavaSda.pdb. 2021-03-08.

docx4java. 2012. "JAXB Can Be Made to Run on Android". https://www.docx4java.org/blog/2012/05/jaxb-can-be-made-to-run-on-android/. 2012-05-17.

Google. 2014. "Android Design Guide". Android Developers. https://developer.android.com/design. 2014.

———. 2018. "Android Developer Guides". https://developer.android.com/guide/index.html. 2018.

———. 2019. "Android Developers > Training > Android Developer Fundamentals > Codelabs for Android Developer Fundamentals". https://developer.android.com/courses/fundamentals-training/toc-v2. 2019.

———. 2020. "Android Developers > Android Studio > User Guide > Android Debug Bridge (Adb)". Android Developers. https://developer.android.com/studio/command-line/adb. 2020-09-10.

———. 2020. "App Widget Design Guidelines". Android Developers. https://developer.android.com/guide/practices/ui_guidelines/widget_design. 2020-08-31.

———. 2021. "Android Developers > Developer Guides". Android Developers. https://developer.android.com/guide. 2021.

———. 2021. "Google Play > Best Practices (Strategies)". https://developer.android.com/distribute/best-practices. 2021.

———. n.d. "Android API Reference > Andoid Support Library". Accessed 2021-02-19. https://developer.android.com/reference/android/support/classes. n.d.

———. n.d. "Android API Reference > Android Platorm > Packages". Android Developers Reference. https://developer.android.com/reference/packages. n.d.

———. n.d. "Android Developers Blog". https://android-developers.googleblog.com/. n.d.

———. n.d. "Android Studio User Guide". https://developer.android.com/studio/intro. n.d. Citation Key: google-a.

———. n.d. "Google Issue Tracker". https://issuetracker.google.com/. n.d.

———. n.d. "Material Design". *Material Design*. https://material.io/. n.d. Citation Key: unknown-a.

———. n.d. "Support Library | Android Developers". Accessed 2021-02-13. https://developer.android.com/topic/libraries/support-library. n.d.

Internet Assigned Numbers Authority. 2021. "Media Types". https://www.iana.org/assignments/media-types/media-types.xhtml. 2021-02-01.

JetBrains. 2014. "IntelliJ IDEA > Getting Started". https://www.jetbrains.com/help/idea/getting-started.html. 2014.

Lockwood, Alex. 2016. "An Introduction to Icon Animation Techniques". https://www.androiddesignpatterns.com/2016/11/introduction-to-icon-animation-techniques.html. 2016-11-29.

Stackoverflow. 2018. "Stackoverflow". https://stackoverflow.com/. 2018.

Stuetz, Andreas. 2013. **Android PagerSlidingTabStrip**. https://github.com/astuetz/PagerSlidingTabStrip. 2013.

Szabo, David. 2018. **Android Swipe Collapse Animation (Java)**. https://github.com/david-szabo97/Android-Swipe-Collapse-Animation. 2018.

Theah. 2011. "How to View the SQLite Database on Your Android Emulator". *Web and Android Development Tips* (blog). http://blog.kwyps.com/2011/07/how-to-view-sqlite-database-on-your.html. 2011.

W3C. The latest. "Scalable Vector Graphics (SVG) 1.1 (Second Edition)". https://www.w3.org/TR/SVG11/. The latest.

Document Licence