LA TROBE
UNIVERSITY

All kinds of clever

CSE2MAD

Mobile Application Development

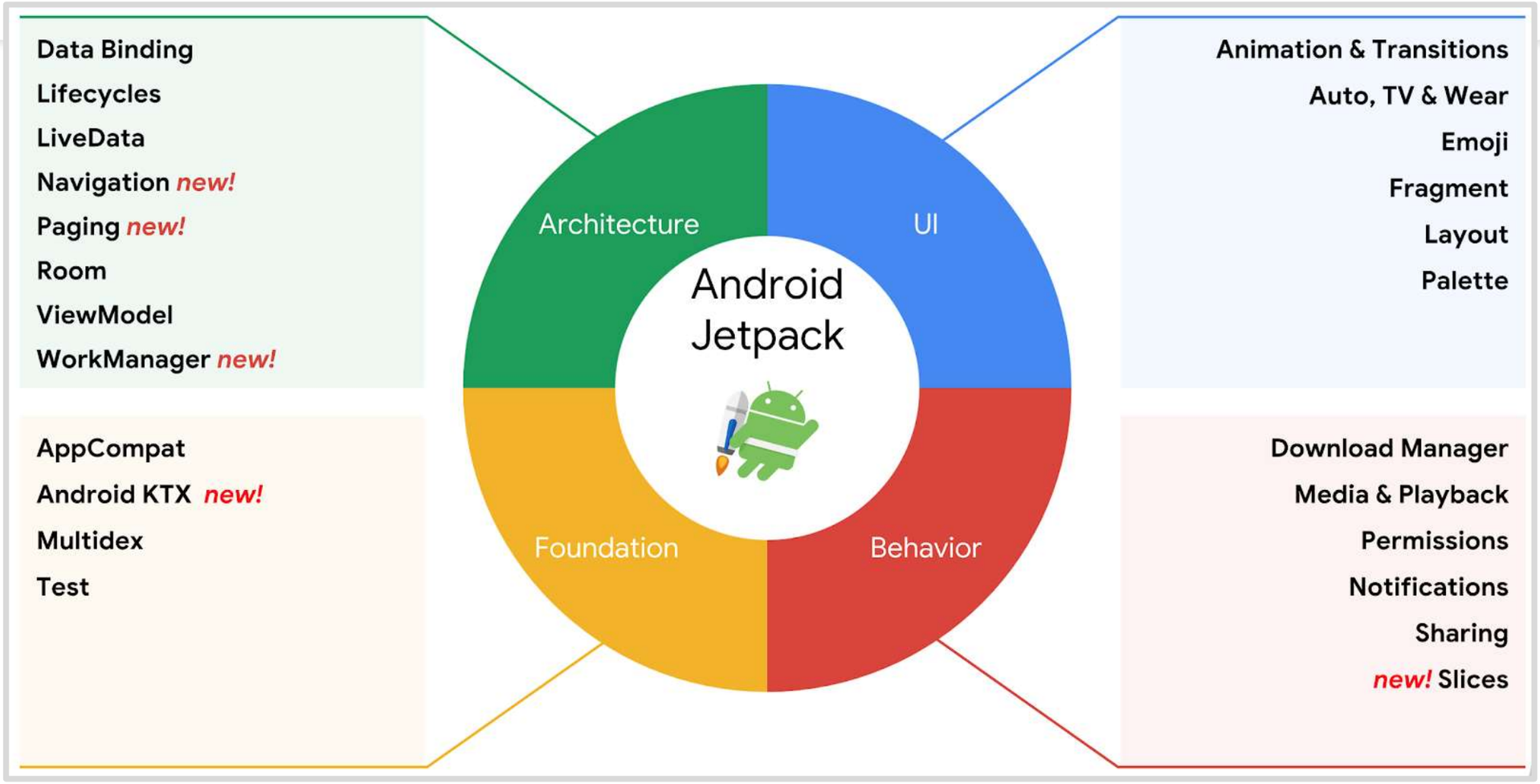
Lecture 11

Outline

- Android Jetpack
- Gradle
- Building Your App
- Preparing your app for Google Play

https://youtu.be/LmkKFCfmnhQ

Android Jetpack consists of Android Studio, the Android Architecture Components and Android Support Library together with a set of guidelines that recommend how an Android App should be structured.



https://android-developers.googleblog.com/2018/05/use-android-jetpack-to-accelerate-your.html

Smyth, N., *Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack*, Payload Media.

AndroidX Libraries

Artifacts within the androidx namespace comprise the Android Jetpack libraries. Like the Support Library, libraries in the androidx namespace ship separately from the Android platform and provide backward compatibility across Android releases.

https://developer.android.com/jetpack/androidx/versions#version-table

```
dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.1'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
```

https://android-developers.googleblog.com/2018/05/use-android-jetpack-to-accelerate-your.html

Modern Android Architecture Principles

- Single activity apps where different screens are loaded as content within the same activity.
- Modern architecture guidelines also recommend separating different areas of responsibility within an app into entirely separate modules (“**separation of concerns**”). One of the keys to this approach is the **ViewModel** component.
- Another important principle is that you should **drive your UI from a model**, preferably a persistent model. Models are components that are responsible for handling the data for an app. They're independent from the View objects and app components in your app, so they're unaffected by the app's lifecycle and the associated concerns.

https://developer.android.com/jetpack/guide

Smyth, N., *Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack*, Payload Media.

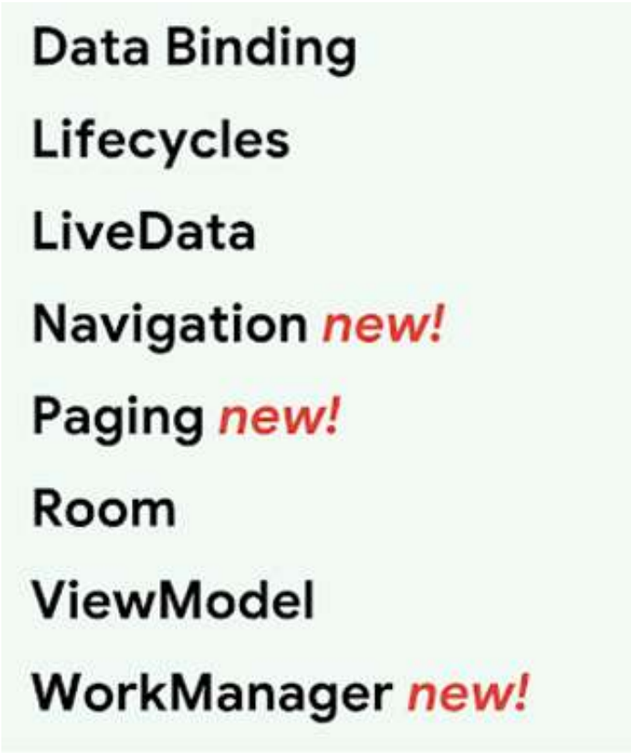
https://youtu.be/7p22cSzniBM

Android Architecture Components

The Android Architecture Components are designed to make it quicker and easier both to perform common tasks when developing Android apps while also conforming to the key principle of the architectural guidelines.

Android architecture components are a collection of libraries that help us in the following:

- Building the robust Android application.
- Building the testable Android application.
- Building the maintainable Android Apps.



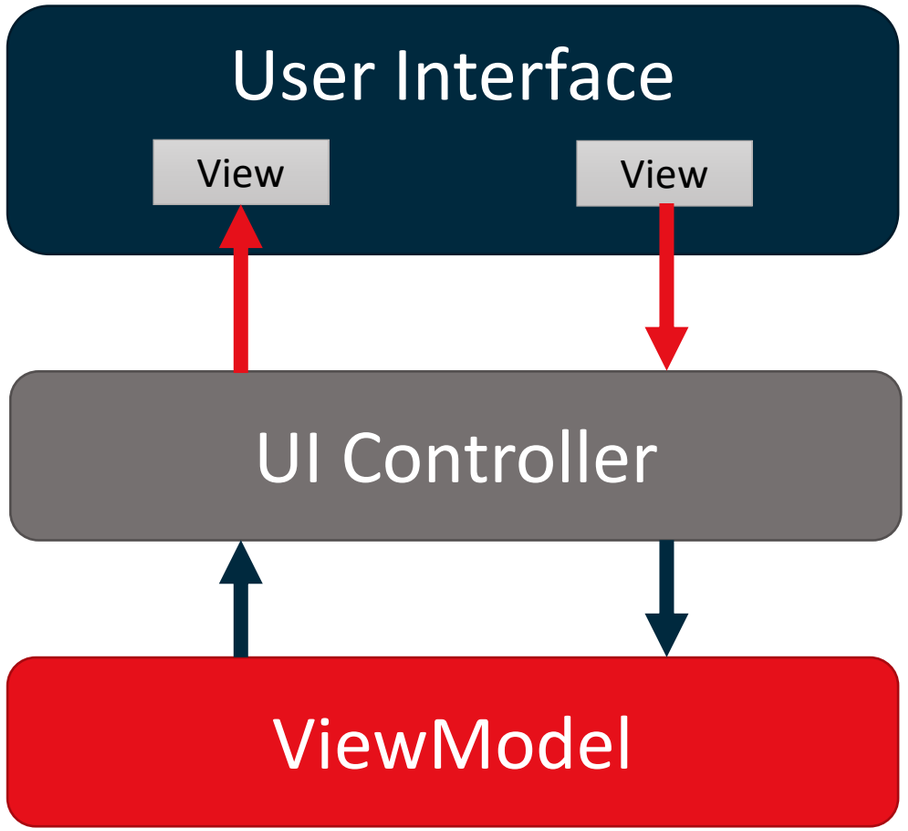
https://developer.android.com/studio/build

Smyth, N., *Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack*, Payload Media.

The ViewModel Component

The purpose of ViewModel is to separate the user interface-related data model and logic of an app from the code responsible for actually displaying and managing the user interface and interacting with the operating system.

When designed in this way, an app will consist of one or more UI Controllers, such as an activity, together with ViewModel instances responsible for handling the data needed by those controllers



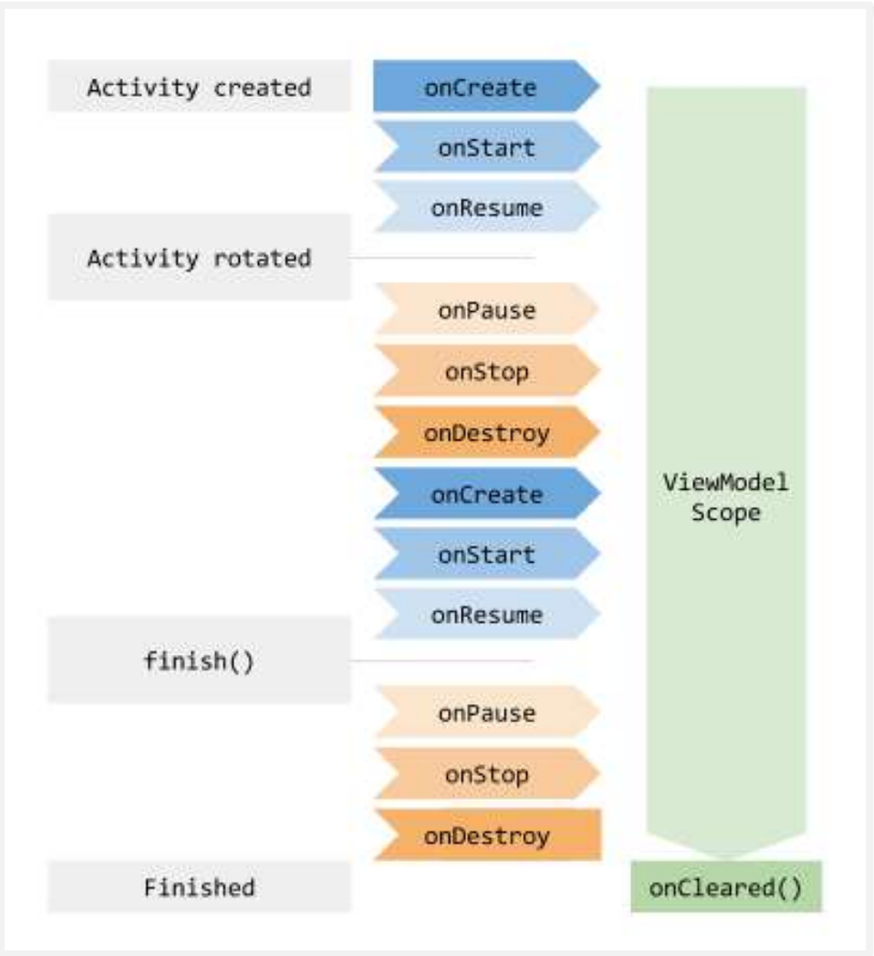
https://developer.android.com/studio/build

Smyth, N., *Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack*, Payload Media.

The lifecycle of a ViewModel

ViewModel objects are scoped to the Lifecycle passed to the ViewModelProvider when getting the ViewModel. The ViewModel remains in memory until the Lifecycle it's scoped to goes away permanently: in the case of an activity, when it finishes, while in the case of a fragment, when it's detached.

You usually request a ViewModel the first time the system calls an activity object's onCreate() method. The system may call onCreate() several times throughout the life of an activity, such as when a device screen is rotated. The ViewModel exists from when you first request a ViewModel until the activity is finished and destroyed.



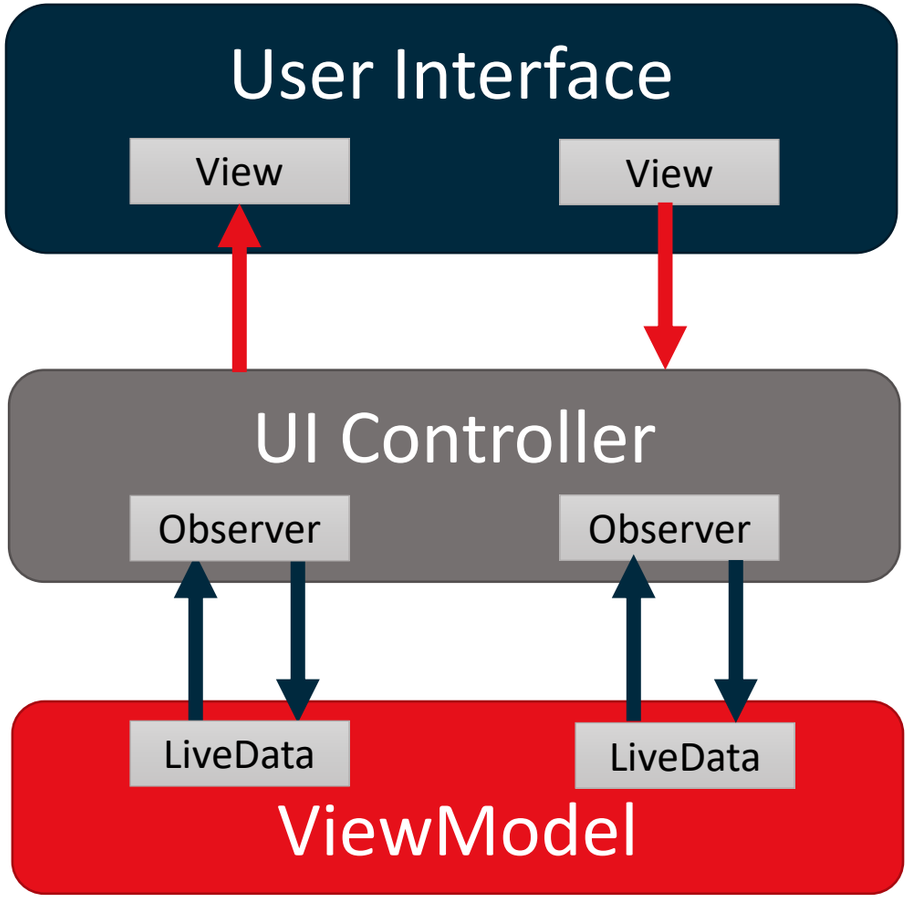
https://developer.android.com/topic/libraries/architecture/viewmodel#java

ViewModel Demo

https://github.com/latrobe-cs-educator/CSE2MAD_Lecture11_ViewModelDemo

The LiveData Component

LiveData is an observable data holder class. Unlike a regular observable, LiveData is lifecycle-aware, meaning it respects the lifecycle of other app components, such as activities, fragments, or services. This awareness ensures LiveData only updates app component observers that are in an active lifecycle state.



https://developer.android.com/topic/libraries/architecture/livedata

Smyth, N., *Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack*, Payload Media.

Android Jetpack Navigation component

The Navigation component consists of three key parts that are described below:

- **Navigation graph:** An XML resource that contains all navigation-related information in one centralized location. This includes all of the individual content areas within your app, called destinations, as well as the possible paths that a user can take through your app.

- **NavHost:** An empty container that displays destinations from your navigation graph. The Navigation component contains a default NavHost implementation, NavHostFragment, that displays fragment destinations.

- **NavController:** An object that manages app navigation within a NavHost. The NavController orchestrates the swapping of destination content in the NavHost as users move throughout your app.

https://developer.android.com/guide/navigation

Android Jetpack Navigation component

The Navigation component consists of three key parts that are described below:

- **Navigation graph:** An XML resource that contains all navigation-related information in one centralized location. This includes all of the individual content areas within your app, called destinations, as well as the possible paths that a user can take through your app.

- **NavHost:** An empty container that displays destinations from your navigation graph. The Navigation component contains a default NavHost implementation, NavHostFragment, that displays fragment destinations.

- **NavController:** An object that manages app navigation within a NavHost. The NavController orchestrates the swapping of destination content in the NavHost as users move throughout your app.

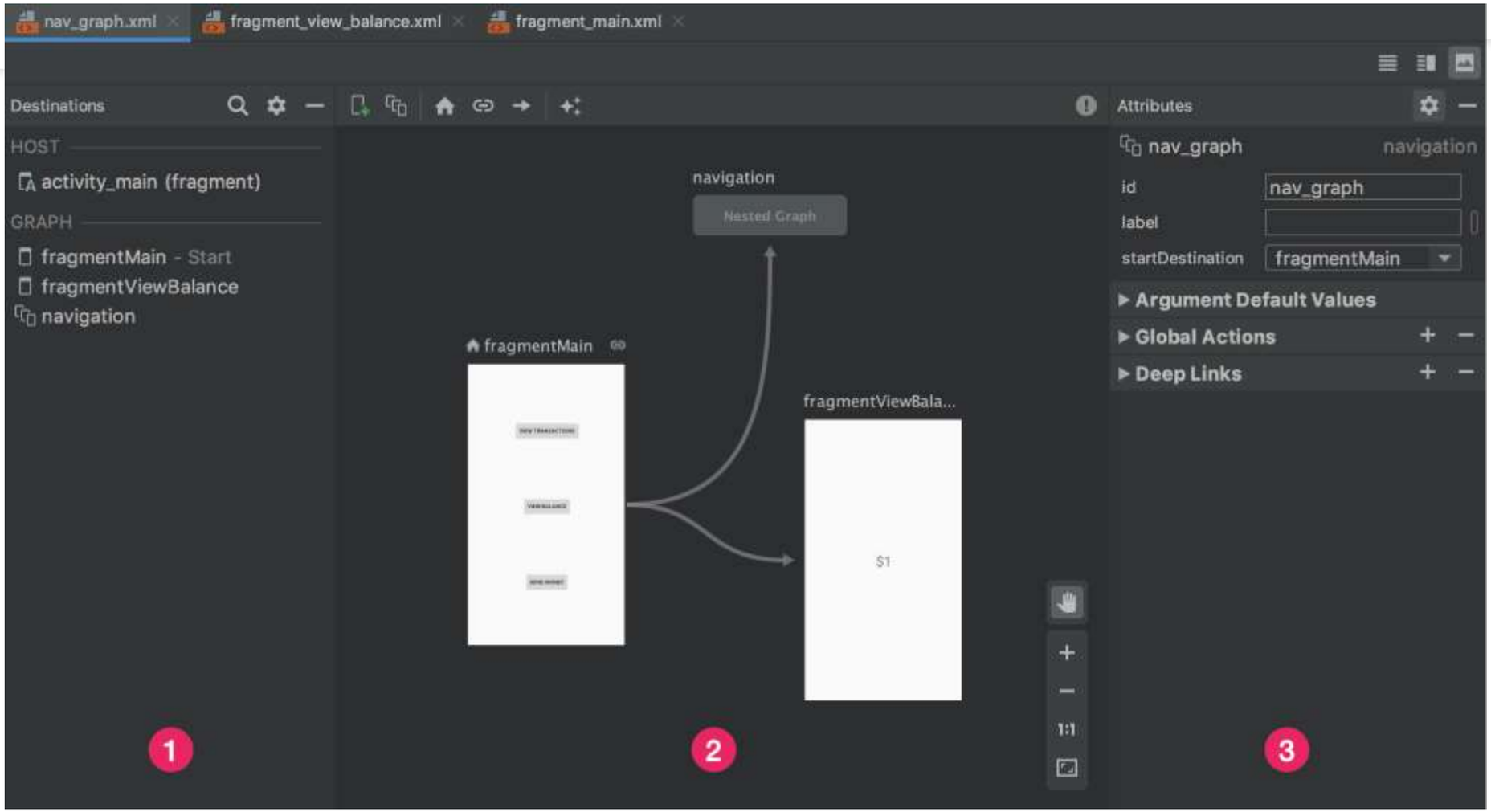
https://developer.android.com/guide/navigation

Navigation Editor

1. **Destinations panel:** Lists your navigation host and all destinations currently in the Graph Editor.

2. **Graph Editor:** Contains a visual representation of your navigation graph. You can switch between Design view and the underlying XML representation in the Text view.

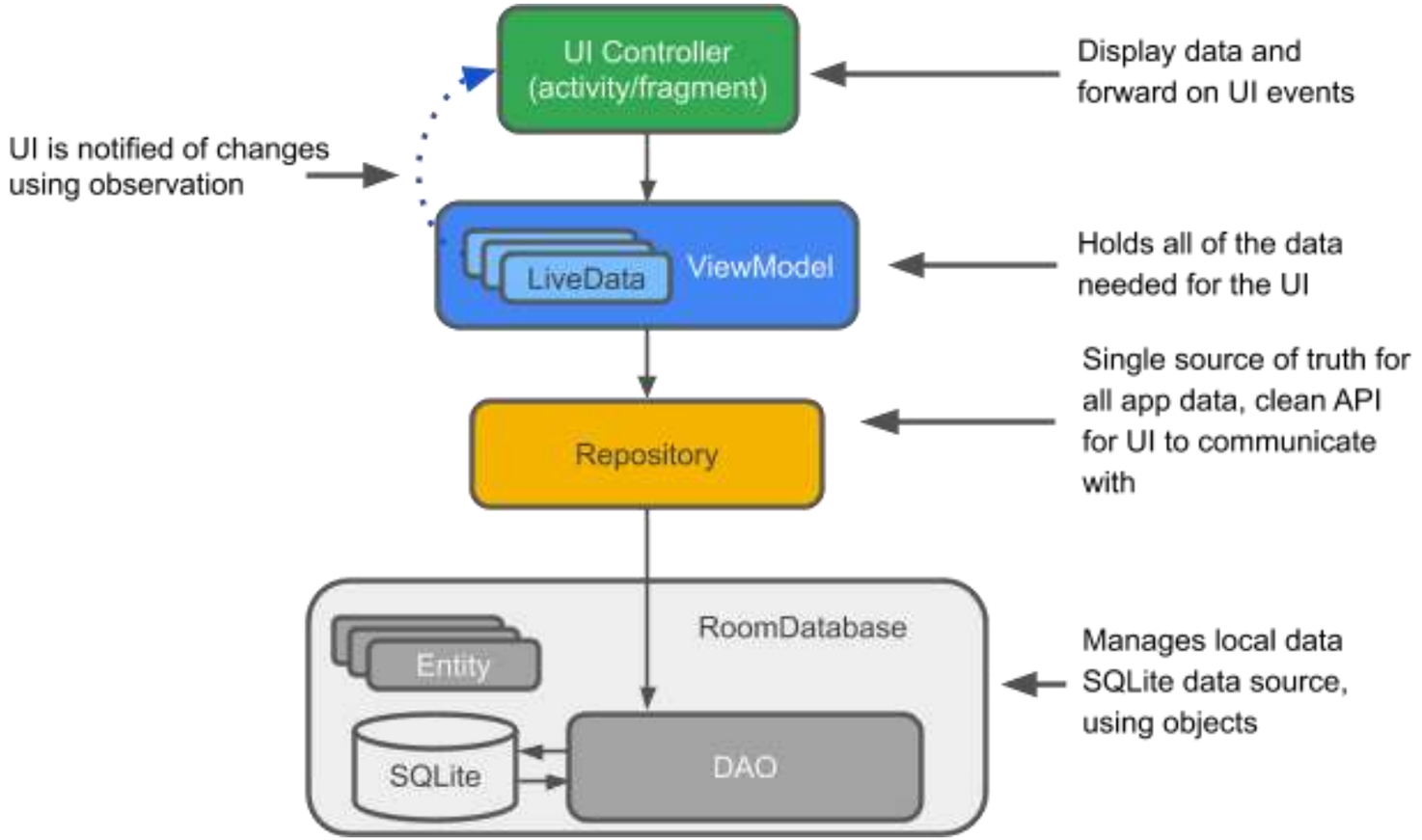
3. **Attributes:** Shows attributes for the currently-selected item in the navigation graph



https://developer.android.com/guide/navigation/navigation-getting-started#nav-editor

Room Persistence Library

The Room persistence library provides an abstraction layer over SQLite to allow for more robust database access while harnessing the full power of SQLite.



https://developer.android.com/topic/libraries/architecture/room Smyth, N., *Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack*, Payload Media.

Gradle is an automated build toolkit that allows the way in which projects are built to be configured and managed through a set of build configuration files. This includes defining how a project is to be built, what dependencies need to be fulfilled for the project to build successfully and what the end result (or results) of the build process should be.

Features

- **High performance** - Gradle avoids unnecessary work by only running the tasks that need to run because their inputs or outputs have changed.
- **JVM foundation** - Gradle runs on the JVM
- **Conventions** - Gradle makes common types of projects easy to build by implementing conventions.
- **Insight** - Build scans provide extensive information about a build run that you can use to identify build issues.

https://commons.wikimedia.org/wiki/File:Gradle_logo.png https://docs.gradle.org/current/userguide/what_is_gradle.html

Smyth, N., *Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack*, Payload Media.

Gradle & Android Studio features

Sensible Defaults

Gradle implements a concept referred to as *convention over configuration* . This simply means that Gradle has a pre-defined set of sensible default configuration settings that will be used unless they are overridden by settings in the build files.

Dependencies

Gradle dependencies can be categorized as local or remote. A local dependency references an item that is present on the local file system of the computer system on which the build is being performed. A **remote dependency** refers to an item that is present on a remote server (typically referred to as a repository).

```
dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.1'
    implementation 'com.firebaseui:firebase-ui-firestore:6.3.0'
    implementation 'com.google.firebase:firebase-firestore:21.6.0'
```

https://developer.android.com/studio/build

Smyth, N., *Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack*, Payload Media.

Gradle & Android Studio features

Build Variants

This allows multiple variations of an application to be built from a single project.

Manifest entries

You can specify values for some properties of the manifest file in the build variant configuration.

Code and resource shrinking

When building your app, the build system applies the appropriate set of rules to shrink your code and resources using its built-in shrinking tools.

Multiple APK support

The build system enables you to automatically build different APKs that each contain only the code and resources needed for a specific screen density or Application Binary Interface (ABI).

https://developer.android.com/studio/build

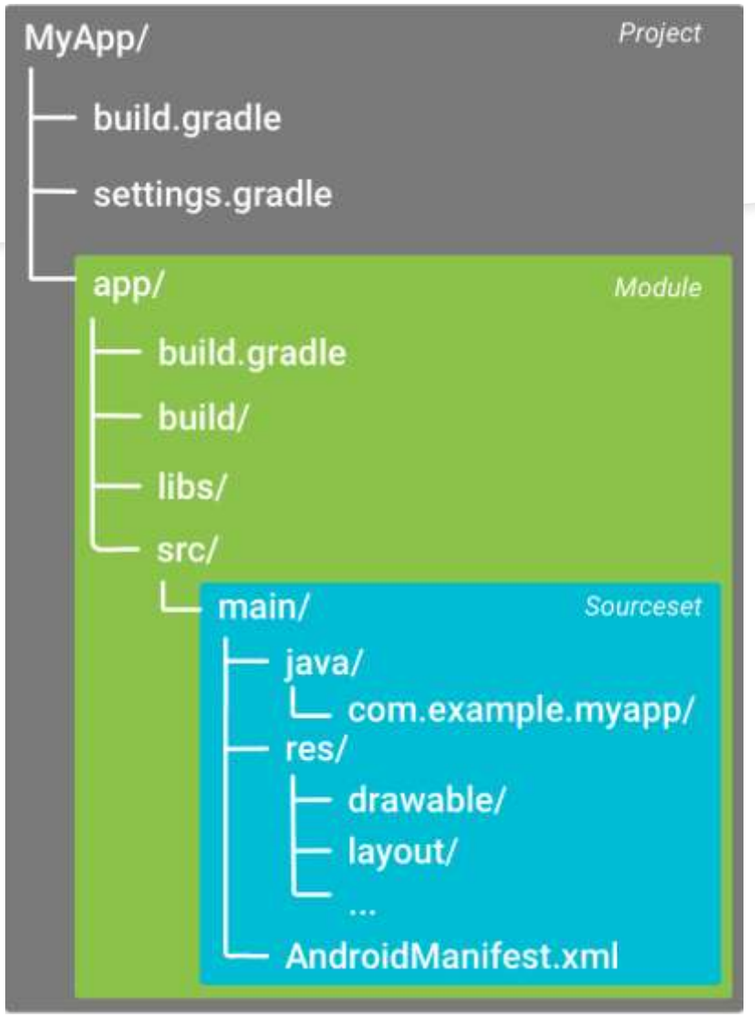
Smyth, N., *Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack*, Payload Media.

Build configuration files

Creating custom build configurations requires you to make changes to one or more build configuration files, or build.gradle files.

These plain text files use Domain Specific Language (DSL) to describe and manipulate the build logic using Groovy, which is a dynamic language for the Java Virtual Machine (JVM).

When starting a new project, Android Studio automatically creates some of these files for you, and populates them based on *sensible* defaults.



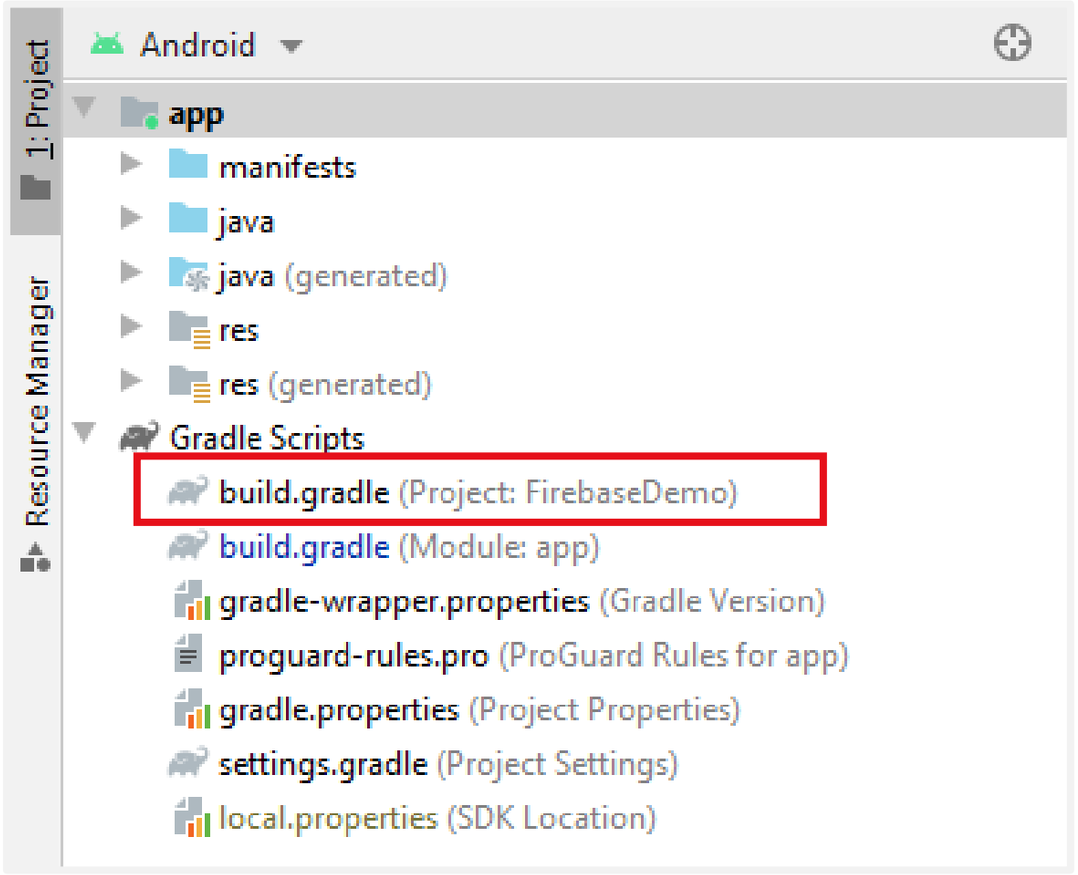
Default Project Structure

https://developer.android.com/studio/build#build-files

The Top-Level Gradle Build File

Each project contains one top-level Gradle build file. This file is listed as build.gradle (Project:) and can be found in the project tool window.

As it stands all the file does is declare that remote libraries are to be obtained using the jcenter repository and that builds are dependent on the Android plugin for Gradle. In most situations it is not necessary to make any changes to this build file.



Smyth, N., Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack, Payload Media.

The Top-Level Gradle Build File

build.gradle (Geofence)

```
1   // Top-level build file where you can add configuration options common
2   buildscript {
3       repositories {
4           google()
5           jcenter()
6       }
7       dependencies {
8           classpath "com.android.tools.build:gradle:4.0.1"
9
10          // NOTE: Do not place your application dependencies here; they
11          // in the individual module build.gradle files
12      }
13  }
14
15  allprojects {
16      repositories {
17          google()
18          jcenter()
19      }
20  }
21
22  task clean(type: Delete) {
23      delete rootProject.buildDir
24  }
```

The repositories block configures the repositories Gradle uses to search or download the dependencies. This The code below defines JCenter and Google's Maven repository as the repositories Gradle should use to look for its dependencies.

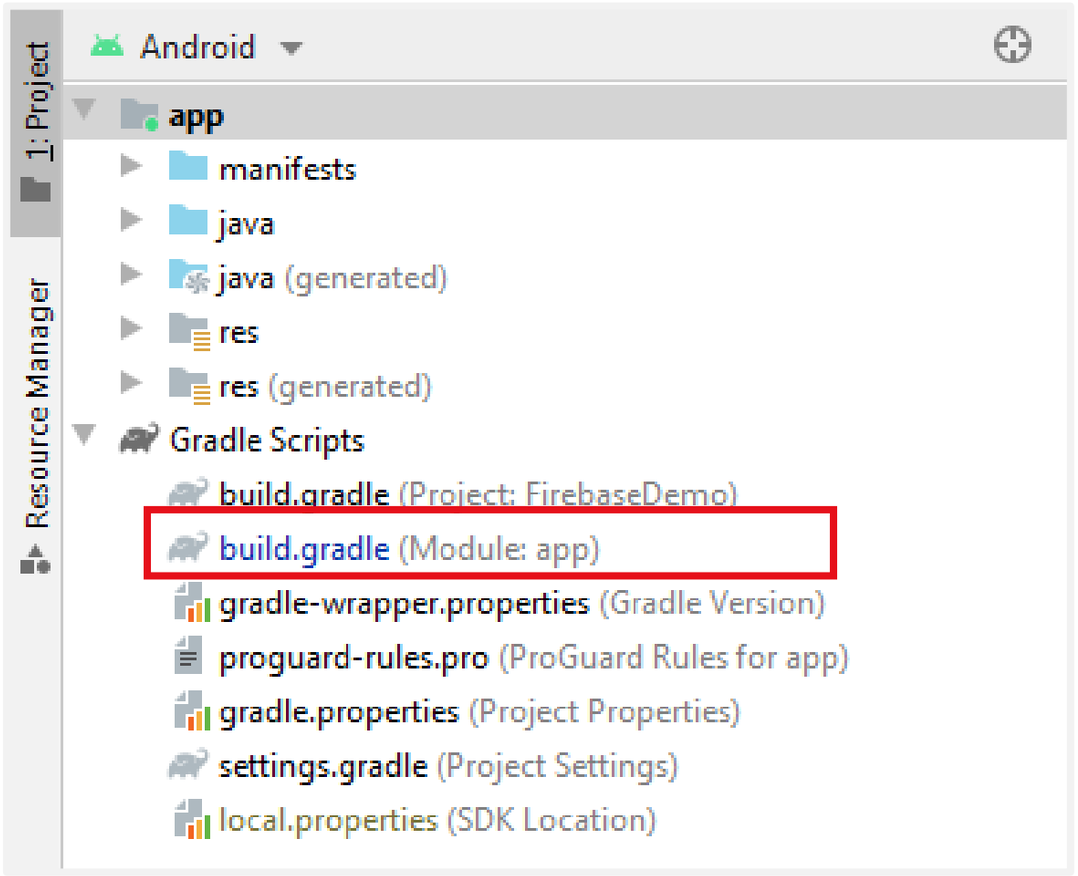
The dependencies block configures the dependencies Gradle needs to use to build your project.

The allprojects block is where you configure the repositories and dependencies used by all modules in your project, such as third-party plugins or libraries. However, you should configure module-specific dependencies in each module-level build.gradle file. For new projects, Android Studio includes JCenter and Google's Maven repository by default, but it does notconfigure any dependencies (unless you select a template that requires some).

The Module Level Gradle Build File

The module-level build.gradle file, located in each project/module/ directory, allows you to configure build settings for the specific module it is located in.

Configuring these build settings allows you to provide custom packaging options, such as additional build types and product flavors, and override settings in the main/ app manifest or top-level build.gradle file.



Smyth, N., Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack, Payload Media.

build.gradle (:app)

```
1  apply plugin: 'com.android.application'
2
3  android {
4      compileSdkVersion 30
5      buildToolsVersion "30.0.2"
6
7      defaultConfig {
8          applicationId "com.example.dateexamples"
9          minSdkVersion 25
10         targetSdkVersion 30
11         versionCode 1
12         versionName "1.0"
13
14         testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
15     }
16
17     buildTypes {
18         release {
19             minifyEnabled false
20             proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
21         }
22     }
23 }
24
25 dependencies {
26     implementation fileTree(dir: "libs", include: ["*.jar"])
27     implementation 'androidx.appcompat:appcompat:1.2.0'
28     implementation 'androidx.constraintlayout:constraintlayout:2.0.1'
29     testImplementation 'junit:junit:4.12'
30     androidTestImplementation 'androidx.test.ext:junit:1.1.2'
31     androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
32 }
```

the build file begins by declaring the use of the Gradle Android application plug-in

The android section of the file then states the version of the SDK to be used

The items declared in the defaultConfig section define elements that are to be generated into the module's AndroidManifest.xml file during the build. These settings, which may be modified in the build file, are taken from the settings entered within Android Studio when the module was first created

The buildTypes section contains instructions on whether and how to run ProGuard on the APK file when a release version of the application is built

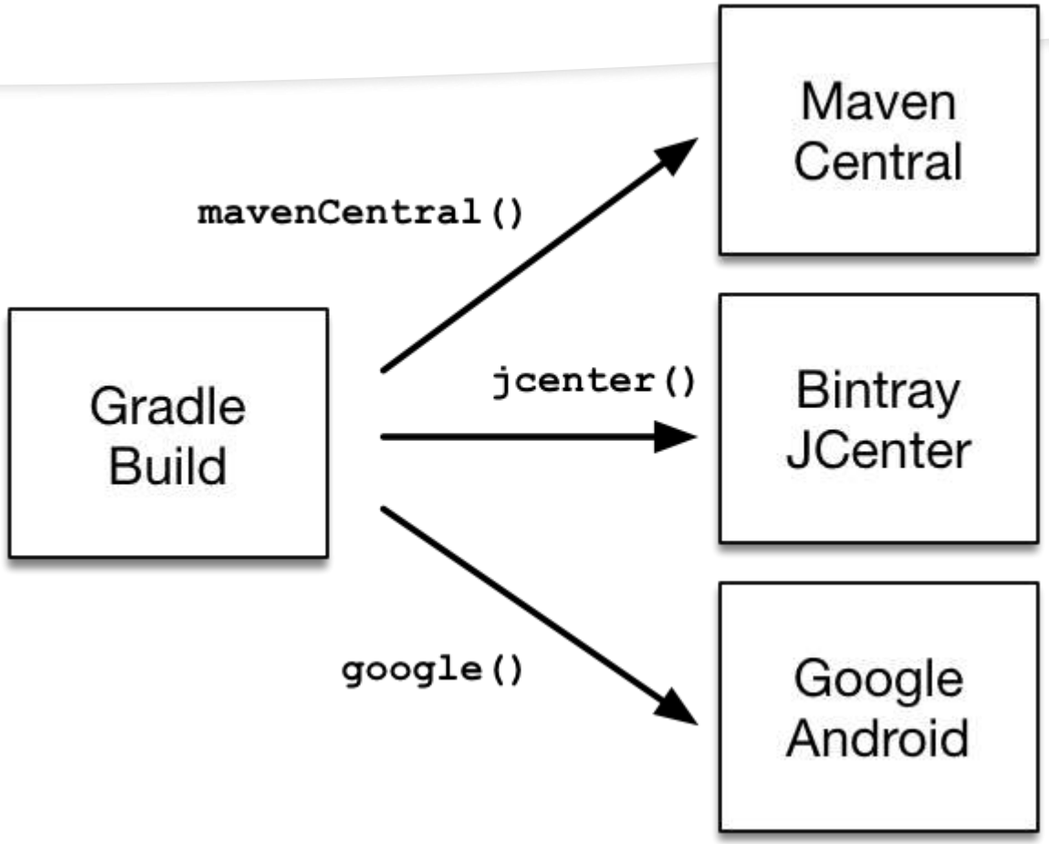
the dependencies section lists any local and remote dependencies on which the module is dependent.

Smyth, N., Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack, Payload Media.

External Gradle Dependencies

How does Gradle know where to find the files for external dependencies?

- Gradle looks for them in a repository. A repository is a collection of modules, organized by group, name and version.
- Gradle understands different repository types, such as Maven Central, Bintray JCenter and the Google Android repository.

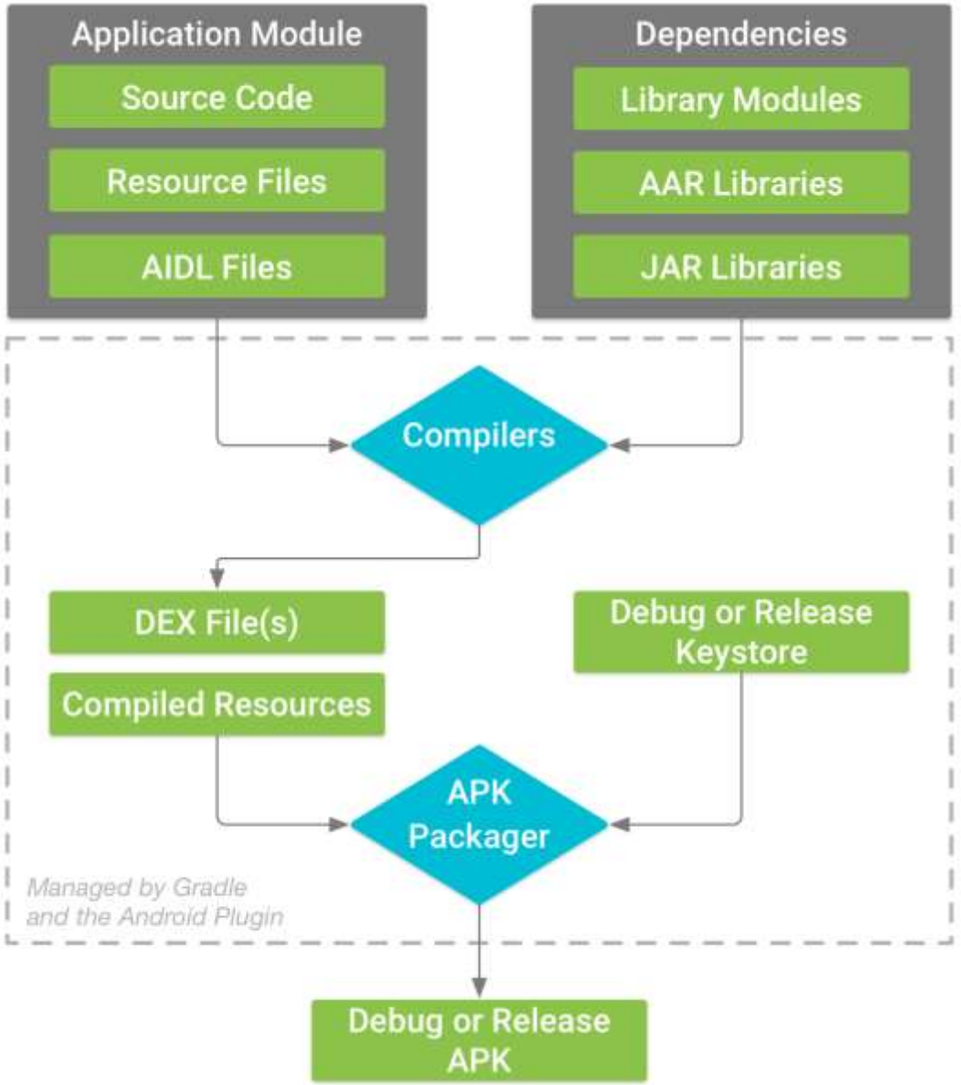


https://docs.gradle.org/current/userguide/declaring_repositories.html

Building your App

The build process for a typical Android app module, follows these general steps:

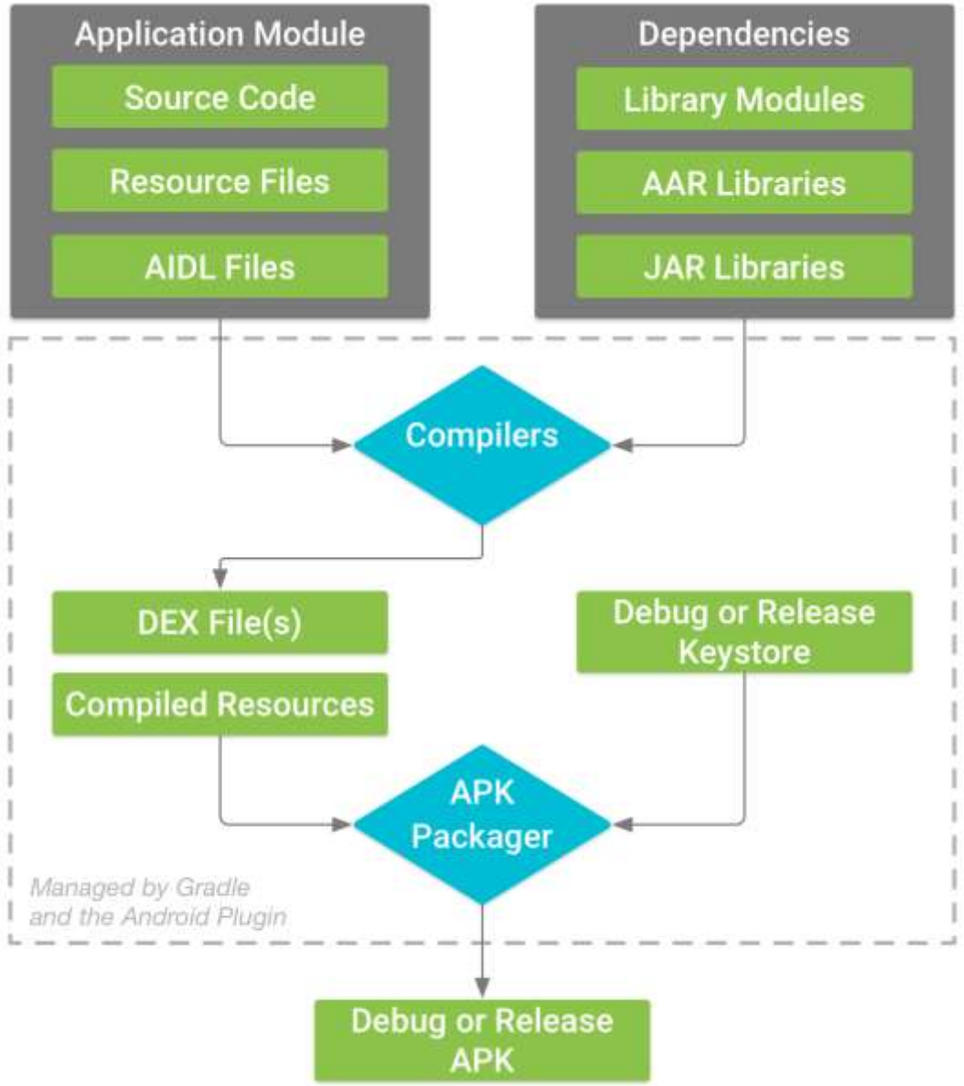
1. The compilers convert your source code into DEX (Dalvik Executable) files, which include the bytecode that runs on Android devices, and everything else into compiled resources.
2. The APK Packager combines the DEX files and compiled resources into a single APK. Before your app can be installed and deployed onto an Android device, however, the APK must be signed.



https://developer.android.com/studio/build

3. The APK Packager signs your APK using either the debug or release keystore:
    a. If you are building a debug version of your app, that is, an app you intend only for testing and profiling, the packager signs your app with the debug keystore. Android Studio automatically configures new projects with a debug keystore.
    b. If you are building a release version of your app that you intend to release externally, the packager signs your app with the release keystore.
4. Before generating your final APK, the packager uses the zipalign tool to optimize your app to use less memory when running on a device.

At the end of the build process, you have either a debug APK or release APK of your app that you can use to deploy, test, or release to external users.



https://developer.android.com/studio/build

Old distribution formats

When a user installs an app from Google Play, the app is downloaded in the form of an APK file. This file contains everything needed to install and run the app on the user's device.

Prior to the introduction of Android Studio 3.2, the developer would generate one or more APK files using Android Studio and upload them to Google Play. In order to support multiple device types, screen sizes and locales this would require either the creation and upload of **multiple APK files customized(slow)** for each target device and locale, or the generation of a **large universal APK containing(less sale conversions)** all of the different configuration resources and platform binaries within a single package.

Smyth, N., Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack, Payload Media.

The New distribution format

An **Android App Bundle** is a publishing format that includes all your app's compiled code and resources, and defers APK generation and signing to Google Play.

Google Play uses your app bundle to generate and serve optimized APKs for each device configuration, so only the code and resources that are needed for a specific device are downloaded to run your app. You no longer have to build, sign, and manage multiple APKs to optimize support for different devices, and users get smaller, more-optimized downloads.

Important: In the second half of 2021, new apps will be required to publish with the Android App Bundle on Google Play.

https://developer.android.com/guide/app-bundle

Smyth, N., Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack, Payload Media.

https://youtu.be/9D63S4ZRBls

Dynamic Delivery

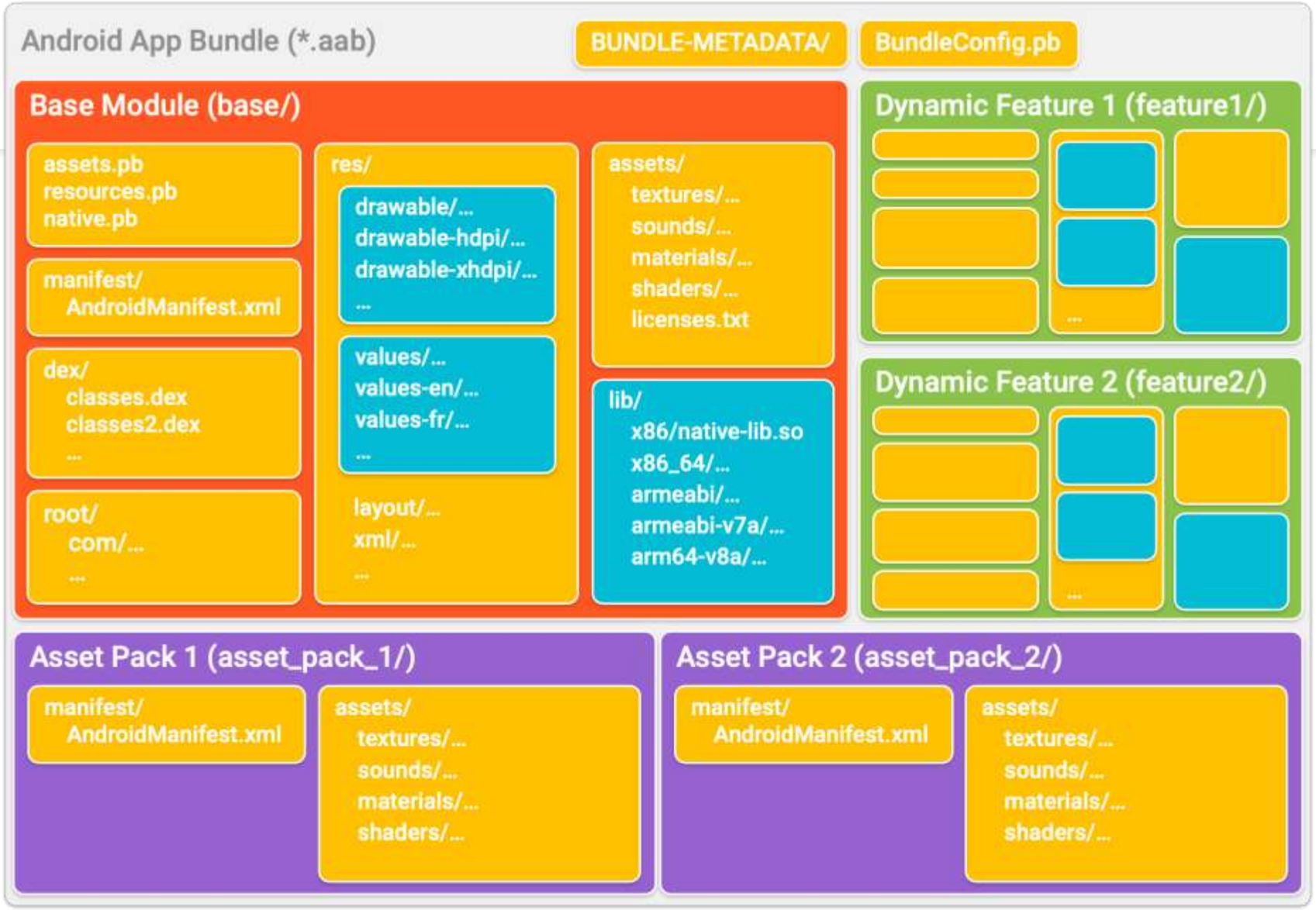
Dynamic Delivery provides a way for developers to create a single package from within Android Studio and have custom APK files automatically generated by Google Play for each individual supported configuration.

When a user installs the app, Google Play receives information about the user's device including the display, processor architecture and locale. Using this information, the appropriate pre-generated APK files are transferred onto the user's device.

An additional benefit of Dynamic Delivery is the ability to split an app into multiple modules, referred to as **dynamic feature modules**, where each module contains the code and resources for a particular area of functionality within the app. Each dynamic feature module is contained within a separate APK file from the base module and is downloaded to the device only when that feature is required by the user.

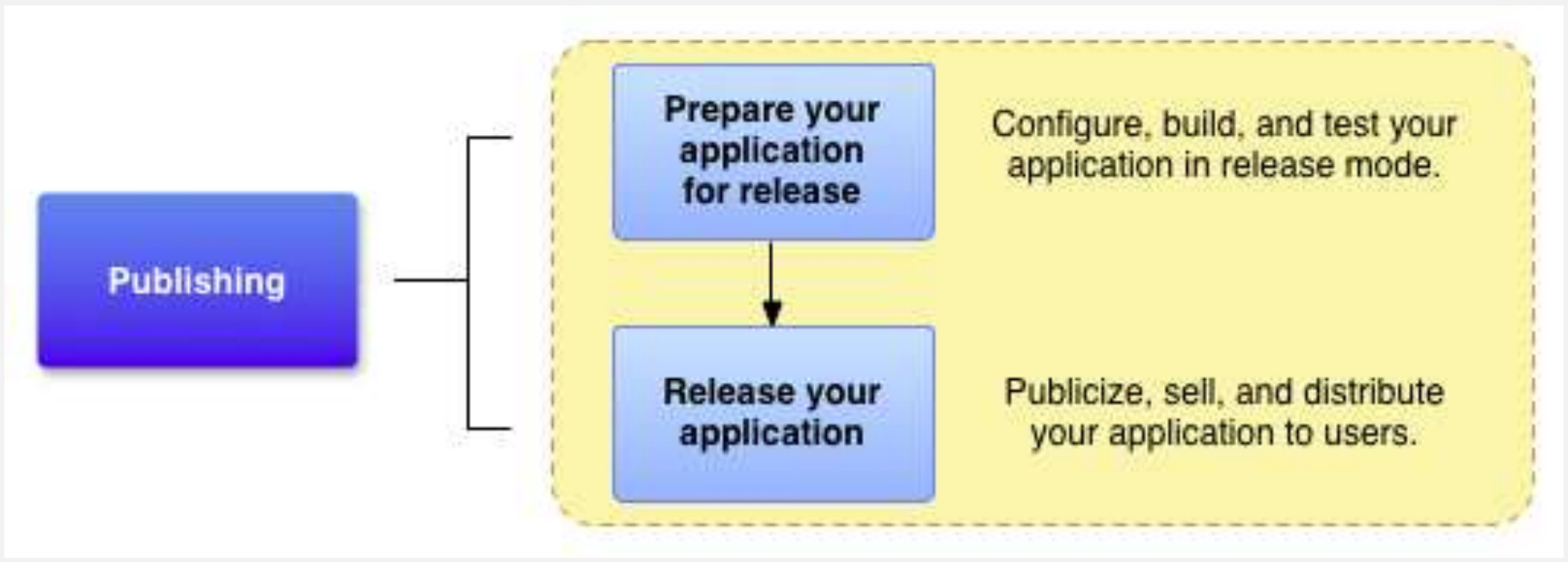
Smyth, N., Android Studio 4. 0 Development Essentials - Java Edition : Developing Android Apps Using Android Studio 4. 0, Java and Android Jetpack, Payload Media.

An Android App Bundle is a file (with the .aab file extension) that you upload to Google Play.



https://developer.android.com/guide/app-bundle

Preparing your App for Release



https://developer.android.com/studio/publish/preparing

Register for a Google Play Developer Console Account

The first step in the application submission process is to create a Google Play Developer Console account.

https://play.google.com/apps/publish/signup/

Note that there is a one-time $25USD fee to register. Once an application goes on sale, Google will keep 30% of all revenues associated with the application.

Gathering materials and resources

Cryptographic keys

The Android system requires that each installed application be digitally signed with a certificate that is owned by the application's developer (that is, a certificate for which the developer holds the private key).

Application icon

Your app will require an Application icon that must meet Googles Icon Giodelines

https://material.io/design/iconography/product-icons.html#design-principles

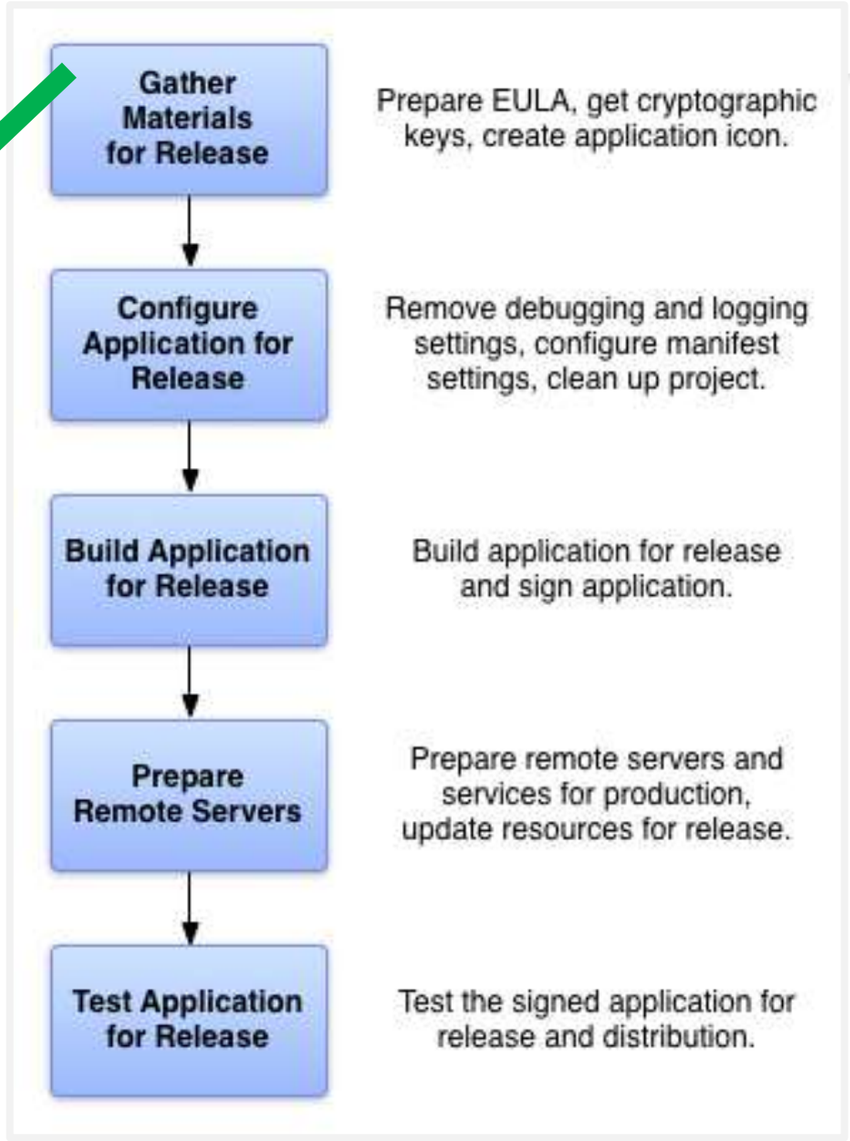
End-user license agreement

Consider preparing an End User License Agreement (EULA) for your application.

Miscellaneous materials

You will also have to prepare promotional and marketing materials to publicize your application which again have to meet certain specifications.

https://support.google.com/googleplay/android-developer/answer/1078870

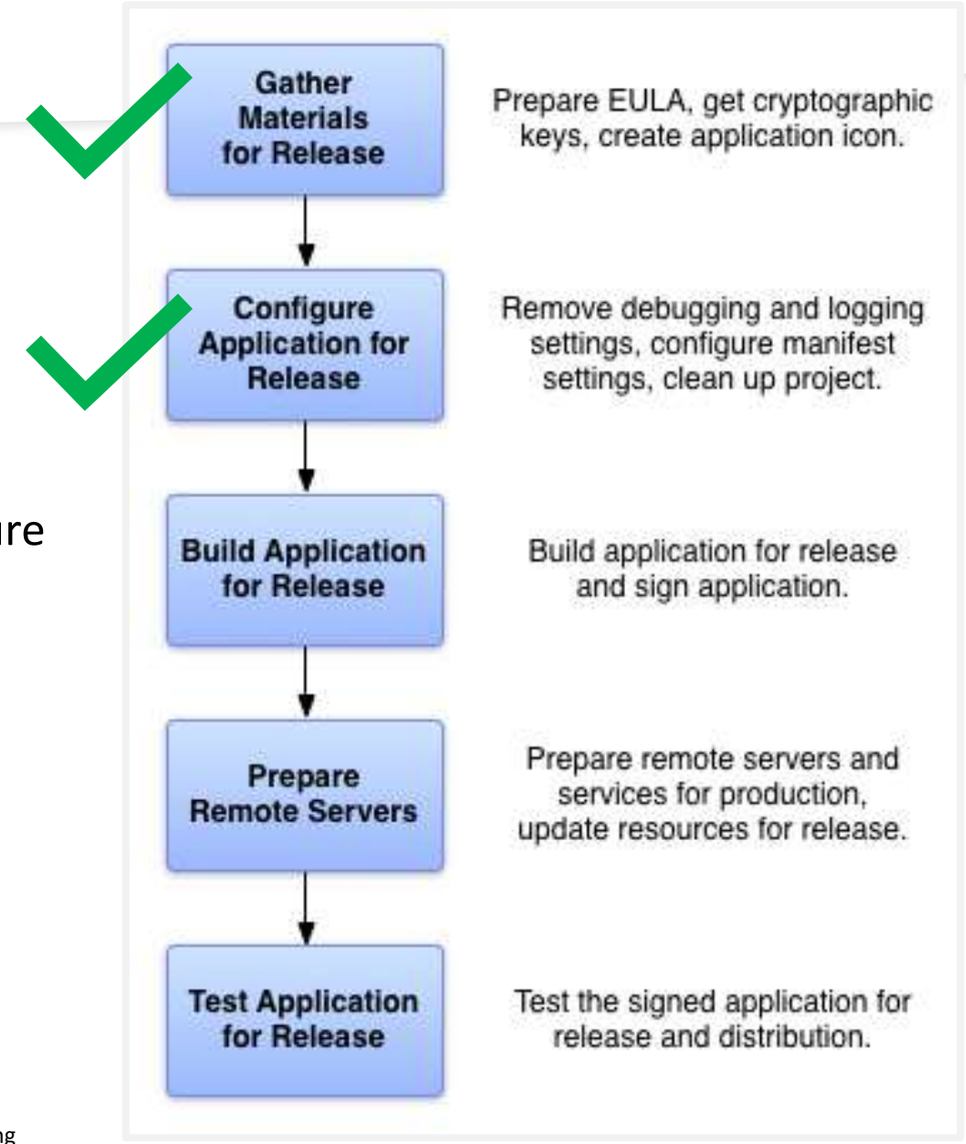


https://developer.android.com/studio/publish/preparing

Configuring Your application for release

- Choose a good package name
- Turn off logging and debugging
- Clean up your project directories
  - Ensure your project conforms to the recommended project structure

    https://developer.android.com/studio/projects
- Review and update your manifest and Gradle build settings
- Address compatibility issues
- Update URLs for servers and services

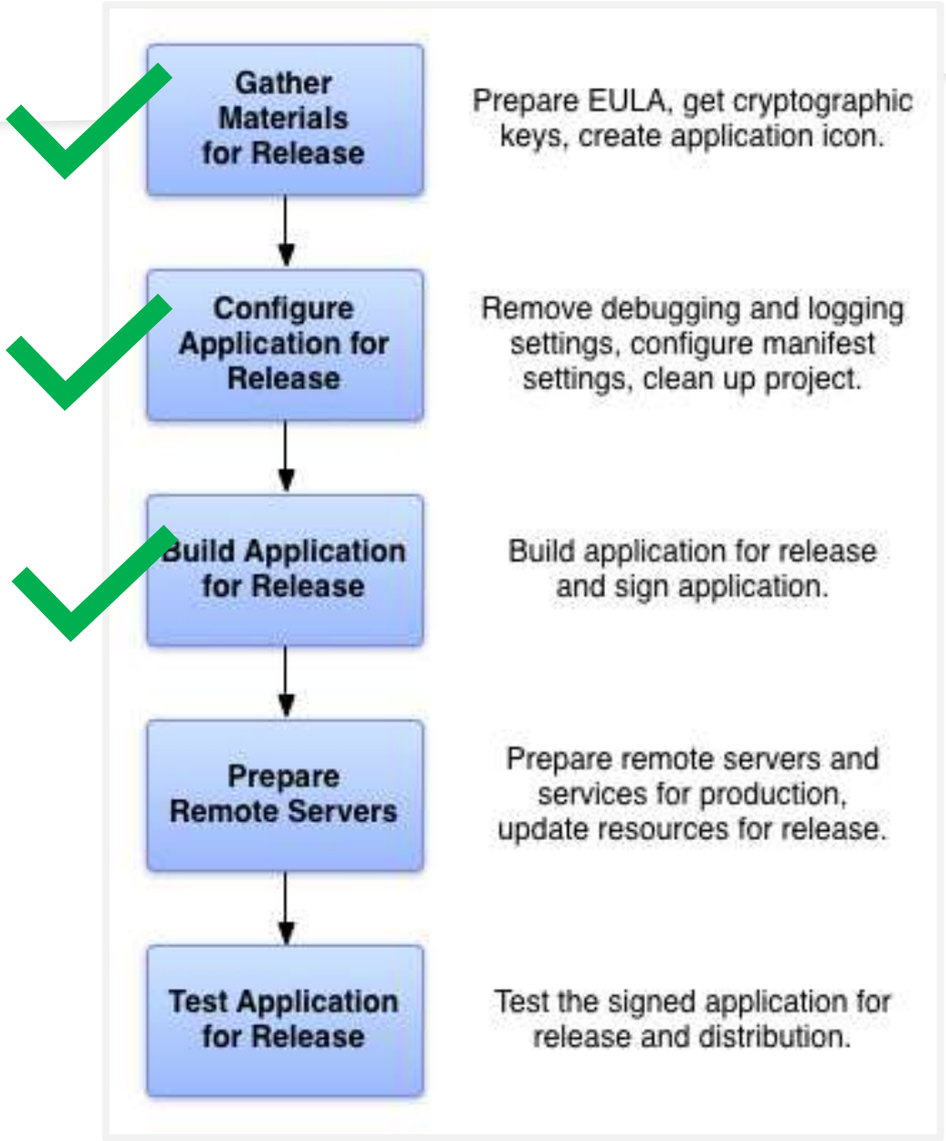


https://developer.android.com/studio/publish/preparing

Build application for release

The recommended way to upload files is to now use a process referred to as Google Play App Signing . For a newly created app, this involves opting in to Google Play App Signing and then generating an **private upload key** that is used to sign the app bundle file within Android Studio.

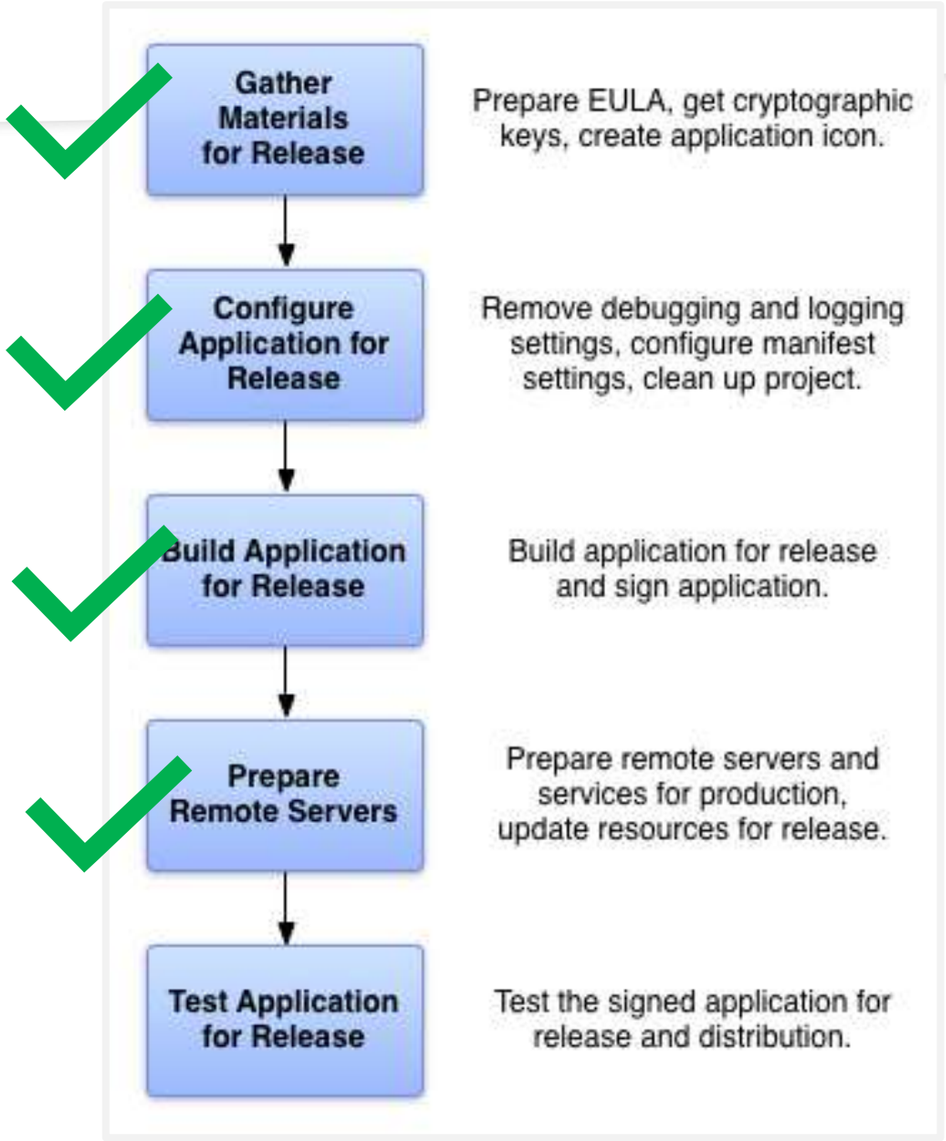
The next step is to instruct Android Studio to build the application app bundle file in release mode and then sign it with the newly created **private key**.



https://developer.android.com/studio/publish/preparing

Preparing external servers and resources

- If your application relies on a remote server, make sure the server is secure and that it is configured for production use.
- If your application fetches content from a remote server or a real-time service (such as a content feed), be sure the content you are providing is up to date and production-ready.

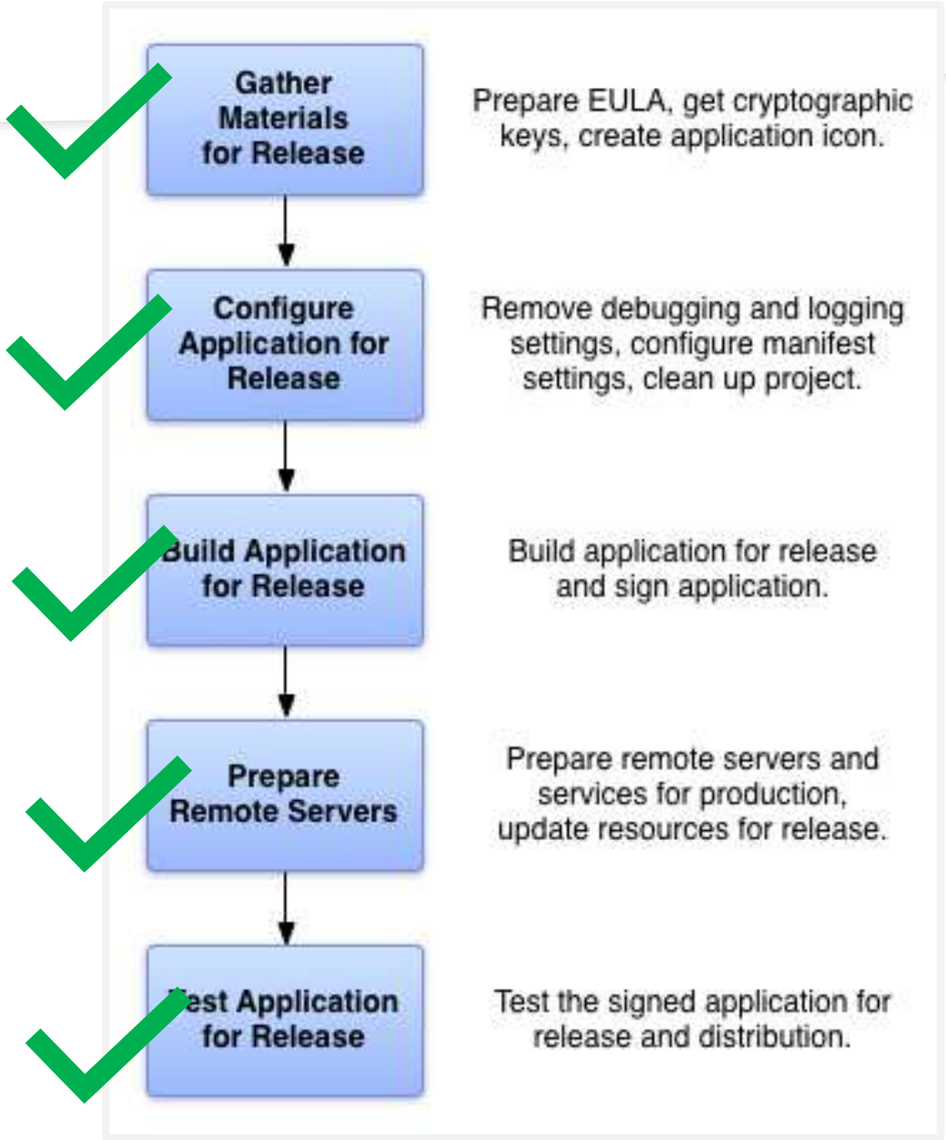


https://developer.android.com/studio/publish/preparing

Testing your application for release

Testing the release version of your application helps ensure that your application runs properly under realistic device and network conditions. Ideally, you should test your application on at least one handset-sized device and one tablet-sized device to verify that your user interface elements are sized correctly and that your application's performance and battery efficiency are acceptable.

https://developer.android.com/docs/quality-guidelines/core-app-quality



https://developer.android.com/studio/publish/preparing

Distributing your app

As an open platform, Android offers choice. You can distribute your Android apps to users in any way you want, using any distribution approach or combination of approaches that meets your needs.

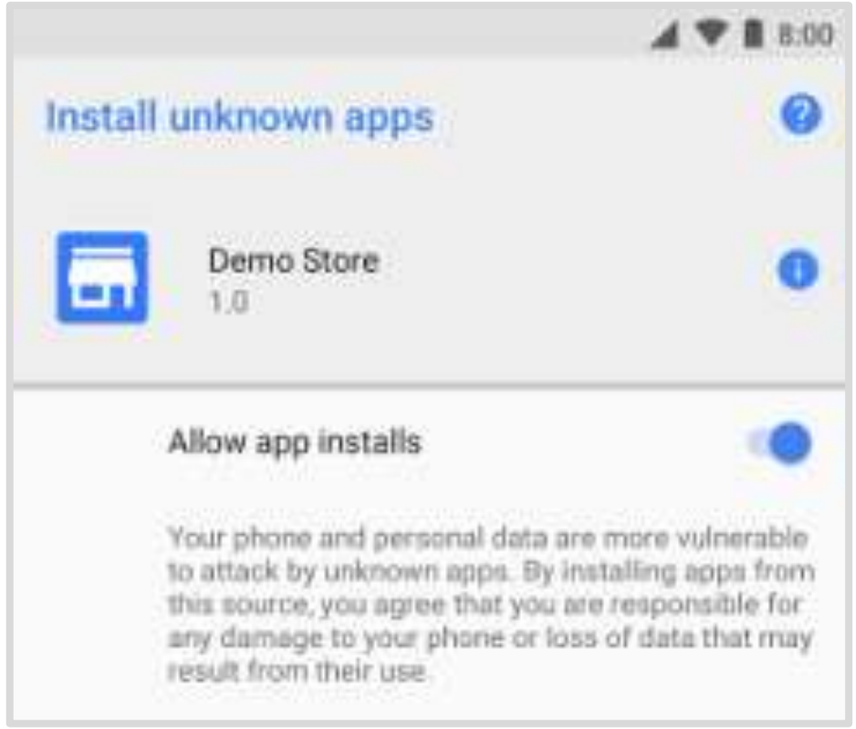
- Distributing through an app marketplace
- Distributing your apps by email*
- Distributing through a website*

The process for building and packaging your apps for distribution is the same, regardless of how you distribute them.

* Requires users to opt-in for installing unknown apps

https://developer.android.com/distribute/marketing-tools/alternative-distribution

User opt-in for installing unknown apps



Android protects users from inadvertent download and install of unknown apps, or apps from sources other than Google Play, which is trusted.

Android blocks such installs until the user opts into allowing the installation of apps from other sources.

Note: Some network providers don't allow users to install applications from unknown sources!

https://developer.android.com/distribute/marketing-tools/alternative-distribution#unknown-sources

LA TROBE
UNIVERSITY
All kinds of clever

CSE2MAD

Mobile Application Development

Lecture 11 Live