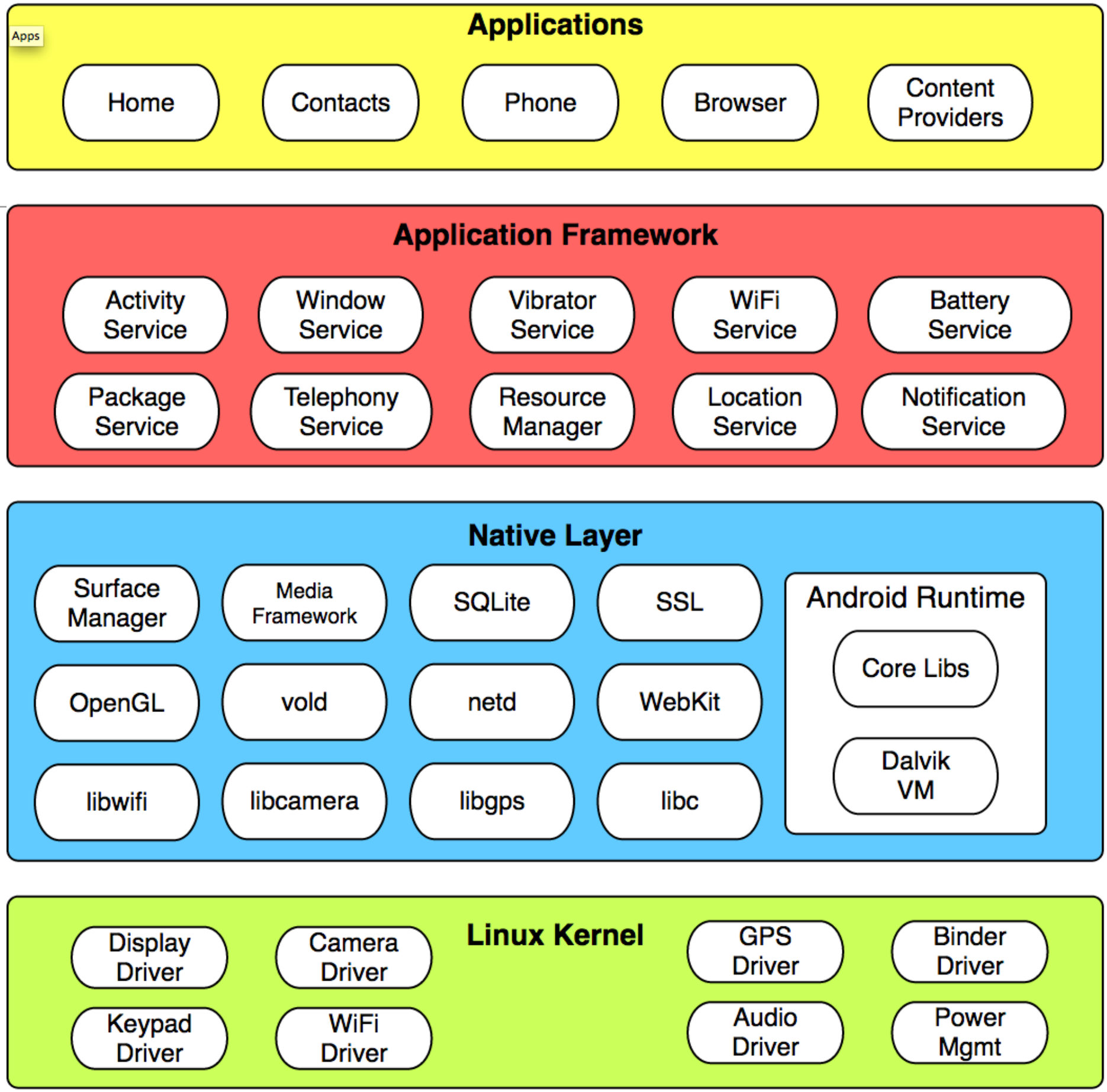
The Android Stack

Overview

- The Android operating system is like a cake consisting of various layers.
- Each layer has its own characteristics and purpose—but the layers are not always cleanly separated and often seep into one another.



Android & Linux

- Although Android is based on linux, it is not just another flavour of Linux, in the way that Ubuntu, Fedora, or Red Hat are.

- Many things you'd expect from a typical Linux distribution aren't available in Android, such as the X11 window manager, the ability to add a person as a Linux user or even the glibc standard C library.

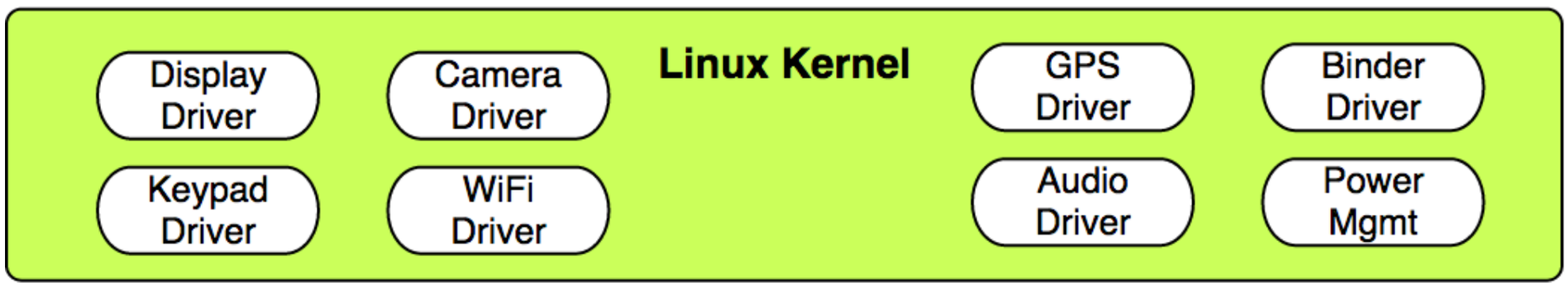
- On the other hand, Android adds quite a bit to the Linux kernel, such as

  - an improved power management that is well-suited for mobile battery-powered devices,

  - a very fast interprocess communication mechanisms

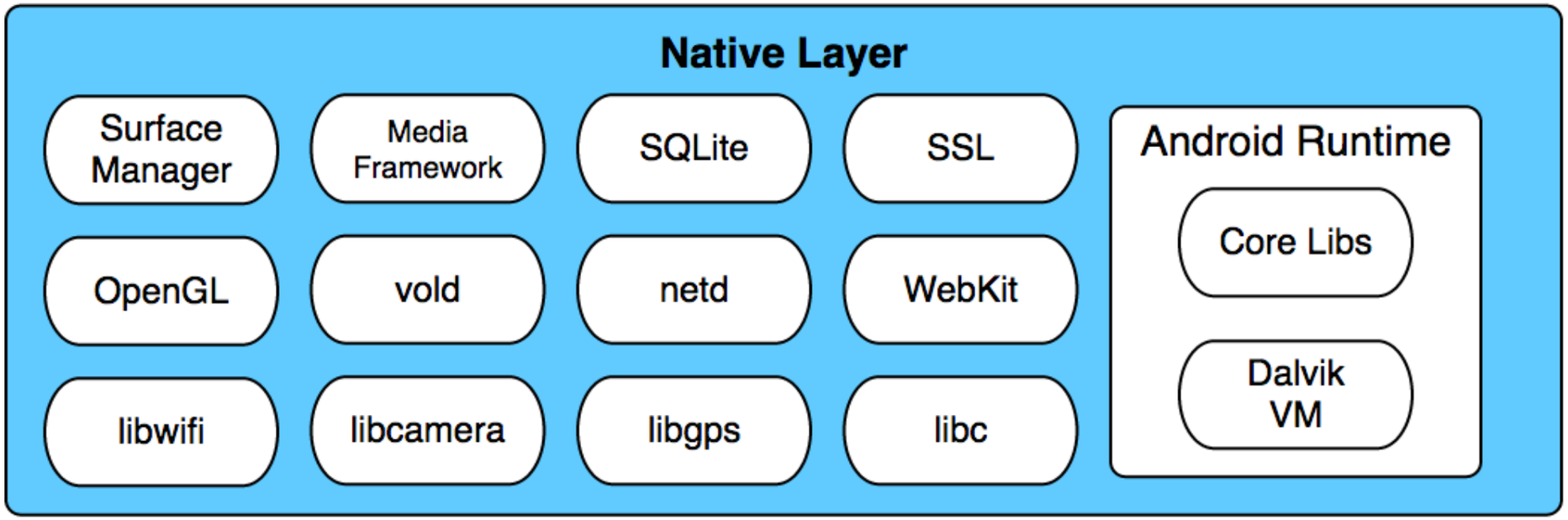
  - mechanisms for sand- boxing applications so they are isolated from one another.

- **Portable**
  - Most low-level parts of Linux have been written in fairly portable C code, which allows for third parties to port Android to a variety of devices.
- **Secure**
  - Linux is a highly secure system, having been tried and tested through some very harsh environments over the decades.
  - Android relies heavily on Linux for security, and all Android applications run as separate Linux processes with permissions set by the Linux system, passing many security concerns to the underlying Linux system.
  - The kernel is the sole enforcer of Android permissions, providing a simple, powerful, security mechanism. It also allows Android apps access to native code, such as fast C implementations of various libraries via the Java Native Interface.
- Features
  - The Linux kernel comes with a range of features. Android leverages many of them, e.g. support for memory and power management, networking and radio functionality.



Native Layer

- The native libraries are C/C++ libraries. Their primary job is to support the Android Application Framework layer



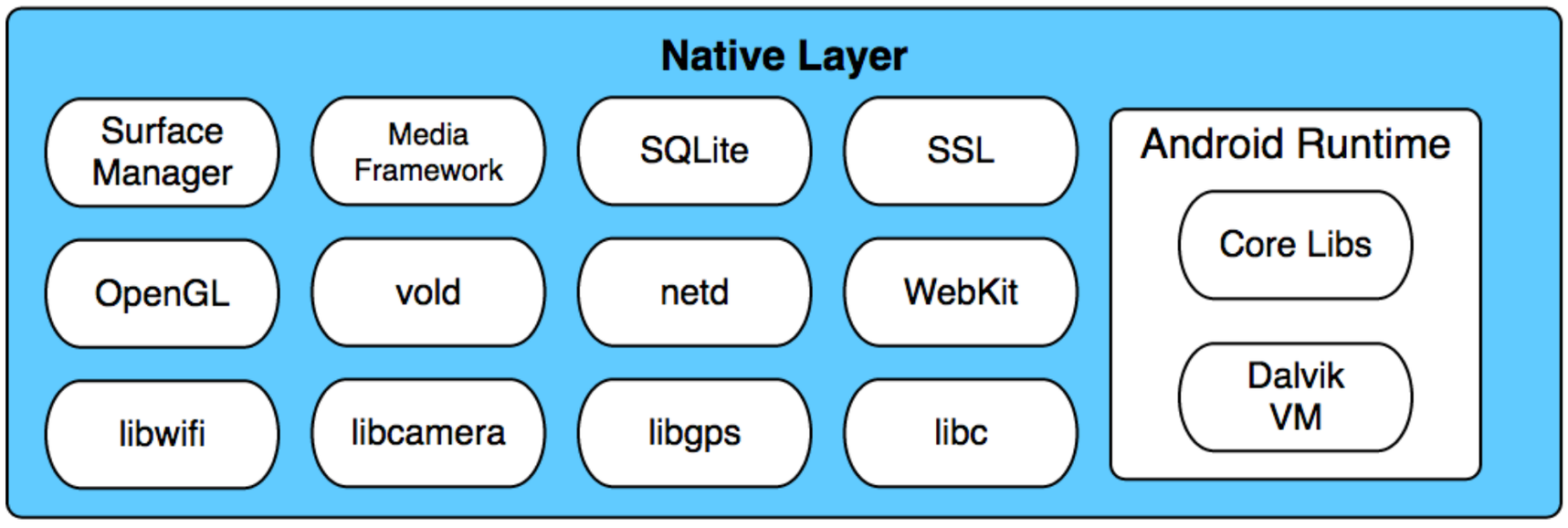
- Some of these libraries are purpose-built for the Android OS, whereas others are often taken from the open source community in order to complete the operating system.

- **Binder:** A very fast inter-process communication mechanism that allows for one Android app to talk to another.

- **Framework libraries**: Various libraries designed to support system services, such as location, media, package installer, telephony, WiFi, voip, and so on.

- **Webkit**: A fast web-rendering engine used by Safari, Chrome, and other browsers.

- **SQLite:** A full-featured SQL database that the Android app framework exposes to applications.



- **Apache Harmony**: An open source implementation of Java libraries.

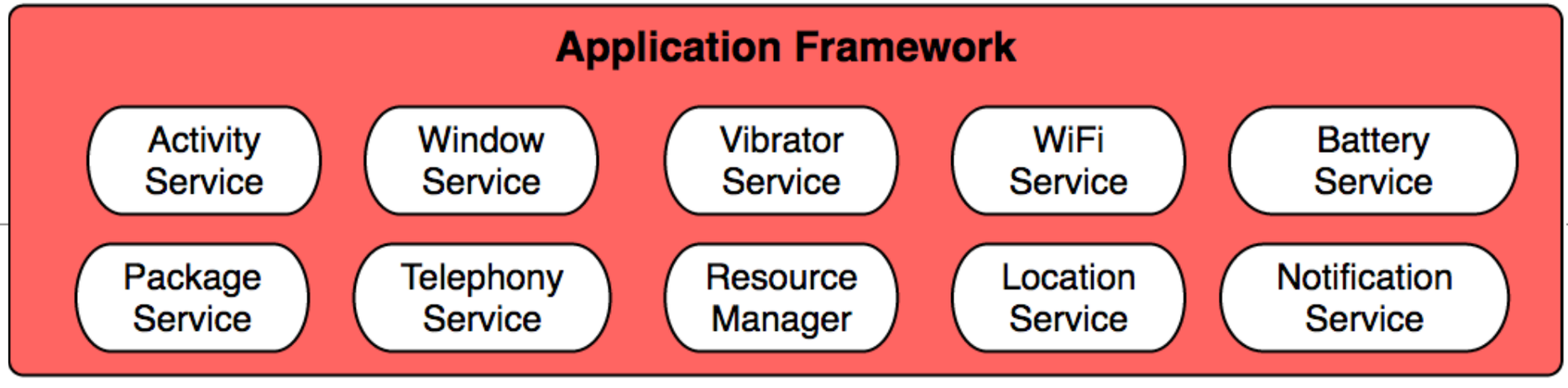
- **OpenGL**: 3D graphics libraries.

- **OpenSSL**: The secure socket layer, allowing for secure point-to-point connectivity.

Native Daemons

- Native daemons are executable code that usually runs to support some kind of system service. Prominent examples:
  - **Service Manager (servicemanager)**: The umbrella process running all other framework services. It is the most critical native daemon.
  - **Radio interface layer daemon (rild):** Responsible for supporting the telephony functionality via GSP or CDMA, usually.
  - **Installation daemon (installd)**: Supports management of apps, including installation, upgrades, as well as granting of permissions.
  - **Media server (mediaserver):** Supports camera, audio, and other media services.
  - **Android Debug Bridge (adbd):** Supports developer connectivity from your PC to the device (including the emuator) so that you can develop apps for Android.

Application Frameworks



- The application framework is a rich environment that provides numerous libraries and services to help the app developer

- This is the best-documented and most extensively covered part of the platform because it is this layer that empowers developers to get applications to the market.

- In the application framework layer, there are numerous Java libraries specifically built for Android. These purpose-built Android classes live in android.* packages.

- There are also most of the standard Java libraries, such as java.lang.*, java.utils.*, java.io.*, java.net.*, etc, which behave as documented in the oracle documentation

- You will also find many services (or managers) that provide the ecosystem of capabilities your application can tap into, such as location, sensors, WiFi, telephony, etc…

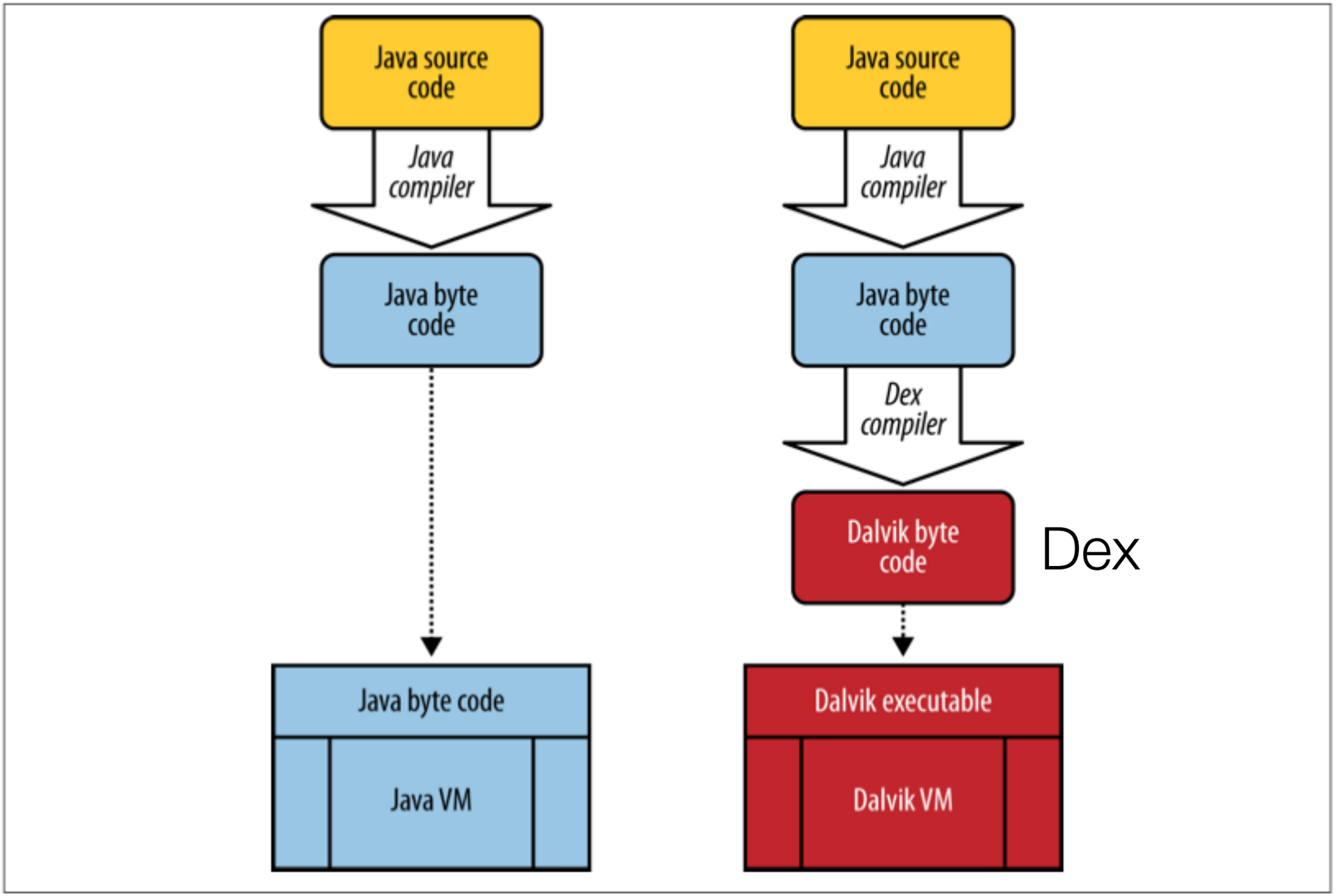
JVM vs Dalvik

- In Java, you write your Java source file, compile it into Java **byte code** using the Java compiler, and then run this byte code on the Java VM.

- In Android you write the Java source file, and you still compile it to Java byte code using the same Java compiler.

- But at that point, you recompile it once again to Dalvik byte code using the Dalvik compiler - producing a **DEX** file

- It is this Dalvik byte code - **DEX Code** - that is then executed on the Dalvik

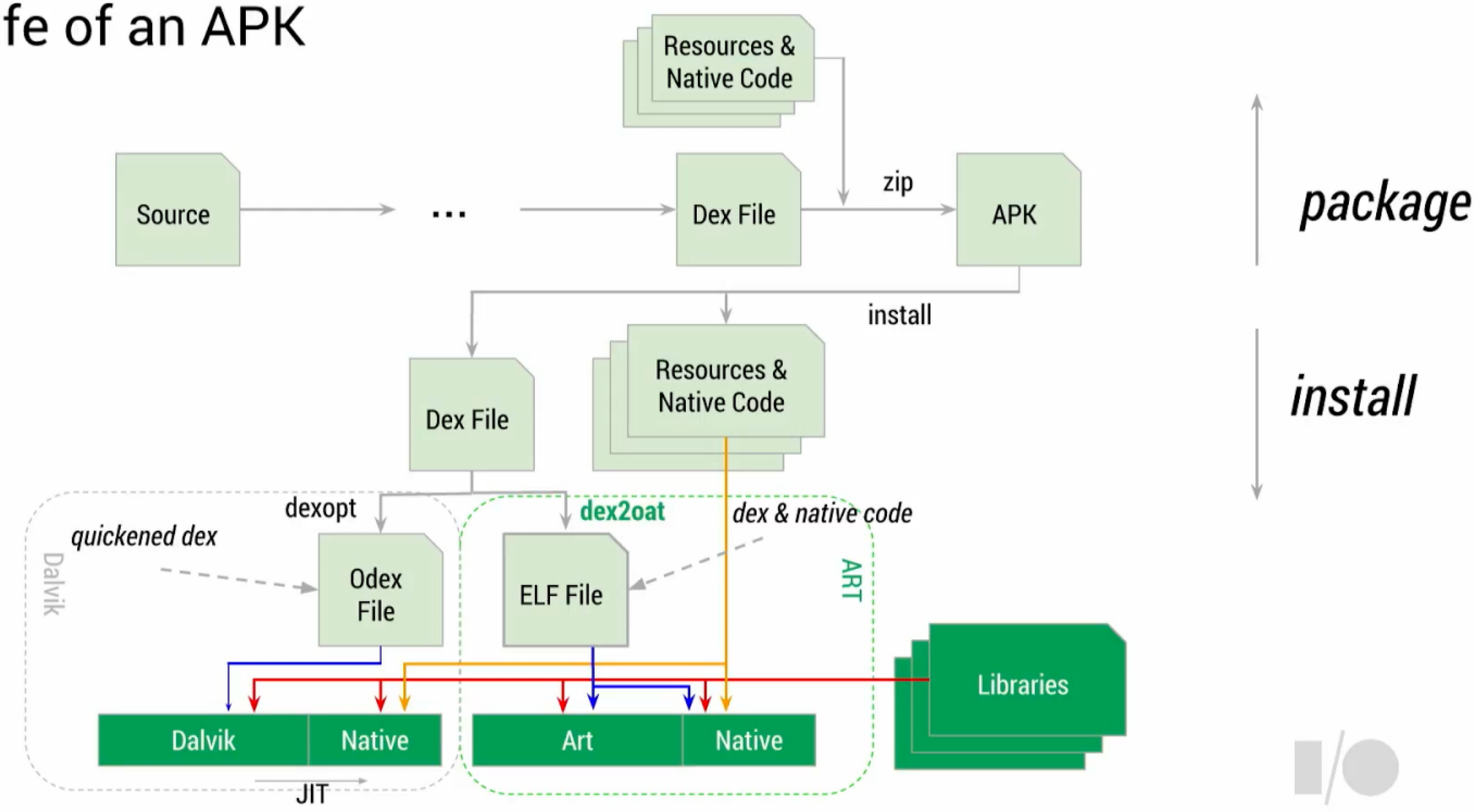

Java source code
Java compiler
Java byte code
Java byte code
Java VM
Java source code
Java compiler
Java byte code
Dex compiler
Dalvik byte code
Dex
Dalvik executable
Dalvik VM

- ART, which stands for Android Runtime, handles app execution in a fundamentally different way from Dalvik.

- The big shift that ART brings, is that instead of being a Just-in-Time (JIT) compiler, it now compiles application code Ahead-of-Time (AOT).

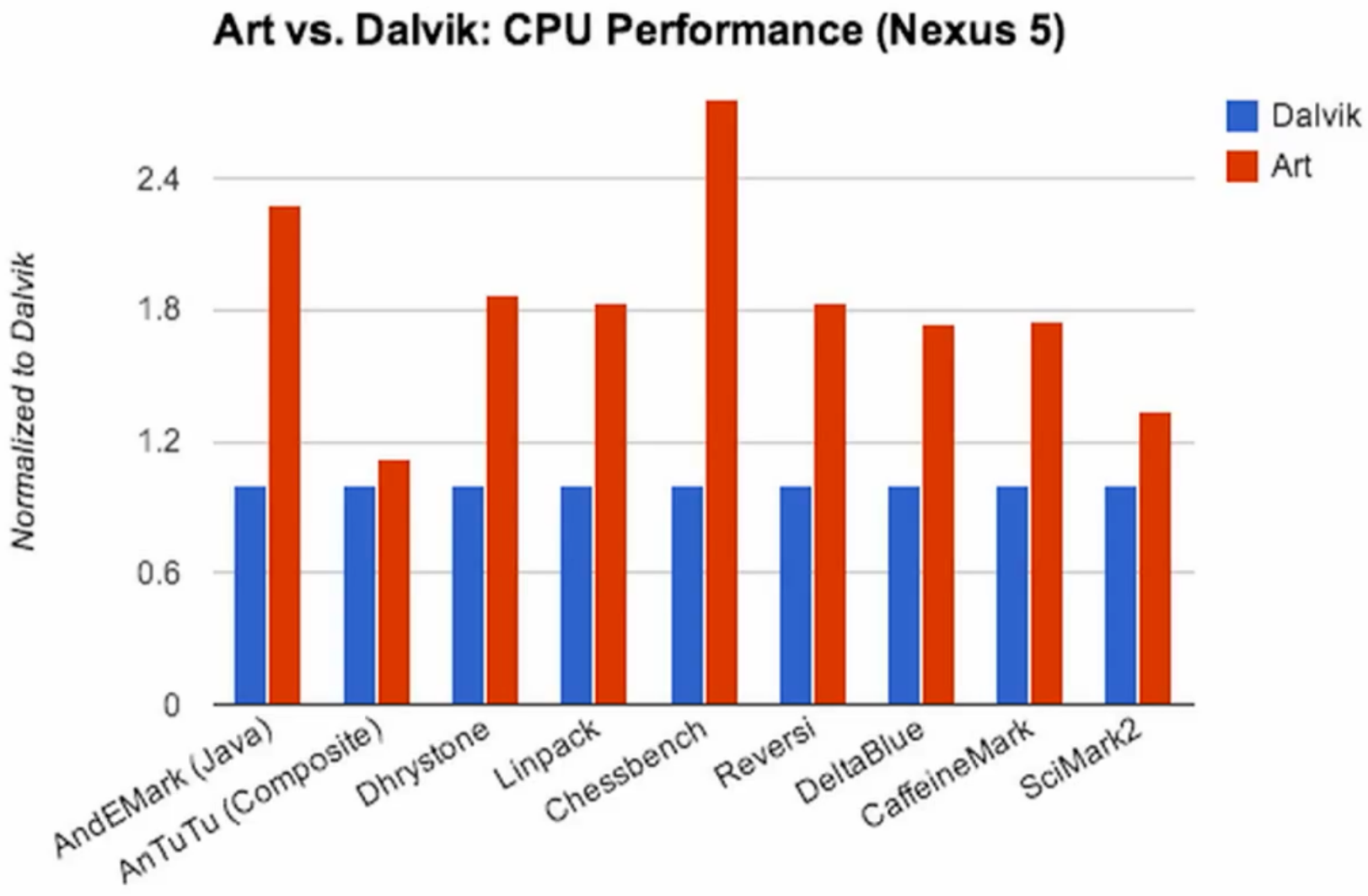
- The runtime goes from having to compile from bytecode to native code each time you run an application, to having it to do it only once, and any subsequent execution from that point forward is done from the existing compiled native code.

The life of an APK



- ART is compatible with Dalvik's existing byte-code format ("dex").
- From a developer's perspective, there are no changes at all in terms of having to write applications for one or the other runtime and no need to worry about compatibilities.

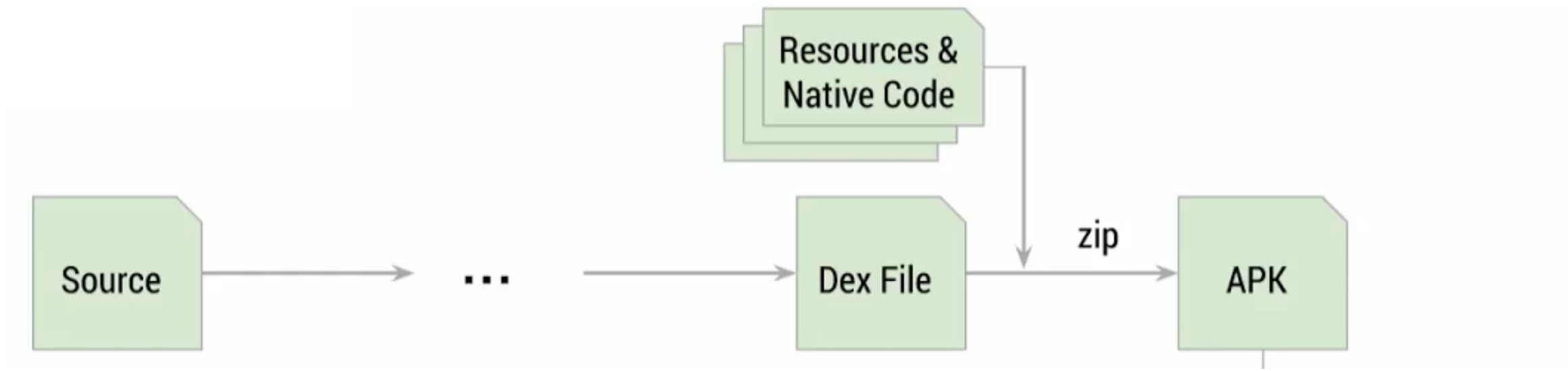
Performance Boosting Thing, realized



Applications

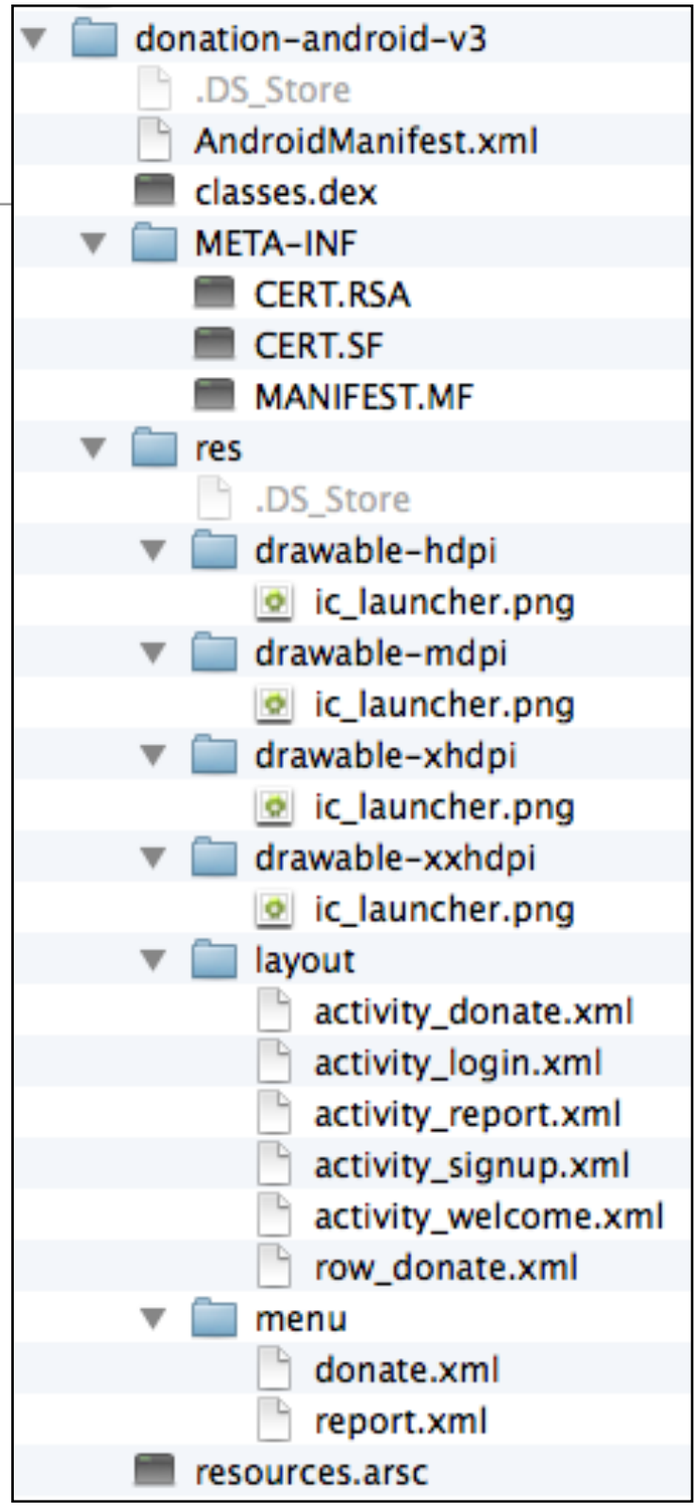
- An application is a single file. We call it an Android application package, or APK for short.

- It is a ZIP file that you can unzip and look inside using any archiving tool



Components of an APK (1)

- **Android Manifest file**
  - This is the main file that provides the big picture about your app—all of its components, permissions, version, and minimum API level needed to run it.
- **Dalvik executable**
  - This is all your Java source code compiled down to a Dalvik executable. The Dalvik executable is the code that runs your application. It is located in a file called classes.dex.
- **Resources**
  - Resources are everything that is not code. Your application may contain a number of images and audio/video clips, as well as numerous XML files describing layouts, language packs, and so on. Collectively, these items are the resources. They are in a file called resources.arsc inside the APK archive as well as in subdirectories such as drawable for images.



Components of an APK (2)

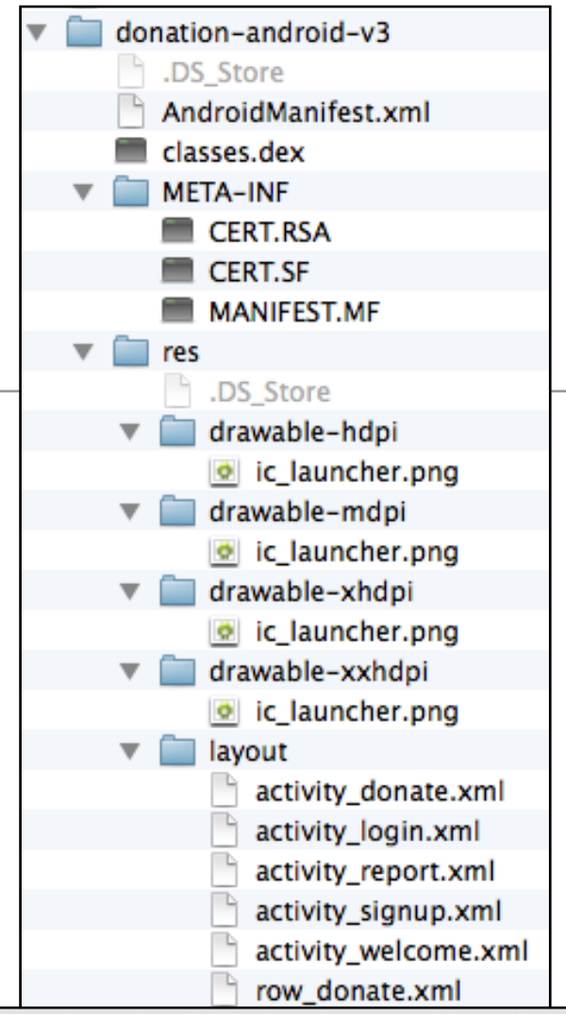
Native libraries

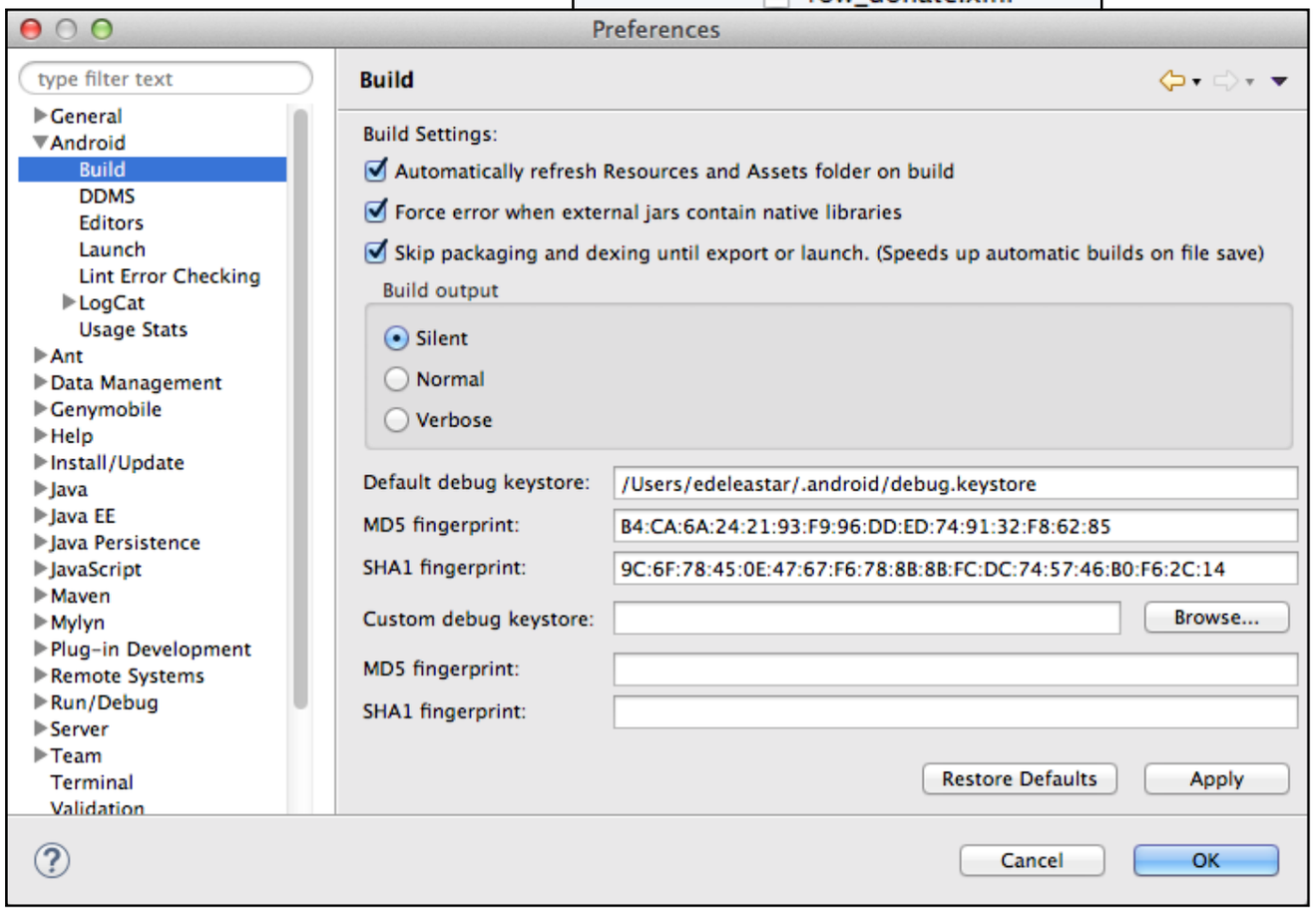
- Optionally, your application may include some native code, such as C/C++ libraries. These libraries could be packaged together with your APK file.

Signatures

- Your APK file also contains a digital signature certifying that you are the author of this application. Signatures are in the META-INF folder.

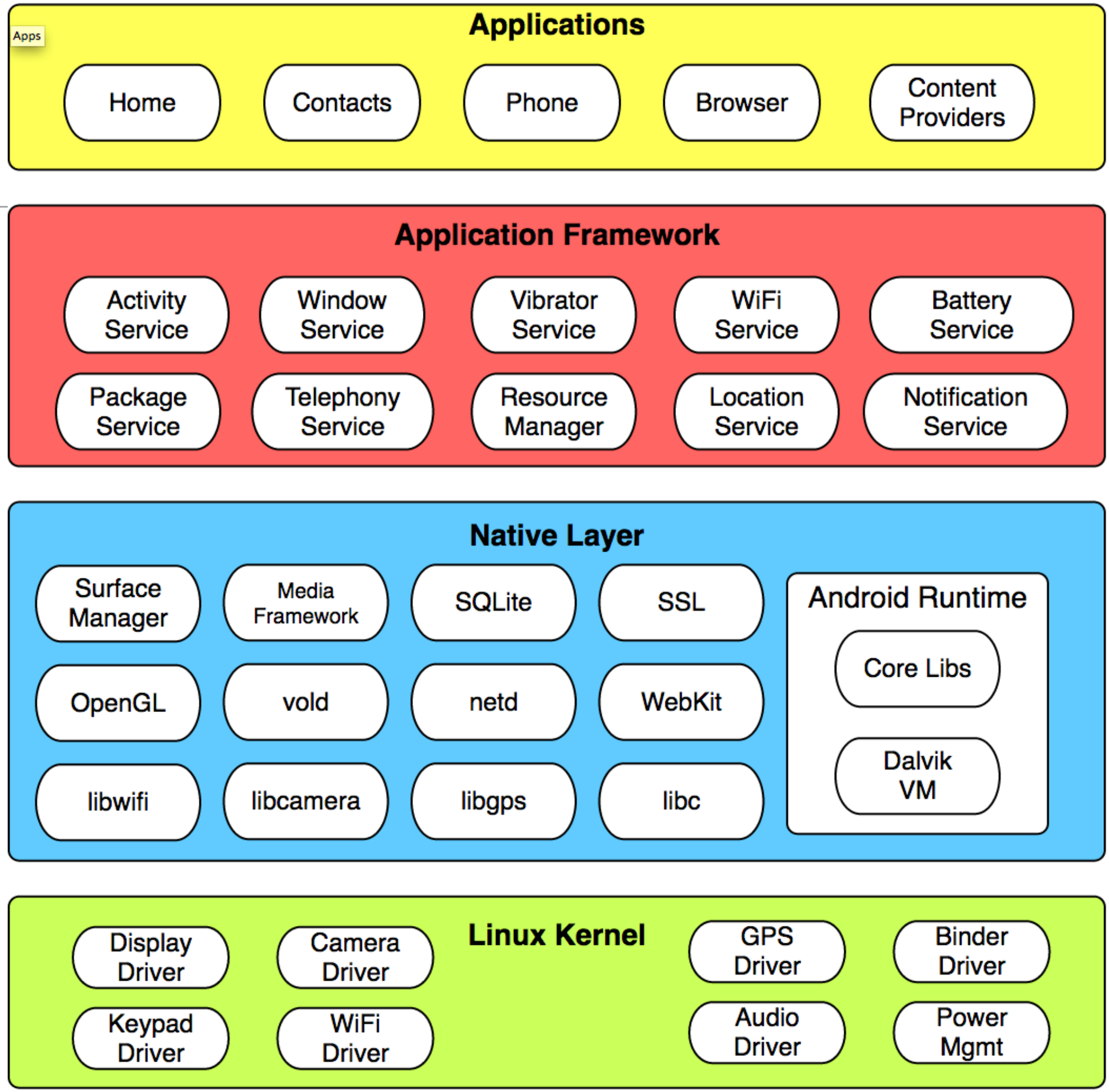
- Android applications must be signed before they can be installed on a device. For development purposes, we sign applications with a debug key—a key that you already have on your development platform. However, when you distribute your application commercially, you must sign it with your own key.



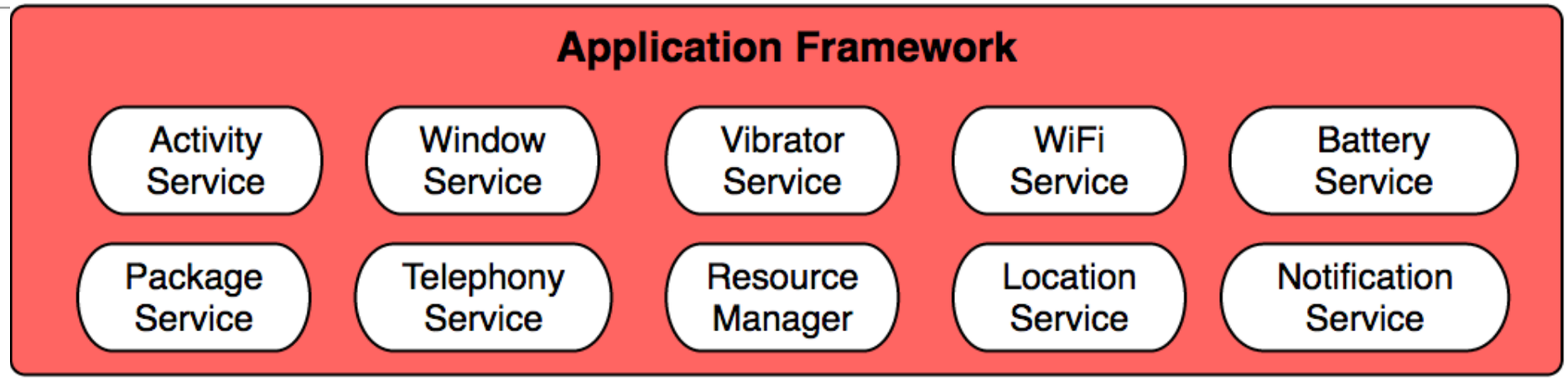


Android Stack

- Which part should we learn?

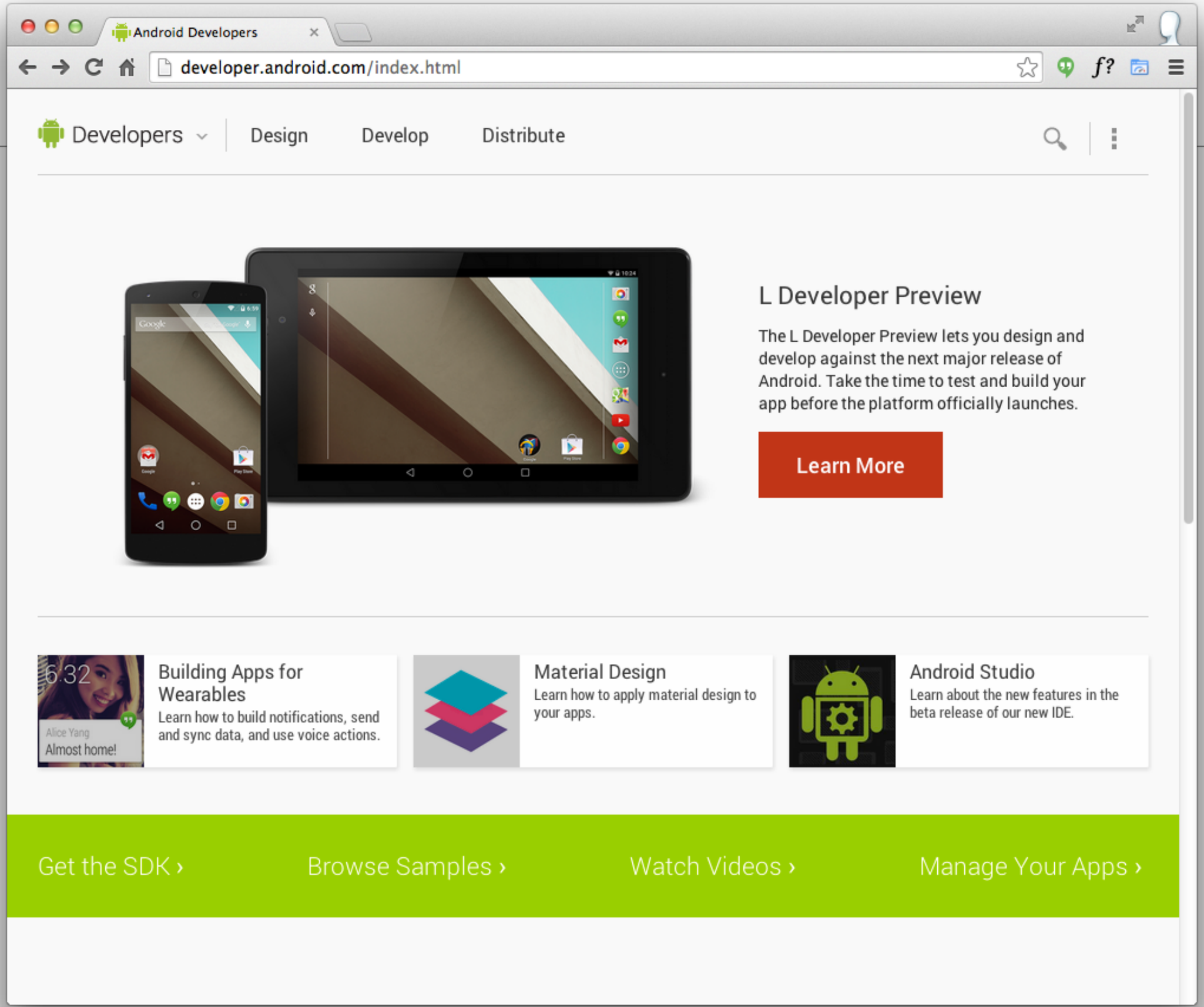


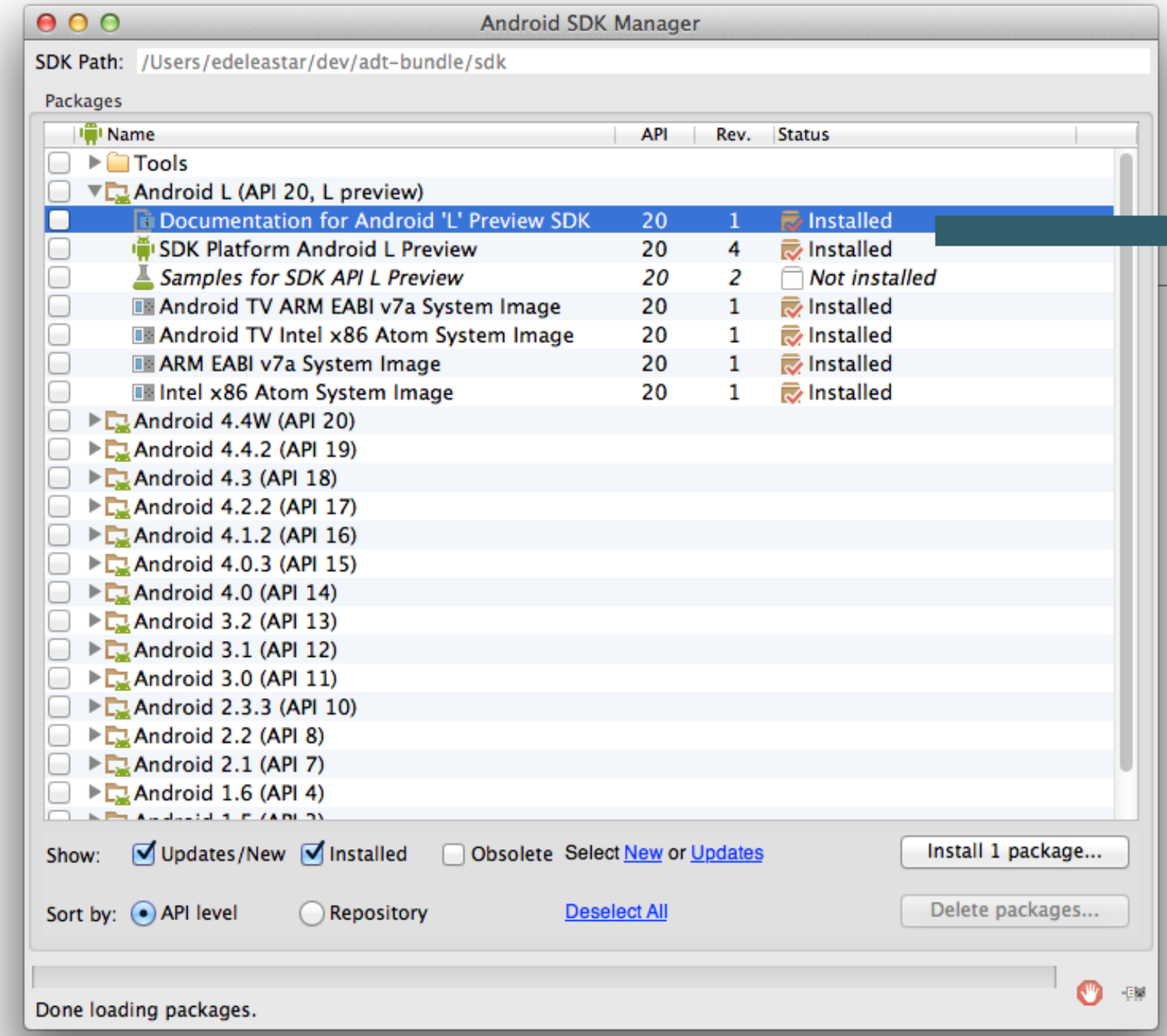
Android Stack

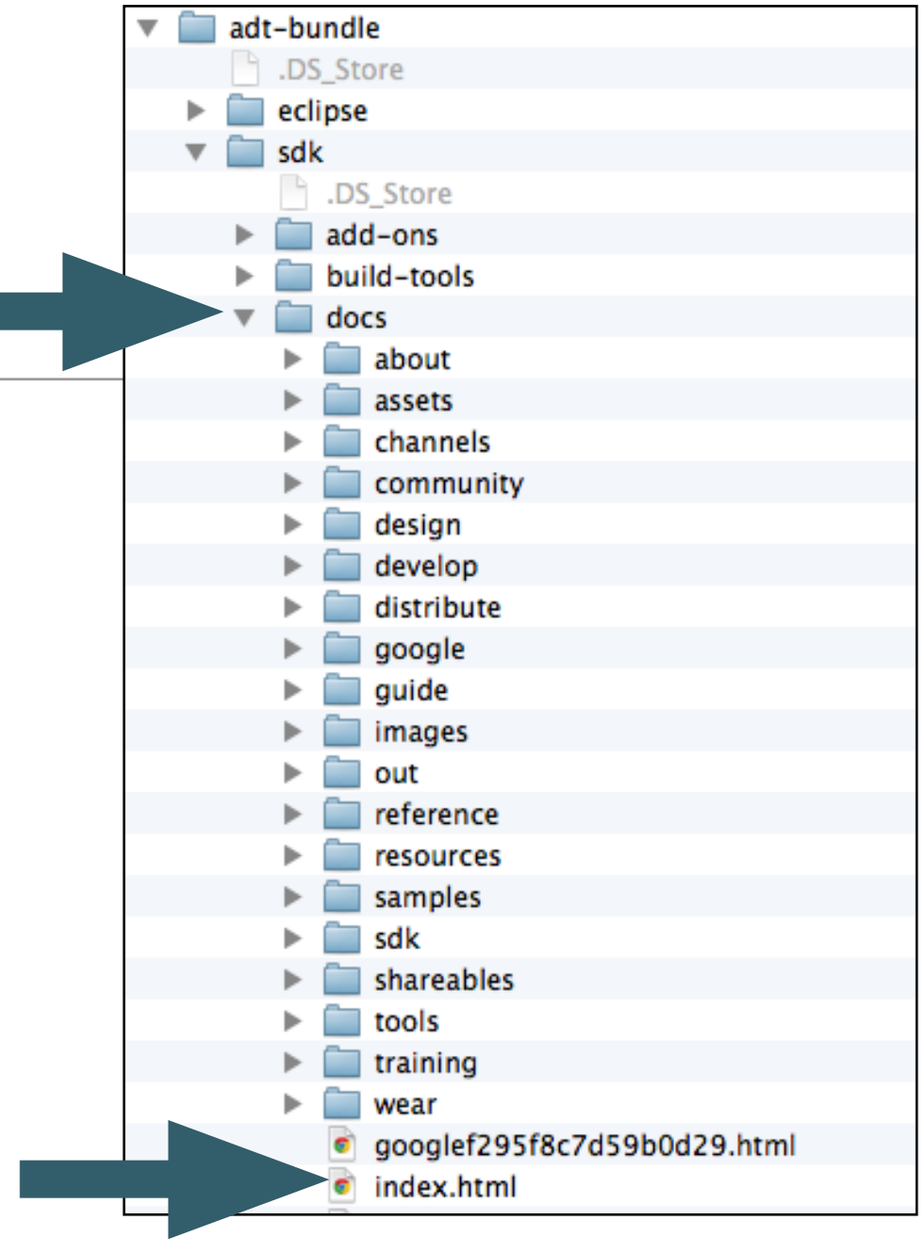


- Almost exclusively - the Application Framework
- Learning Resources?

developer.android.com



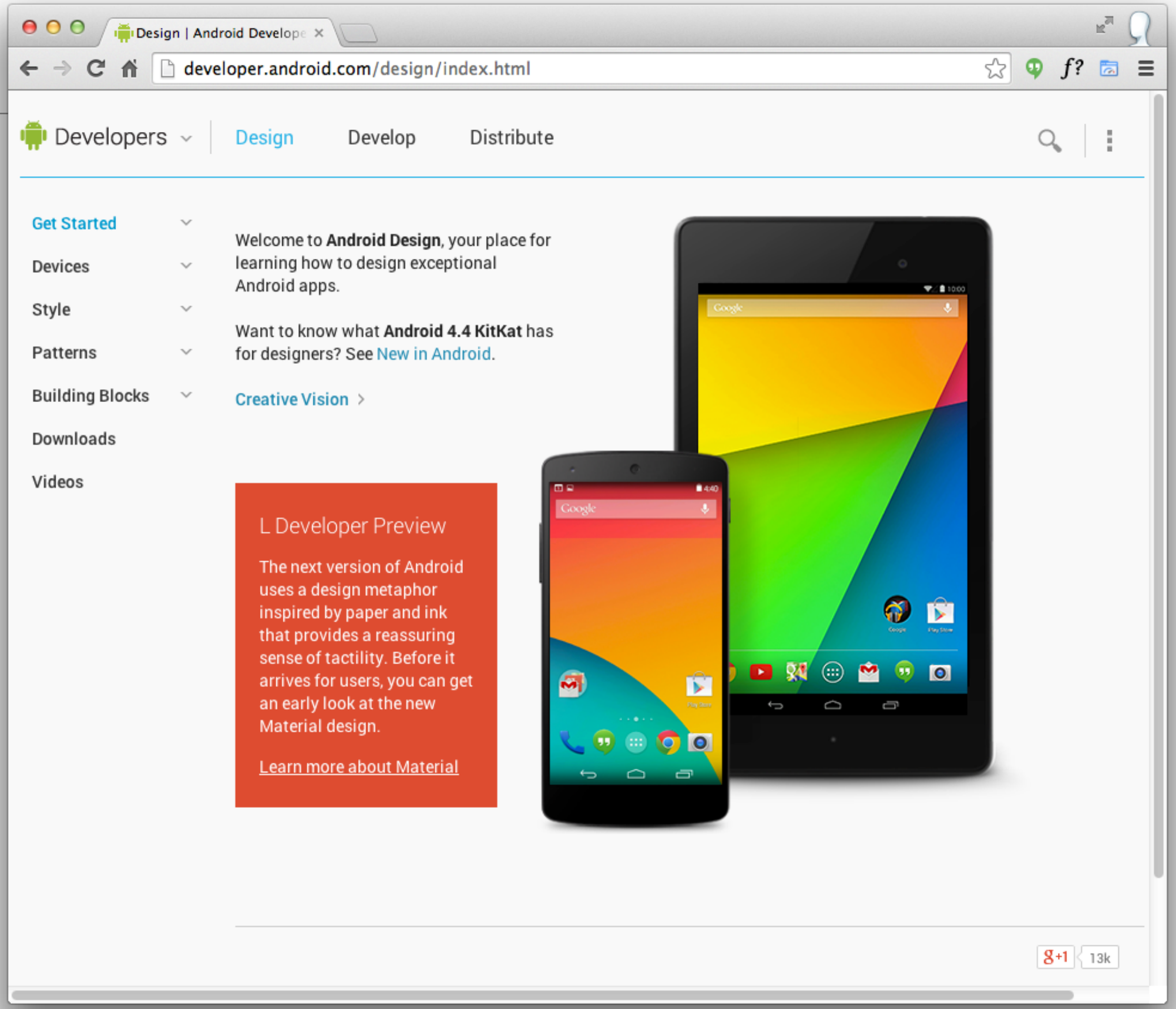




- Latest version of documentation can be downloaded locally via the SDK Manager

- Can then be browsed as a static web site

Design


Design | Android Develope
developer.android.com/design/index.html
Developers
Design
Develop
Distribute
Get Started
Devices
Style
Patterns
Building Blocks
Downloads
Videos
Welcome to Android Design, your place for learning how to design exceptional Android apps.
Want to know what Android 4.4 KitKat has for designers? See New in Android.
Creative Vision
L Developer Preview
The next version of Android uses a design metaphor inspired by paper and ink that provides a reassuring sense of tactility. Before it arrives for users, you can get an early look at the new Material design.
Learn more about Material
+1
13k

Developers

Get Started

Devices

Style

Patterns

Building Blocks

Downloads

Videos

Get Started

- Creative Vision
- Design Principles

Devices

- Phones & Tablets
- Wear
- TV
- Auto

Style

- Devices and Displays
- Themes
- Touch Feedback
- Metrics and Grids
- Typography
- Color
- Iconography
- Your Branding
- Writing Style

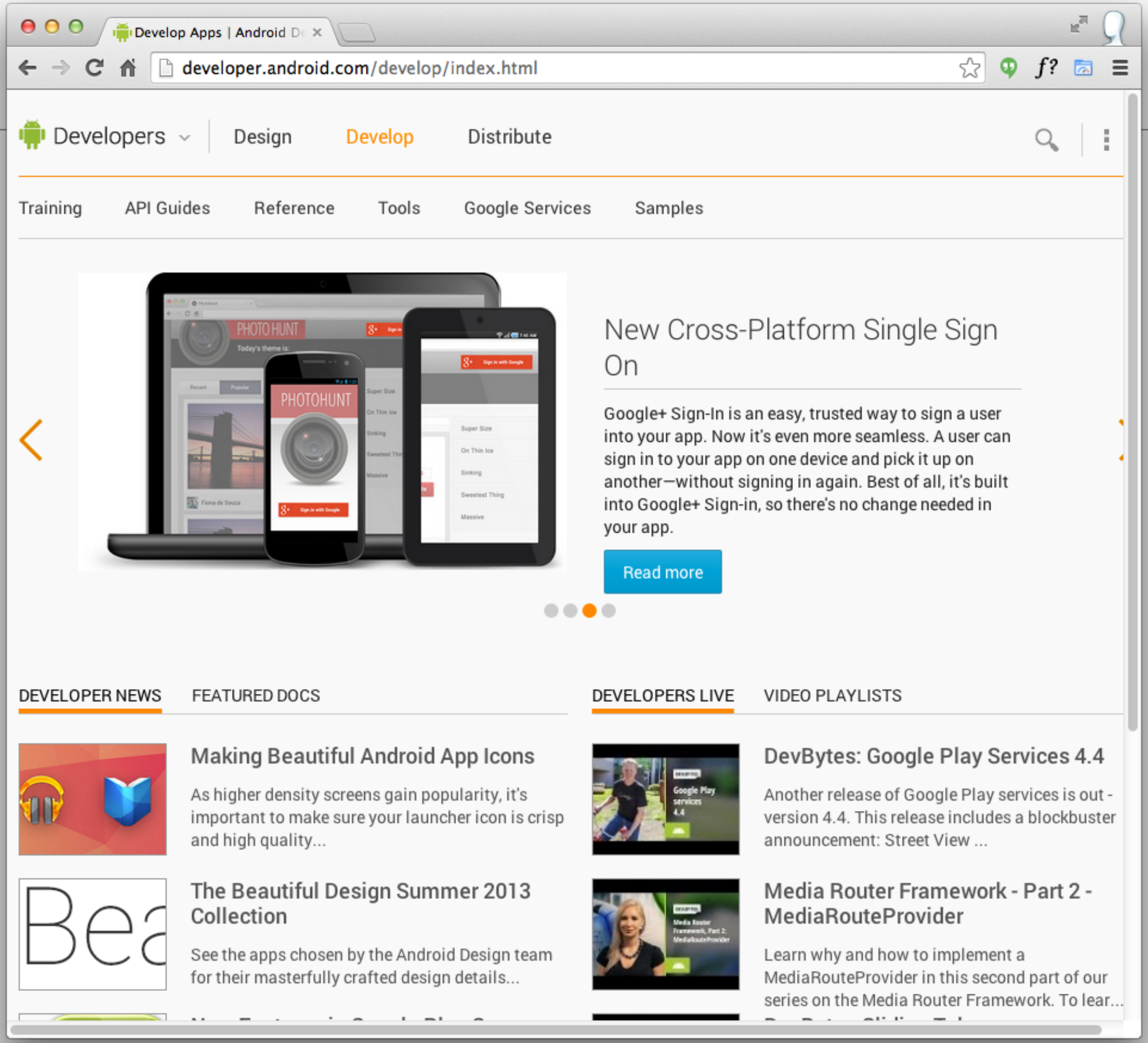
Building Blocks

- Tabs
- Lists
- Grid Lists
- Scrolling
- Spinners
- Buttons
- Text Fields
- Seek Bars
- Progress & Activity
- Switches
- Dialogs
- Pickers

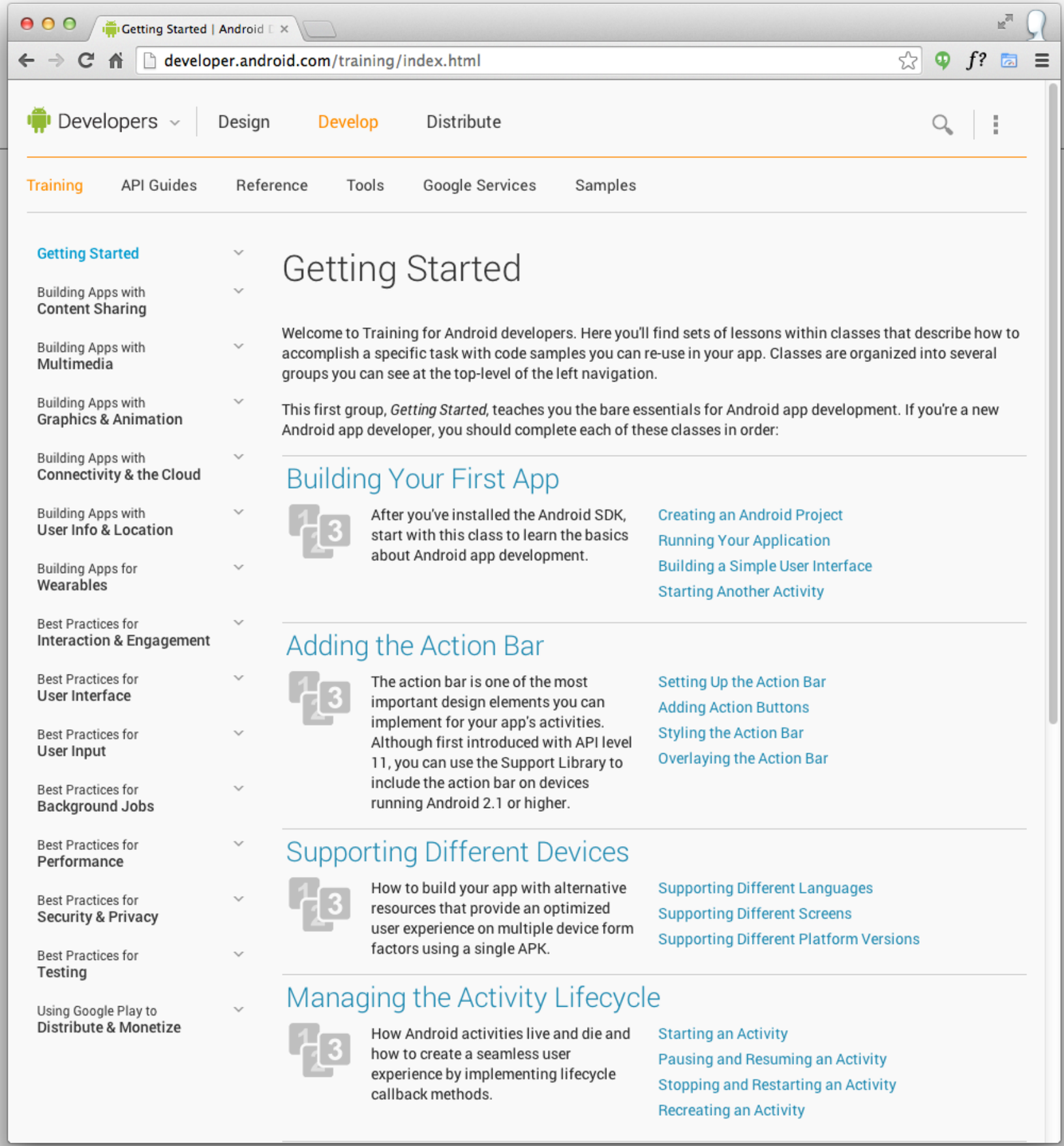
Patterns

- New in Android
- Gestures
- App Structure
- Navigation
- Action Bar
- Navigation Drawer
- Multi-pane Layouts
- Swipe Views
- Full Screen
- Selection
- Confirming & Acknowledging
- Notifications
- Widgets
- Settings
- Help
- Compatibility
- Accessibility
- Pure Android

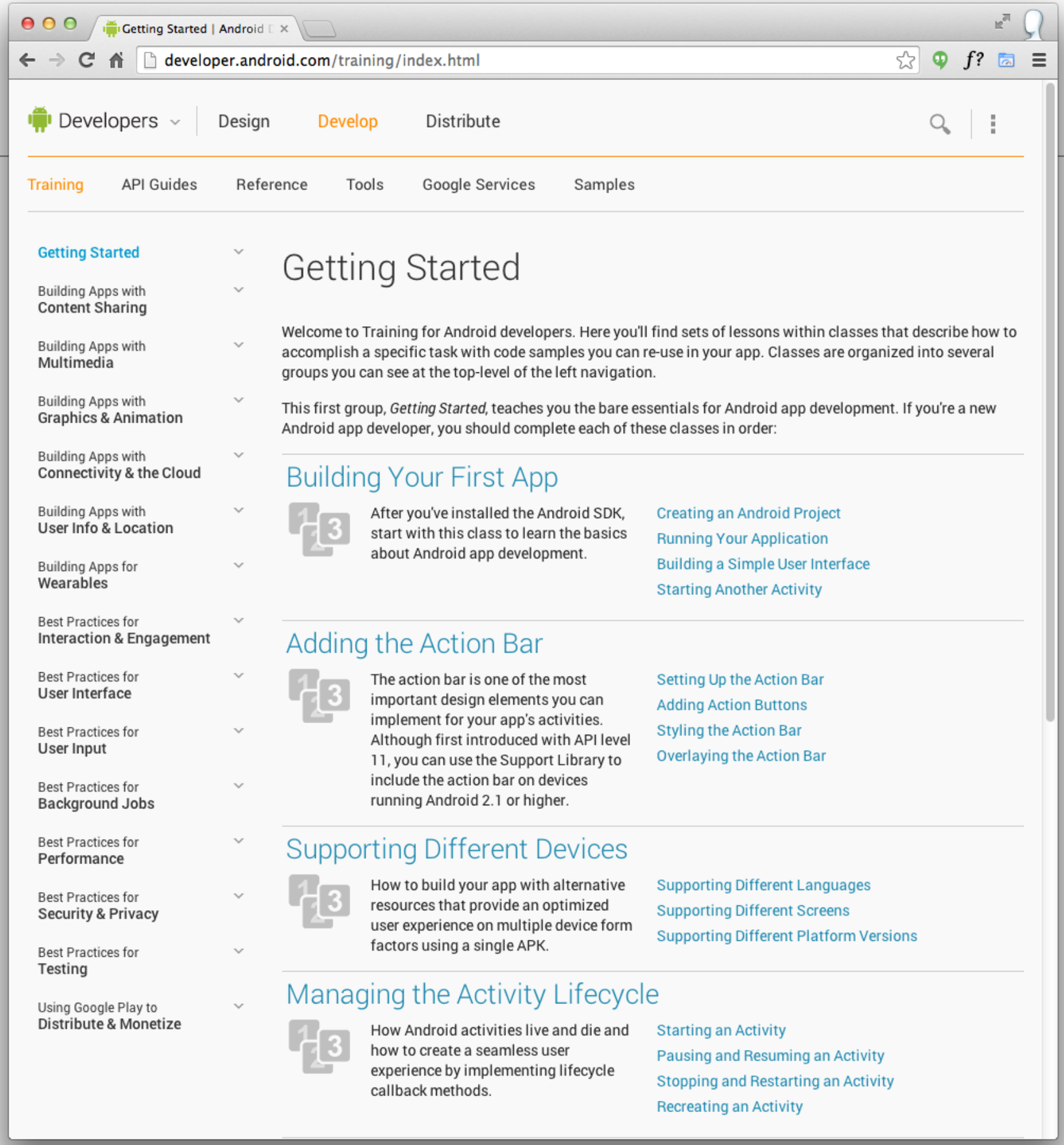
Develop


Develop Apps | Android D
developer.android.com/develop/index.html
Developers
Design
Develop
Distribute
Training
API Guides
Reference
Tools
Google Services
Samples
PHOTO HUNT
PHOTOHUNT
New Cross-Platform Single Sign On
Google+ Sign-In is an easy, trusted way to sign a user into your app. Now it's even more seamless. A user can sign in to your app on one device and pick it up on another—without signing in again. Best of all, it's built into Google+ Sign-in, so there's no change needed in your app.
Read more
DEVELOPER NEWS
FEATURED DOCS
DEVELOPERS LIVE
VIDEO PLAYLISTS
Making Beautiful Android App Icons
As higher density screens gain popularity, it's important to make sure your launcher icon is crisp and high quality...
The Beautiful Design Summer 2013 Collection
See the apps chosen by the Android Design team for their masterfully crafted design details...
DevBytes: Google Play Services 4.4
Another release of Google Play services is out - version 4.4. This release includes a blockbuster announcement: Street View ...
Media Router Framework - Part 2 - MediaRouteProvider
Learn why and how to implement a MediaRouteProvider in this second part of our series on the Media Router Framework. To lear...

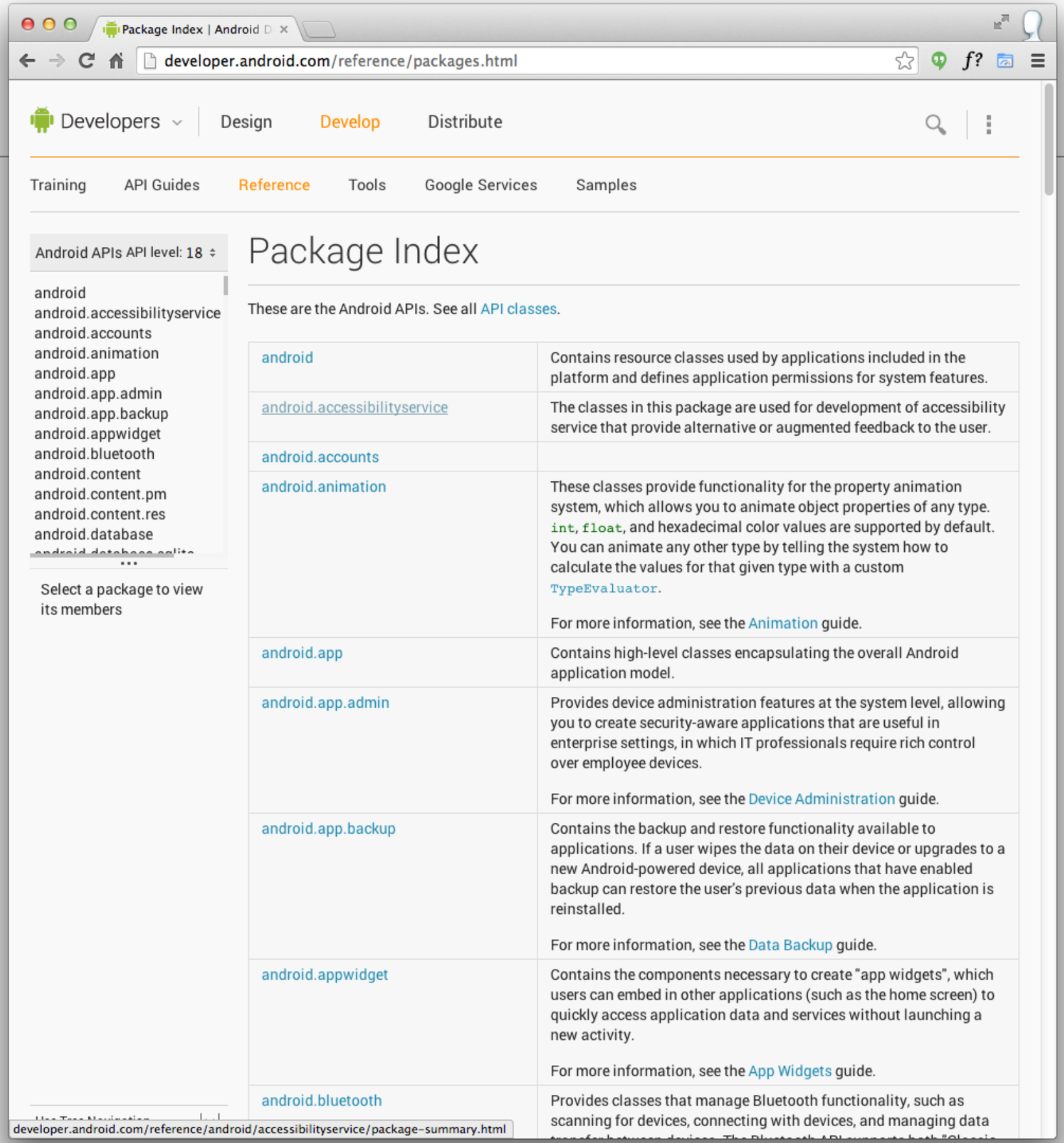
Develop/Training



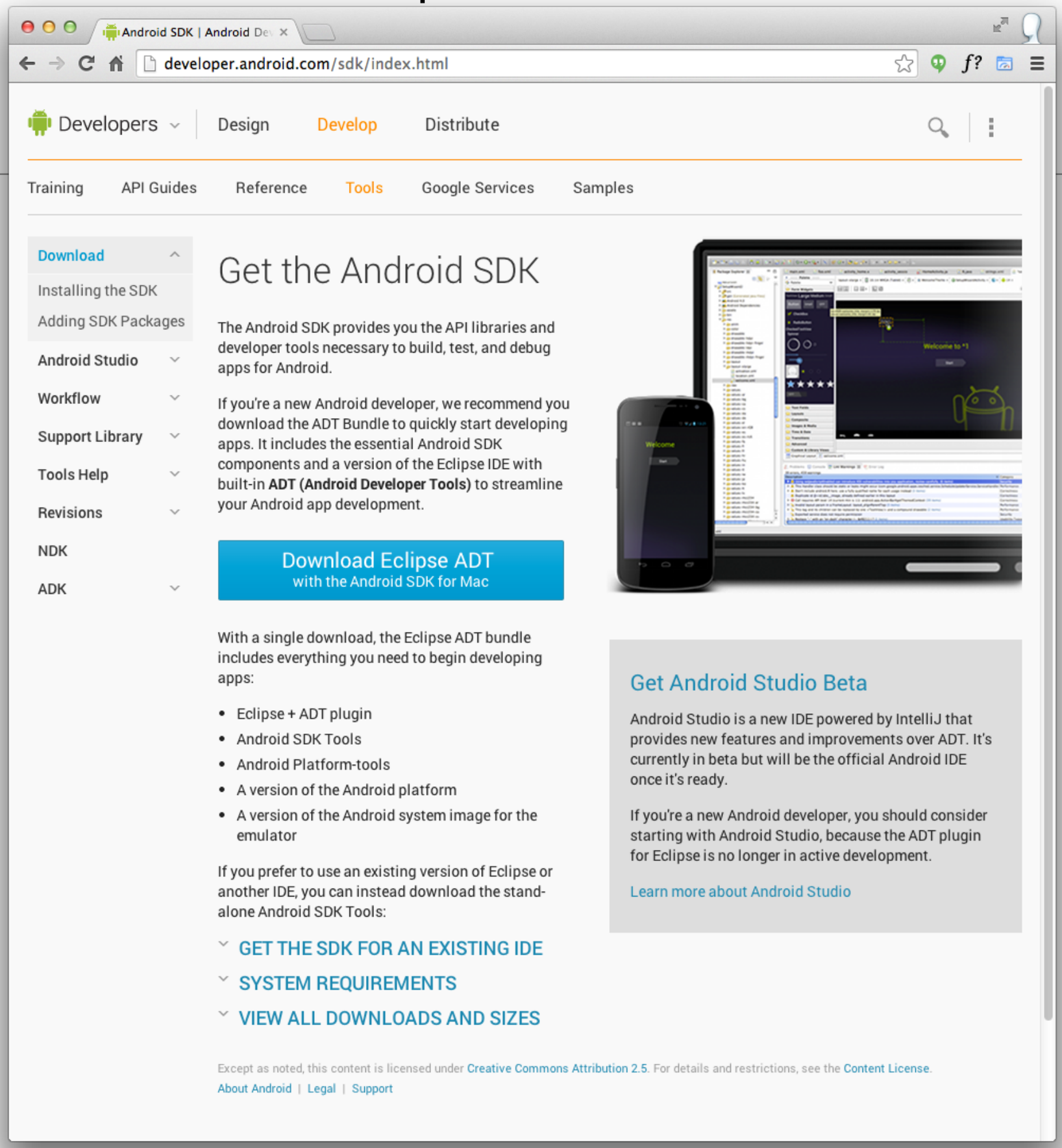
Develop/API Guides



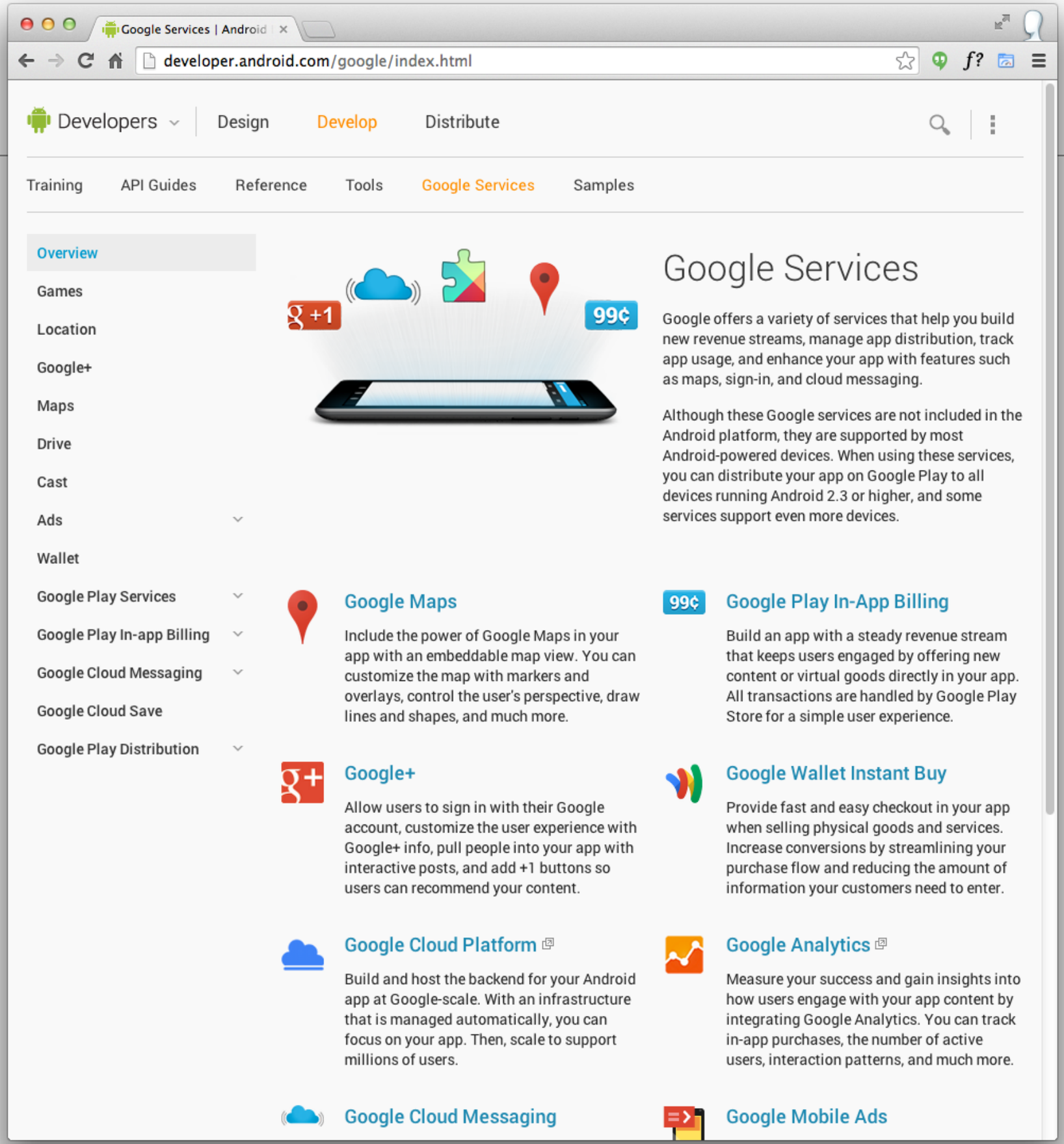
Develop/Reference



Develop/tools



Develop/Google Services



Recommended Texts