Android Developer Fundamentals Course

Learn to develop Android Applications

Concept Reference

Developed by Google Developer Training Team

December 2016

Table of Contents

Kursus Dasar-Dasar Developer Android - Konsep

Belajar mengembangkan aplikasi Android

Referensi Konsep

Dikembangkan oleh Tim Pelatihan Developer Google

Terakhir diperbarui: Desember 2016

1.0: Pengantar Android

Materi:

- Apa yang dimaksud dengan Android?
- Mengapa mengembangkan aplikasi untuk Android?
- Versi Android
- Tantangan development aplikasi Android
- Ketahui selengkapnya

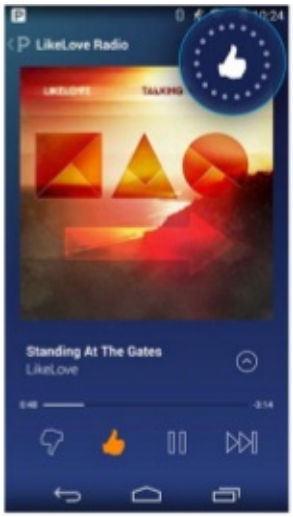
Apa yang dimaksud dengan Android?

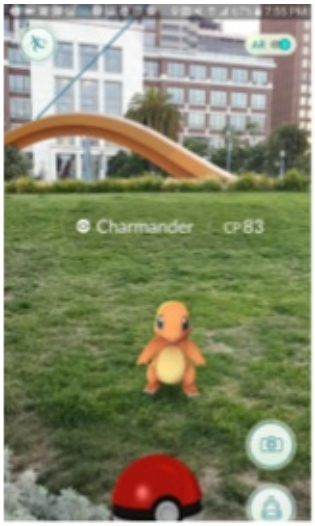
Android adalah sistem operasi dan platform pemrograman yang dikembangkan oleh Google untuk ponsel cerdas dan perangkat seluler lainnya (seperti tablet). Android bisa berjalan di beberapa macam perangkat dari banyak produsen yang berbeda. Android menyertakan kit development perangkat lunak untuk penulisan kode asli dan perakitan modul perangkat lunak untuk membuat aplikasi bagi pengguna Android. Android juga menyediakan pasar untuk mendistribusikan aplikasi. Secara keseluruhan, Android menyatakan ekosistem untuk aplikasi seluler.

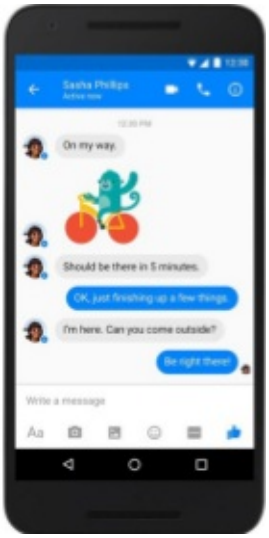
Mengapa mengembangkan aplikasi untuk Android?

Aplikasi dikembangkan untuk berbagai alasan: menjawab kebutuhan bisnis, membangun layanan baru, membuat bisnis baru, dan menyediakan game serta jenis materi lainnya untuk pengguna. Developer memilih untuk mengembangkan bagi Android agar bisa menjangkau sebagian besar pengguna perangkat seluler.

Platform paling populer untuk aplikasi seluler







Pengalaman terbaik untuk pengguna aplikasi

Android menyediakan antarmuka pengguna (UI) layar sentuh untuk berinteraksi dengan aplikasi. Antarmuka pengguna Android sebagian besar berdasarkan pada manipulasi langsung, menggunakan isyarat sentuhan seperti menggesek, mengetuk, dan mencubit untuk memanipulasi objek di layar. Selain keyboard, ada keyboard virtual yang bisa disesuaikan untuk masukan teks. Android juga bisa mendukung pengontrol game dan keyboard fisik berukuran penuh yang dihubungkan dengan Bluetooth atau USB.

Layar utama Android bisa berisi sejumlah laman *ikon aplikasi*, yang akan meluncurkan aplikasi terkait, dan *widget*, dengan menampilkan materi langsung yang diperbarui secara otomatis seperti cuaca, kotak masuk email pengguna, atau ticker berita. Android juga bisa memutar materi multimedia seperti musik, animasi, dan video. Gambar di atas menampilkan ikon aplikasi pada layar utama (kiri), musik yang diputar (tengah), dan widget yang ditampilkan (kanan). Sepanjang bagian atas layar terdapat bilah status, yang menampilkan informasi tentang perangkat dan konektivitasnya. Layar utama Android bisa terdiri dari sejumlah laman, yang bisa digesek mundur dan maju oleh pengguna.

Android didesain untuk menyediakan respons cepat terhadap masukan pengguna. Selain antarmuka sentuh yang berubah-ubah, kemampuan getaran perangkat Android bisa menyediakan umpan balik sentuhan. Perangkat keras internal seperti akselerometer, giroskop, dan sensor kedekatan, digunakan oleh banyak aplikasi untuk merespons tindakan pengguna

Android didesain untuk menyediakan respons cepat terhadap masukan pengguna. Selain antarmuka sentuh yang berubah-ubah, kemampuan getaran perangkat Android bisa menyediakan umpan balik sentuhan. Perangkat keras internal seperti akselerometer, giroskop, dan sensor kedekatan, digunakan oleh banyak aplikasi untuk merespons tindakan pengguna tambahan. Sensor tersebut bisa mendeteksi rotasi layar dari potret ke lanskap untuk tampilan yang lebih lebar atau sensor bisa memungkinkan pengguna untuk menyetir kendaraan virtual dengan memutar perangkat seolah-olah setir mobil.

Platform Android, berdasarkan pada kernel Linux, terutama didesain untuk perangkat seluler layar sentuh seperti ponsel cerdas dan tablet. Karena perangkat Android biasanya bertenaga baterai, Android didesain untuk mengelola proses guna menjaga konsumsi daya tetap minimum, sehingga menyediakan penggunaan baterai lebih lama.

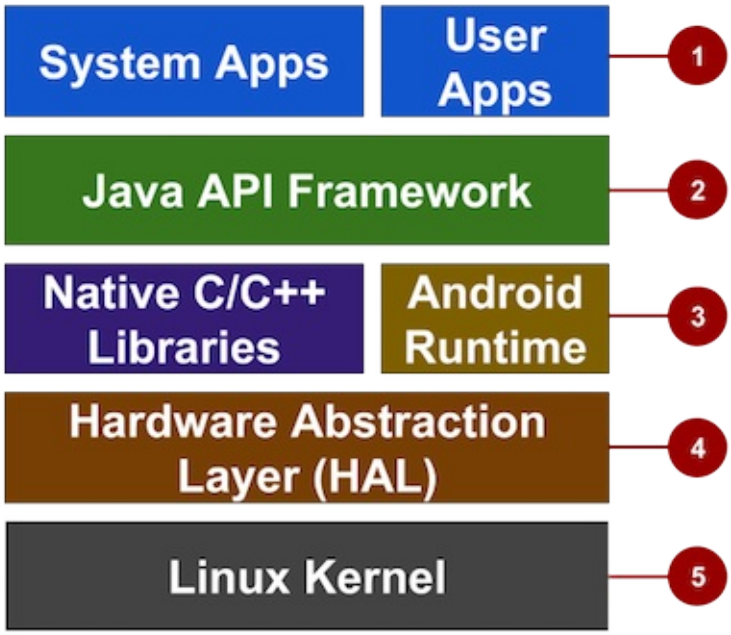
Mudah mengembangkan aplikasi

Gunakan Android Software Development Kit (SDK) Android untuk mengembangkan aplikasi yang memanfaatkan UI dan sistem operasi Android. SDK terdiri dari serangkaian alat development menyeluruh yang menyertakan debugger, pustaka perangkat lunak kode pratulis, emulator perangkat, dokumentasi, kode contoh, dan tutorial. Gunakan alat-alat ini untuk membuat aplikasi yang terlihat hebat dan manfaatkan kemampuan perangkat keras yang tersedia di setiap perangkat.

Untuk mengembangkan aplikasi menggunakan SDK, gunakan bahasa pemrograman Java untuk mengembangkan aplikasi dan file Extensible Markup Language (XML) untuk menjelaskan sumber daya data. Dengan menulis kode di Java dan membuat biner aplikasi tunggal, Anda akan memiliki aplikasi yang bisa berjalan pada faktor bentuk ponsel dan tablet. Anda bisa mendeklarasikan UI dalam rangkaian sumber daya XML ringan, satu rangkaian untuk bagian UI yang umum bagi semua faktor bentuk, dan rangkaian lain untuk fitur yang khusus bagi ponsel atau tablet. Pada waktu proses, Android menerapkan rangkaian sumber daya yang tepat berdasarkan ukuran layar, kepadatan, lokal, dan sebagainya.

Untuk membantu Anda mengembangkan aplikasi secara efisien, Google menawarkan Lingkungan Development Terintegrasi (IDE) Java lengkap yang disebut Android Studio, dengan fitur lanjutan untuk pengembangan, debug, dan pemaketan aplikasi Android. Dengan menggunakan Android Studio, Anda bisa mengembangkan perangkat Android yang tersedia, atau membuat perangkat virtual yang mengemulasikan konfigurasi perangkat keras apa pun.

Android menyediakan arsitektur development yang kaya. Anda tidak perlu mengetahui banyak tentang komponen arsitektur ini, namun perlu mengetahui apa yang tersedia dalam sistem yang digunakan untuk aplikasi Anda. Diagram berikut menampilkan komponen utama sistem *tumpukan* Android — sistem operasi dan arsitektur development.



Dalam gambar di atas:

1. **Aplikasi:** Aplikasi berada pada tingkat ini, bersama dengan aplikasi sistem inti untuk email, perpesanan SMS, kalender, penjelajahan Internet, atau kontak.
2. **Kerangka Kerja API Java:** Semua fitur Android tersedia untuk developer melalui antarmuka pemograman aplikasi

- Sistem Tampilan digunakan untuk membangun UI aplikasi, termasuk daftar, tombol, dan menu.
- Pengelola Referensi digunakan untuk mengakses sumber daya non-kode seperti string, grafik, dan file layout yang dilokalkan.
- Pengelola Notifikasi digunakan untuk menampilkan peringatan khusus di bilah status.
- Pengelola Aktivitas yang mengelola daur hidup aplikasi.
- Penyedia Materi yang memungkinkan aplikasi untuk mengakses data dari aplikasi lain.
- Semua API kerangka kerja yang digunakan aplikasi sistem Android.

3. **Pustaka dan Waktu Proses Android:**Setiap aplikasi berjalan dalam prosesnya sendiri dan dengan instance Android Runtime sendiri, yang memungkinkan beberapa mesin sekaligus virtual pada perangkat bermemori rendah. Android juga menyertakan rangkaian pustaka waktu proses inti yang menyediakan sebagian besar fungsionalitas bahasa pemrograman Java, termasuk beberapa fitur bahasa Java 8 yang digunakan kerangka kerja Java API. Banyak layanan dan komponen sistem Android inti dibangun dari kode asli yang memerlukan pustaka asli yang ditulis dalam C dan C++. Pustaka asli tersebut tersedia untuk aplikasi melalui kerangka kerja Java API.
4. **Hardware Abstraction Layer (HAL):** Lapisan ini menyediakan antarmuka standar yang menunjukkan kemampuan perangkat keras di perangkat ke kerangka kerja Java API yang lebih tinggi. HAL terdiri atas beberapa modul pustaka, masing-masing mengimplementasikan antarmuka untuk komponen perangkat keras tertentu, seperti modul kamera atau bluetooth.
5. **Kernel Linux:** Fondasi platform Android adalah kernel Linux. Lapisan di atas mengandalkan kernel Linux untuk fungsionalitas pokok seperti threading dan manajemen memori tingkat rendah. Menggunakan kernel Linux memungkinkan Android memanfaatkan fitur keamanan utama dan memungkinkan produsen perangkat mengembangkan driver perangkat keras untuk kernel yang cukup dikenal.

Banyak opsi distribusi

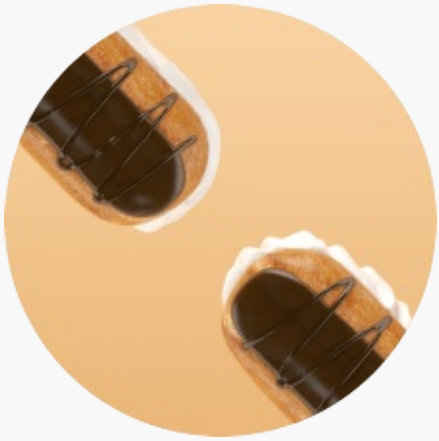
Anda bisa mendistribusikan aplikasi Android dalam banyak cara: email, situs web, atau pasar aplikasi seperti Google Play. Pengguna Android mengunduh jutaan aplikasi dan game dari Google Play store setiap bulan (ditampilkan dalam gambar di bawah ini). Google Play adalah layanan distribusi digital, yang dioperasikan dan dikembangkan oleh Google, yang berfungsi sebagai toko aplikasi resmi untuk Android, yang memungkinkan konsumen menjelajah dan mengunduh aplikasi

yang dikembangkan dengan Android SDK dan dipublikasikan melalui Google.

Versi Android

Versi Android

Google menyediakan peningkatan versi bertahap utama untuk sistem operasi Android setiap enam hingga sembilan bulan, menggunakan nama bertema makanan. Rilis utama yang terbaru adalah Android 7.0 "Nougat".

Nama kode	Nomor versi	Tanggal rilis awal	API level
N/A	1.0	23 September 2008	1
N/A	1.1	9 Februari 2009	2
Cupcake	1.5	27 April 2009	3
Donut 	1.6	15 September 2009	4
Eclair 	2.0 – 2.1	26 Oktober 2009	5–7
Froyo			

Gingerbread 	2.3 – 2.3.7	6 Desember 2010	9–10
Honeycomb 	3.0 – 3.2.6	22 Februari 2011	11–13
Ice Cream Sandwich 	4.0 – 4.0.4	18 Oktober 2011	14–15
Jelly Bean	4.1 – 4.3.1	9 Juli 2012	16–18

			
Jelly Bean 	4.1 – 4.3.1	9 Juli 2012	16–18
KitKat 	4.4 – 4.4.4	31 Oktober 2013	19–20
Lollipop			

	6.0 – 6.0.1	5 Oktober 2015	23
Nougat 	7.0	22 Agustus 2016	24

Lihat versi sebelumnya dan fiturnya dalam Cerita Android.

Dasbor untuk Versi Platform diperbarui secara berkala untuk menampilkan distribusi perangkat aktif yang menjalankan setiap versi Android, berdasarkan jumlah perangkat yang mengunjungi Google Play Store. Mendukung sekitar 90% perangkat aktif adalah praktik yang baik, sekaligus menargetkan aplikasi Anda ke versi terbaru.

Catatan: Untuk menyediakan fitur dan fungsionalitas terbaik di seluruh versi Android, gunakan Pustaka Dukungan Android di aplikasi Anda. Pustaka dukungan ini memungkinkan aplikasi Anda menggunakan API platform terbaru di perangkat lama.

Tantangan development aplikasi Android

Sewaktu platform Android menyediakan fungsionalitas kaya untuk development aplikasi, masih ada sejumlah tantangan yang perlu Anda tangani, seperti:

- Membangun untuk dunia multilayar
- Mendapatkan kinerja yang tepat
- Membuat kode dan pengguna Anda tetap aman
- Tetap kompatibel dengan versi platform yang lebih lama
- Memahami pasar dan pengguna.

Membangun untuk dunia multilayar

Android berjalan pada miliaran perangkat genggam di seluruh dunia, dan mendukung beragam faktor bentuk termasuk perangkat yang dapat dikenakan dan televisi. Perangkat bisa tersedia dalam ukuran dan bentuk berbeda yang memengaruhi desain layar untuk elemen UI di aplikasi Anda.

- Mendapatkan kinerja yang tepat
- Membuat kode dan pengguna Anda tetap aman
- Tetap kompatibel dengan versi platform yang lebih lama
- Memahami pasar dan pengguna.

Membangun untuk dunia multilayar

Android berjalan pada miliaran perangkat genggam di seluruh dunia, dan mendukung beragam faktor bentuk termasuk perangkat yang dapat dikenakan dan televisi. Perangkat bisa tersedia dalam ukuran dan bentuk berbeda yang memengaruhi desain layar untuk elemen UI di aplikasi Anda.

Sebagai tambahan, produsen perangkat mungkin menambahkan elemen UI, gaya, dan warna sendiri untuk membedakan produk mereka. Setiap produsen menawarkan fitur berbeda dalam hal bentuk keyboard, ukuran layar, atau tombol kamera. Aplikasi yang berjalan pada satu perangkat mungkin terlihat sedikit berbeda di perangkat lain. Tantangan bagi sebagian besar developer adalah untuk mendesain elemen UI yang bisa bekerja di semua perangkat Selain itu, developer juga bertanggung jawab untuk menyediakan sumber daya aplikasi seperti ikon, logo, grafik lain, dan gaya teks untuk mempertahankan keseragaman penampilan di seluruh perangkat yang berbeda.

Memaksimalkan kinerja aplikasi

Kinerja aplikasi, seberapa cepat aplikasi berjalan, seberapa mudah aplikasi menghubungkan ke jaringan, dan seberapa baik aplikasi mengelola baterai dan penggunaan memori, dipengaruhi oleh beberapa faktor seperti daya tahan baterai, materi multimedia, dan akses Internet. Anda harus memperhatikan batasan tersebut dan menulis kode sedemikian rupa sehingga penggunaan sumber daya diseimbangkan dan didistribusikan secara optimal. Misalnya, Anda harus menyeimbangkan layanan latar belakang dengan mengaktifkannya hanya jika perlu; hal ini akan menghemat daya tahan baterai perangkat pengguna.

Membuat kode dan pengguna Anda tetap aman

Anda perlu melakukan tindakan pencegahan untuk mengamankan kode dan pengalaman pengguna saat menggunakan aplikasi. Gunakan alat seperti ProGuard (disediakan di Android Studio), yang mendeteksi dan membuang kelas, bidang, metode, dan atribut yang tidak digunakan serta mengenkripsi semua kode dan sumber daya aplikasi sewaktu memaketkan aplikasi. Untuk melindungi informasi penting milik pengguna seperti proses masuk dan sandi, Anda harus mengamankan saluran komunikasi untuk melindungi data yang bergerak (di Internet) serta data yang tidak bergerak (di perangkat).

Tetap kompatibel dengan versi platform yang lebih lama

    - Ringkasan UI
    - Versi Platform
    - Mendukung Versi Platform Berbeda
- Lainnya:
    - Panduan Pengguna Android Studio: Image Asset Studio
    - Wikipedia: Rangkuman Riwayat Versi Android

1.1: Buat Aplikasi Android Anda yang Pertama

Materi:

- Proses development
- Menggunakan Android Studio
- Menjelajahi proyek
- Menampilkan dan mengedit kode Java
- Menampilkan dan mengedit layout
- Memahami proses pembangunan
- Menjalankan aplikasi di emulator atau perangkat
- Menggunakan log
- Praktik terkait
- Ketahui selengkapnya

Bab ini menjelaskan cara mengembangkan aplikasi menggunakan Android Studio Integrated Development Environment (IDE).

Proses development

Proyek aplikasi Android dimulai dengan ide dan definisi persyaratan yang diperlukan untuk mewujudkan ide tersebut. Saat proyek berjalan, maka akan melalui desain, development, dan pengujian.

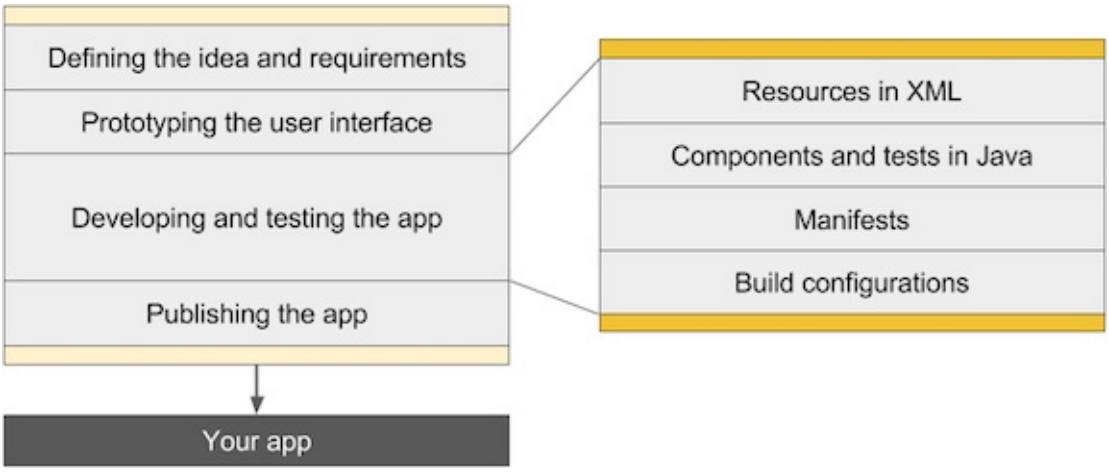


Diagram di atas adalah gambar proses development tingkat tinggi, dengan langkah-langkah berikut:

- *Mendefinisikan ide dan persyaratannya:* Sebagian besar aplikasi dimulai dengan ide tentang hal yang harus dilakukan, yang didukung pasar dan penelitian pengguna. Selama tahap ini, persyaratan aplikasi didefinisikan.
- *Membuat prototipe antarmuka pengguna:* Gunakan gambar, rekaan, dan prototipe untuk menampilkan tampilan antarmuka pengguna nantinya, dan cara kerjanya.
- *Mengembangkan dan menguji aplikasi:* Aplikasi terdiri dari satu atau beberapa aktivitas. Untuk setiap aktivitas, Anda bisa menggunakan Android Studio untuk melakukan hal berikut, tidak dalam urutan tertentu:
  - *Buat layout:* Tempatkan elemen UI pada layar di layout, dan tetapkan sumber daya string serta item menu, menggunakan Extensible Markup Language (XML).
  - *Tulis kode Java:* Buat referensi kode sumber untuk komponen dan pengujian, serta gunakan alat pengujian dan debug.
  - *Daftarkan aktivitas:* Deklarasikan aktivitas dalam file manifes.
  - *Definisikan versi:* Gunakan konfigurasi pembangunan default atau buat pembangunan khusus untuk versi aplikasi yang berbeda.
- *Mempublikasikan aplikasi*: Rakit APK (file paket) final dan distribusikan melalui saluran seperti Google Play.

Menggunakan Android Studio

Android Studio menyediakan alat untuk pengujian, dan mempublikasikan tahap proses development, serta lingkungan development terpadu untuk membuat aplikasi bagi semua perangkat Android. Lingkungan development menyertakan kode template dengan kode contoh untuk fitur aplikasi umum, alat pengujian dan kerangka kerja yang banyak, dan sistem pembangunan yang fleksibel.

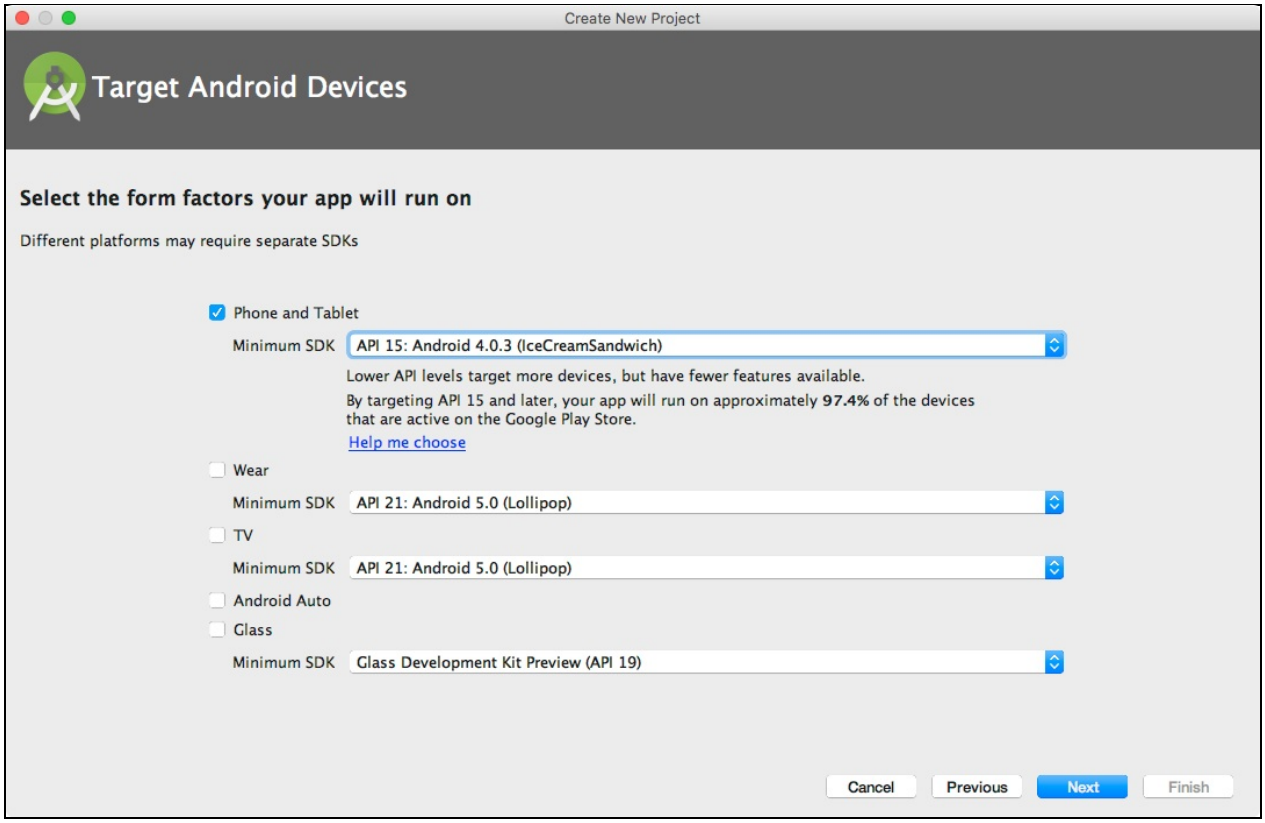
Memulai proyek Android Studio

Setelah berhasil memasang Android Studio IDE, klik dua kali ikon aplikasi Android Studio untuk memulainya. Pilih **Start a new Android Studio project** di jendela Welcome, dan beri nama proyek dengan nama yang sama yang ingin Anda gunakan untuk aplikasi.

Saat memilih Domain Perusahaan unik, ingat bahwa aplikasi yang dipublikasikan ke Google Play harus memiliki nama paket unik. Karena domain bersifat unik, yang mengawali nama aplikasi dengan nama Anda, atau nama domain perusahaan, sebaiknya sediakan nama paket unik yang memadai. Jika tidak berencana untuk mempublikasikan aplikasi, Anda bisa menerima domain contoh default. Ketahuilah bahwa mengubah nama paket di lain waktu memerlukan kerja tambahan.

Memilih perangkat target dan SDK minimum

Saat memilih Target Android Devices, Phone and Tablet dipilih secara default, seperti yang ditampilkan dalam gambar di bawah ini. Pilihan yang ditampilkan dalam gambar untuk Minimum SDK — **API 15: Android 4.0.3 (IceCreamSandwich)** — membuat aplikasi Anda kompatibel dengan 97% perangkat Android yang aktif di Google Play Store.



Perangkat berbeda menjalankan versi sistem Android yang berbeda, seperti Android 4.0.3 atau Android 4.4. Setiap versi yang berurutan umumnya menambahkan API baru yang tidak tersedia di versi sebelumnya. Untuk menunjukkan rangkaian API yang tersedia, setiap versi menetapkan API level. Misalnya, Android 1.0 adalah API level 1 dan Android 4.0.3 adalah API level 15.

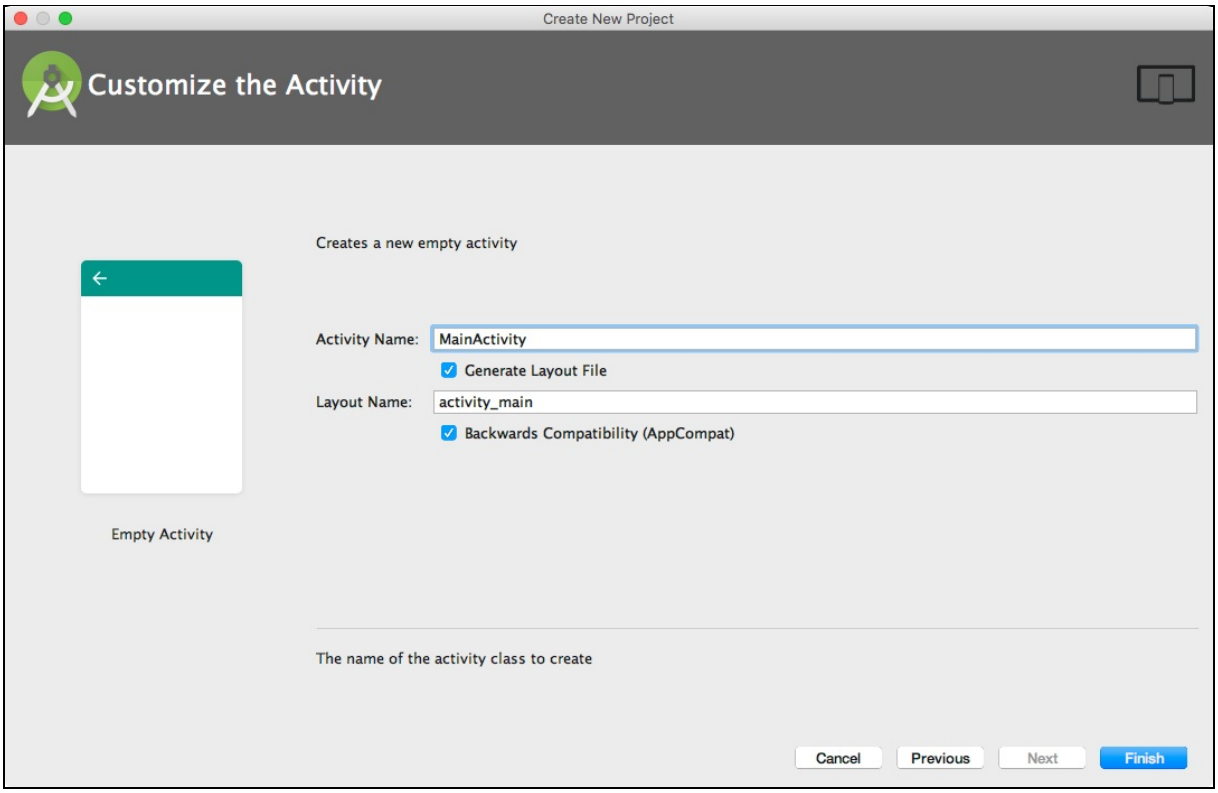
Minimum SDK mendeklarasikan versi Android minimum untuk aplikasi Anda. Setiap versi Android yang berurutan menyediakan kompatibilitas untuk aplikasi yang dibangun menggunakan API dari versi sebelumnya, sehingga aplikasi Anda akan *selalu* kompatibel dengan versi Android mendatang sambil menggunakan Android API yang didokumentasikan.

Memilih template

Android Studio mengumpulkan terlebih dulu proyek Anda dengan kode minimum untuk aktivitas dan layout layar berdasarkan *template*. Tersedia berbagai macam template, mulai dari template kosong virtual (Add No Activity) hingga beragam tipe aktivitas.

Anda bisa menyesuaikan aktivitas setelah memilih template. Misalnya, template Empty Activity menyediakan satu aktivitas yang disertai satu sumber daya layout untuk layar. Anda bisa memilih untuk menerima nama yang biasa digunakan untuk aktivitas (seperti **MainActivity**) atau mengubah nama di layar Customize Activity . Selain itu, jika Anda menggunakan template Empty Activity, pastikan memeriksa yang berikut ini jika belum diperiksa:

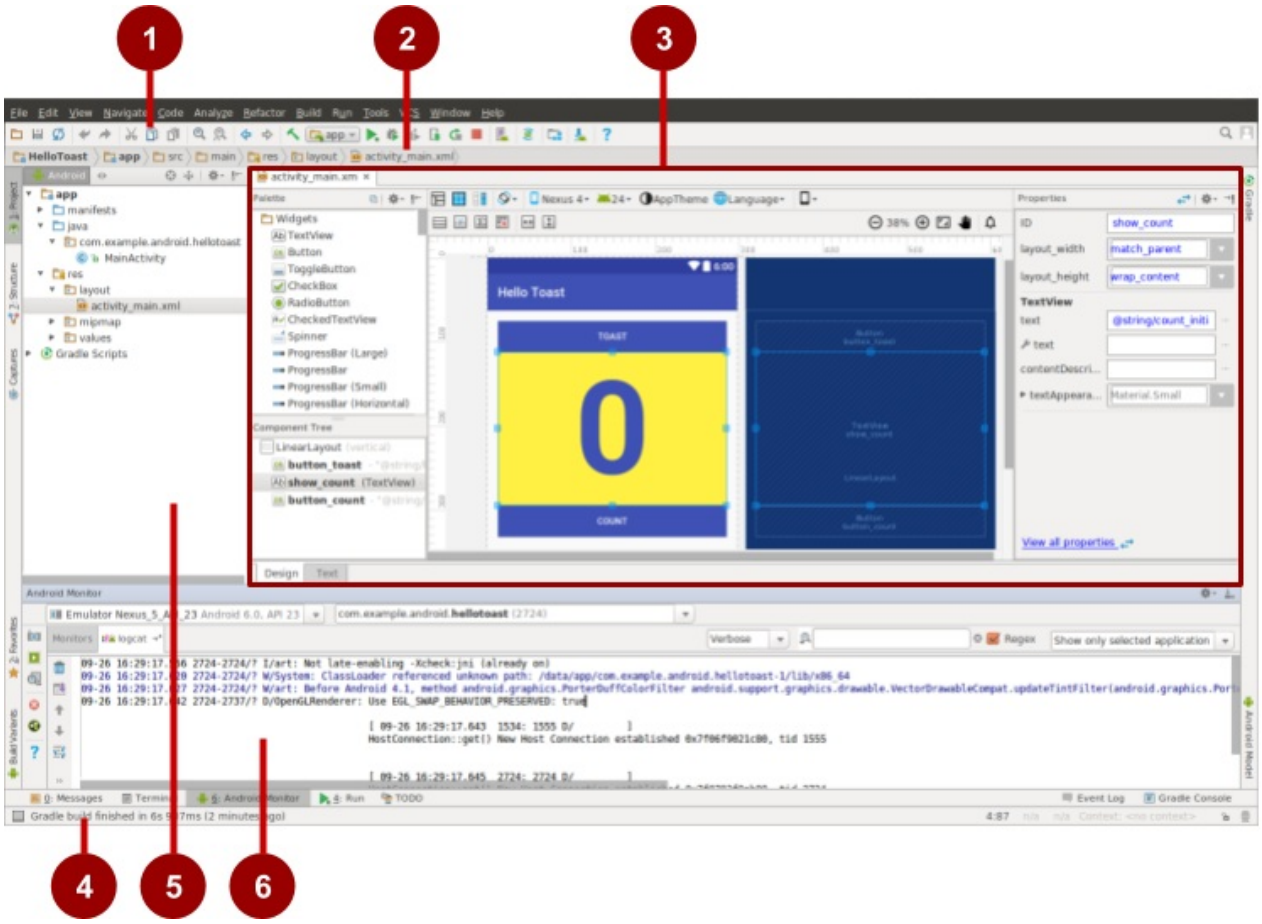
- Generate Layout File: Biarkan ini dicentang untuk membuat sumber daya layout yang terhubung dengan aktivitas ini, yang biasanya diberi nama **activity_main.xml**. Layout mendefinisikan antarmuka pengguna untuk aktivitas.
- Backwards Compatibility (AppCompat)**:** Biarkan ini dicentang untuk menyertakan pustaka AppCompat sehingga aplikasi kompatibel dengan Android versi sebelumnya bahkan jika menggunakan fitur yang hanya ditemukan di versi yang lebih baru.



Android Studio membuat folder untuk proyek yang baru saja dibuat di folder AndroidStudioProjects pada komputer Anda.

Panel jendela Android Studio

Jendela utama Android Studio terdiri dari sejumlah area logis, atau *panel*, seperti yang ditampilkan dalam gambar di bawah ini.



Dalam gambar di atas:

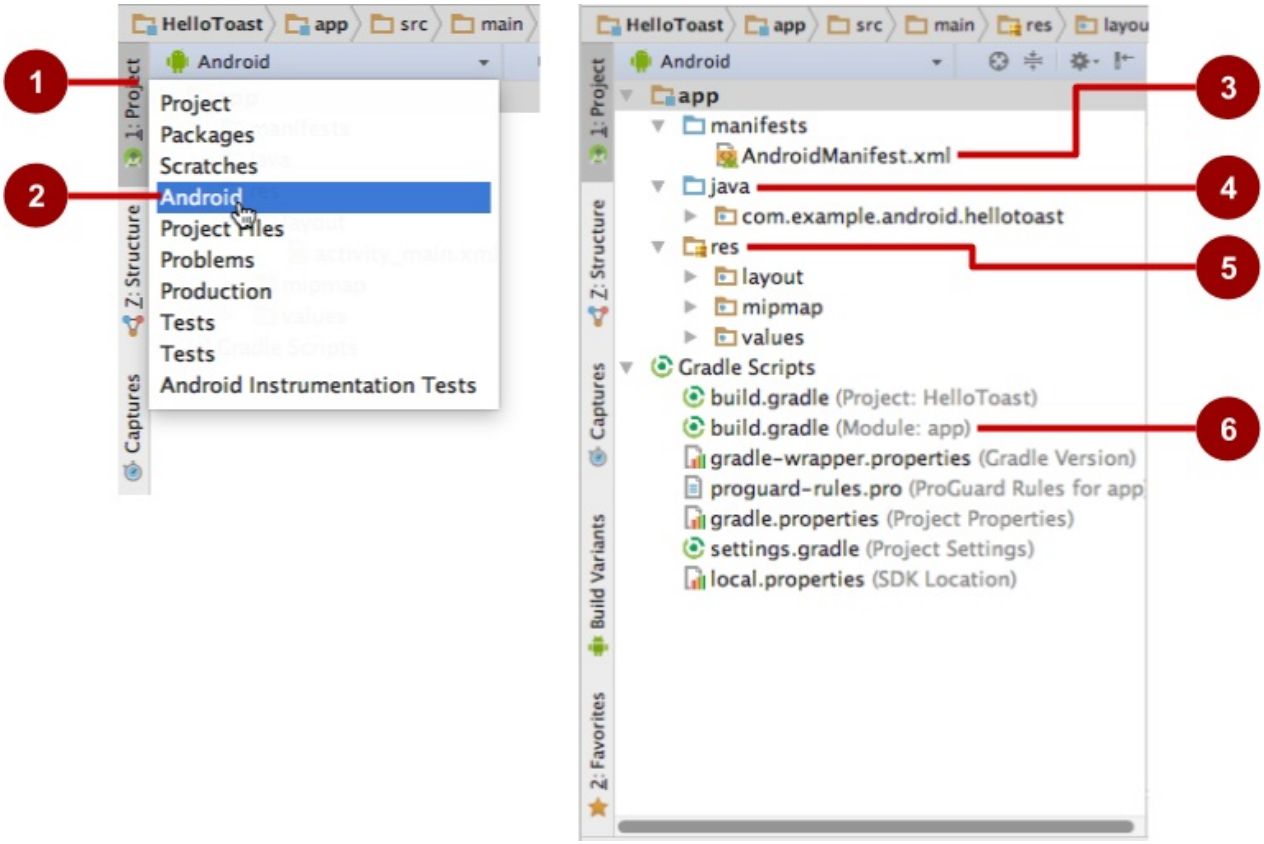
1. **Bilah Alat**. Bilah alat menjalankan beragam tindakan, termasuk menjalankan aplikasi Android dan meluncurkan alat Android.
2. **Bilah Navigasi**. Bilah navigasi memungkinkan navigasi melalui proyek dan membuka file untuk pengeditan. Bilah navigasi menyediakan tampilan struktur proyek yang lebih ringkas.
3. **Panel Editor**. Panel ini menampilkan materi file yang dipilih dalam proyek. Misalnya, setelah memilih layout (seperti yang ditampilkan dalam gambar), panel ini menampilkan editor layout dengan alat untuk mengedit layout. Setelah memilih file kode Java, panel ini menampilkan kode dengan alat untuk mengedit kode.
4. **Bilah Status.** Bilah status menampilkan status proyek dan Android Studio itu sendiri, serta peringatan atau pesan apa pun. Anda bisa mengamati kemajuan pembangunan di bilah status.
5. **Panel Project**. Panel proyek menampilkan file proyek dan hierarki proyek.
6. **Panel Monitor**. Panel monitor menawarkan akses ke daftar TODO untuk mengelola tugas, Android Monitor untuk memantau eksekusi aplikasi (ditampilkan dalam gambar), logcat untuk menampilkan pesan log, dan aplikasi Terminal untuk melakukan aktivitas Terminal.

Tip: Anda bisa mengatur jendela utama untuk memberi Anda lebih banyak ruang layar dengan menyembunyikan atau memindahkan panel. Anda juga bisa menggunakan pintasan keyboard untuk mengakses lebih banyak fitur. Lihat Pintasan Keyboard untuk daftar lengkap.

Menjelajahi proyek

Setiap proyek di Android Studio berisi file AndroidManifest.xml, file kode sumber komponen, dan file sumber daya terkait. Secara default, Android Studio mengatur file proyek Anda berdasarkan jenis file, dan menampilkannya dalam proyek: Tampilan Android di panel alat (bantu) kiri, seperti yang ditampilkan di bawah ini. Tampilan menyediakan akses cepat ke file kunci proyek Anda.

Untuk beralih kembali ke tampilan ini dari tampilan lainnya, klik tab **Project** vertikal di kolom paling kiri panel Project, dan pilih **Android** dari menu munculan di bagian atas panel Project, seperti yang ditampilkan dalam gambar di bawah ini.



Dalam gambar di atas:

1. Tab **Project** . Klik untuk menampilkan tampilan proyek.
2. Pilihan **Android** di menu tarik-turun proyek.
3. File **AndroidManifest.xml**. Digunakan untuk menetapkan informasi tentang aplikasi bagi lingkungan waktu proses Android. Template yang Anda pilih membuat file ini.
4. Folder **java**. Folder ini menyertakan aktivitas, pengujian, dan komponen lain di kode sumber Java. Setiap aktivitas, layanan, dan komponen lain didefinisikan sebagai kelas Java, biasanya dalam file miliknya. Nama aktivitas pertama (layar) yang pengguna lihat, yang juga melakukan inisialisasi sumber daya seluruh aplikasi, biasanya adalah MainActivity.
5. Folder **res**. Folder ini menyimpan sumber daya, seperti layout XML, string UI, dan gambar. Aktivitas biasanya dikaitkan dengan file sumber daya XML yang menetapkan layout tampilannya. File ini biasanya diberi nama setelah aktivitas atau fungsinya.
6. File **build.gradle (Module: App
7.)*. *File ini menetapkan konfigurasi pembangunan modul. Template yang Anda pilih membuat file ini, yang mendefinisikan konfigurasi pembangunan, termasuk atribut* `minSdkVersion` *yang mendeklarasikan versi minimum untuk aplikasi, dan atribut* `targetSdkVersion` *yang mendeklarasikan versi tertinggi (terbaru) untuk aplikasi yang dioptimalkan. File ini juga menyertakan daftar* dependensi*, yaitu pustaka yang diperlukan oleh kode — seperti pustaka AppCompat untuk mendukung beragam versi Android.

1

Menampilkan Manifes Android

Sebelum sistem Android bisa memulai komponen aplikasi, sistem harus mengetahui bahwa komponen ada dengan membaca file AndroidManifest.xml aplikasi. Aplikasi harus mendeklarasikan semua komponennya dalam file ini, yang harus berada pada akar direktori proyek aplikasi.

Untuk menampilkan file ini, luaskan folder manifes di Project: Tampilan Android, dan klik dua kali file (**AndroidManifest.xml**). Isinya muncul di panel pengeditan seperti yang ditampilkan dalam gambar di bawah ini.

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Hello World"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Namespace dan tag aplikasi Android

Manifes Android dikodekan di XML dan selalu menggunakan namespace Android:

```
xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.android.helloworld">
```

Ekspresi `package` menampilkan nama paket unik aplikasi baru. Jangan mengubah ini setelah aplikasi dipublikasikan.

```
<application
  ...
</application>
```

Tag `<application` , dengan tag `</application>` penutupnya, mendefinisikan setelan manifes untuk keseluruhan aplikasi.

Cadangan otomatis

Atribut `android:allowBackup` memungkinkan pencadangan data aplikasi otomatis:

```
...
android:allowBackup="true"
...
```

Menyetel atribut `android:allowBackup` ke `true` memungkinkan aplikasi dicadangkan secara otomatis dan dipulihkan jika perlu. Pengguna menggunakan waktu dan upaya untuk mengonfigurasi aplikasi. Beralih ke perangkat baru bisa membatalkan seluruh konfigurasi yang dibuat dengan susah payah. Sistem melakukan cadangan otomatis ini untuk hampir seluruh data aplikasi secara default, dan melakukannya tanpa mengharuskan developer menulis kode aplikasi tambahan lainnya.

Untuk aplikasi dengan versi SDK target berupa Android 6.0 (API level 23) dan yang lebih tinggi, perangkat yang menjalankan Android 6.0 dan yang lebih tinggi secara otomatis membuat cadangan data aplikasi ke awan karena atribut `android:allowBackup` default ke `true` jika dihilangkan. Untuk aplikasi < API level 22, Anda harus secara eksplisit menambahkan atribut `android:allowBackup` dan menyetelnya ke `true` .

Tip: Untuk mengetahui selengkapnya tentang pencadangan otomatis untuk aplikasi, lihat Mengonfigurasi Cadangan Otomatis untuk Aplikasi.

Ikon aplikasi

Metode `android:icon` menyetel ikon untuk aplikasi:

```
    ...
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    ...
```

Atribut `android:icon` menetapkan ikon di folder **mipmap** (di dalam folder **res** di Project: tampilan Android) untuk aplikasi. Ikon muncul di Peluncur untuk meluncurkan aplikasi. Ikon juga digunakan sebagai ikon default untuk komponen aplikasi.

Sumber daya string dan label aplikasi

Seperti yang bisa Anda lihat di gambar sebelumnya, atribut `android:label` menampilkan string `"Hello World"` yang disorot. Jika Anda mengeklik string ini, maka string berubah untuk menampilkan sumber daya string `@string/app_name`:

```
    ...
    android:label="@string/app_name"
    ...
```

Tip: Ctrl - klik atau klik-kanan `app_name` di panel edit untuk melihat menu konteks. Pilih **Go To > Declaration** untuk melihat lokasi sumber daya string dideklarasikan: dalam file strings.xml. Bila Anda memilih **Go To > Declaration** atau membuka file dengan mengeklik dua kali **strings.xml** di Project: Tampilan Android (di dalam folder **values**), isinya muncul di panel pengeditan.

Setelah membuka file strings.xml, Anda bisa melihat bahwa nama string `app_name` disetel ke `Hello World`. Anda bisa mengubah nama aplikasi dengan mengubah string `Hello World` ke hal lain. Sumber daya string dijelaskan dalam pelajaran terpisah.

Tema aplikasi

Atribut `android:theme` menyetel tema aplikasi, yang mendefinisikan penampilan elemen antarmuka pengguna seperti teks:

```
    ...
    android:theme="@style/AppTheme">
    ...
```

Atribut `theme` disetel ke tema standar `AppTheme`. Tema dijelaskan dalam pelajaran terpisah.

Mendeklarasikan versi Android

Perangkat yang berbeda mungkin menjalankan versi sistem Android yang berbeda, seperti Android 4.0 atau Android 4.4. Setiap versi yang berurutan bisa menambahkan API baru yang tidak tersedia di versi sebelumnya. Untuk menunjukkan rangkaian API yang tersedia, setiap versi menetapkan API level. Misalnya, Android 1.0 adalah API level 1 dan Android 4.4 adalah API level 19.

API level memungkinkan developer untuk mendeklarasikan versi minimum dengan aplikasi yang kompatibel, menggunakan tag manifes `<uses-sdk>` dan atribut `minSdkVersion`. Misalnya, Calendar Provider API telah ditambahkan di Android 4.0 (API level 14). Jika aplikasi Anda tidak berfungsi tanpa API tersebut, deklarasikan API level 14 sebagai versi yang didukung minimum aplikasi seperti ini:

```
<manifest ... >
    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="19" />
    ...
</manifest>
```

Atribut `minSdkVersion` mendeklarasikan versi minimum untuk aplikasi, dan atribut `targetSdkVersion` mendeklarasikan versi tertinggi (terbaru) yang telah dioptimalkan dalam aplikasi. Setiap versi Android yang berurutan menyediakan kompatibilitas untuk aplikasi yang dibangun menggunakan API dari versi sebelumnya, sehingga aplikasi akan *selalu* kompatibel dengan versi Android mendatang sambil menggunakan Android API yang didokumentasikan.

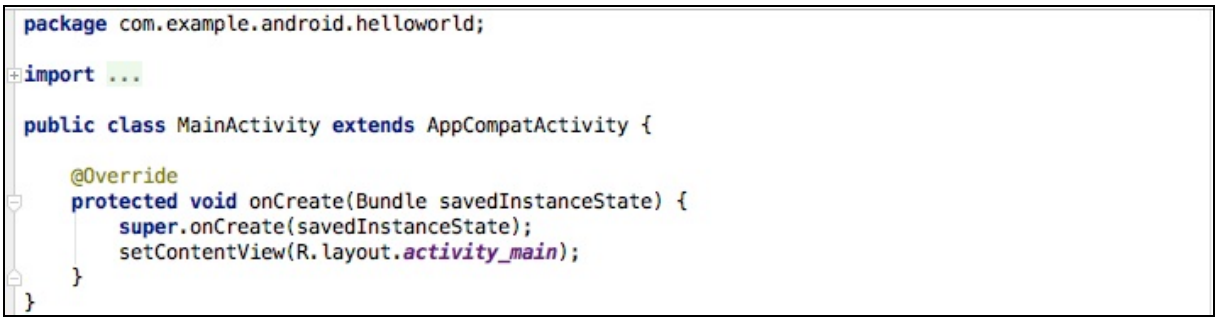
Atribut `targetSdkVersion` *tidak* mencegah aplikasi dipasang pada versi Android yang lebih tinggi (yang lebih baru) dibandingkan nilai yang ditetapkan, namun hal ini penting karena menunjukkan pada sistem apakah aplikasi harus mewarisi perubahan perilaku di versi yang baru. Jika Anda tidak memperbarui `targetSdkVersion` ke versi terbaru, sistem akan menganggap bahwa aplikasi memerlukan perilaku kompatibilitas mundur saat dijalankan pada versi terbaru. Misalnya, di antara perubahan perilaku di Android 4.4, alarm yang dibuat dengan AlarmManager API saat ini tidak tepat secara default sehingga sistem bisa mengumpulkan alarm aplikasi dan mempertahankan kekuatan sistem, namun sistem akan menahan perilaku API sebelumnya untuk aplikasi jika API level target Anda lebih rendah daripada `"19"`.

Menampilkan dan mengedit kode Java

Komponen ditulis di Java dan dicantumkan dalam folder modul di folder **java** dalam Project: Tampilan Android. Setiap nama modul diawali dengan nama domain (seperti **com.example.android**) dan menyertakan nama aplikasi.

Contoh berikut menampilkan komponen *aktivitas*:

1. Klik folder modul untuk meluaskan dan menampilkan file MainActivity untuk aktivitas yang ditulis di Java (kelas `MainActivity`).
2. Klik dua kali **MainActivity** untuk melihat file sumber di panel pengeditan, seperti yang ditampilkan dalam gambar di bawah ini.



Bagian paling atas file MainActivity.java adalah pernyataan `package` yang mendefinisikan paket aplikasi. Ini diikuti dengan blok `import` yang diringkas dalam gambar di atas, dengan " `...` ". Klik titik untuk meluaskan blok guna menampilkannya. Pernyataan `import` akan mengimpor pustaka yang diperlukan untuk aplikasi, seperti berikut ini, yang mengimpor pustaka AppCompatActivity:

```
import android.support.v7.app.AppCompatActivity;
```

Setiap aktivitas dalam aplikasi diimplementasikan sebagai kelas Java. Deklarasi kelas berikut memperluas kelas `AppCompatActivity` untuk mengimplementasikan fitur dengan cara yang kompatibel-mundur dengan Android versi sebelumnya:

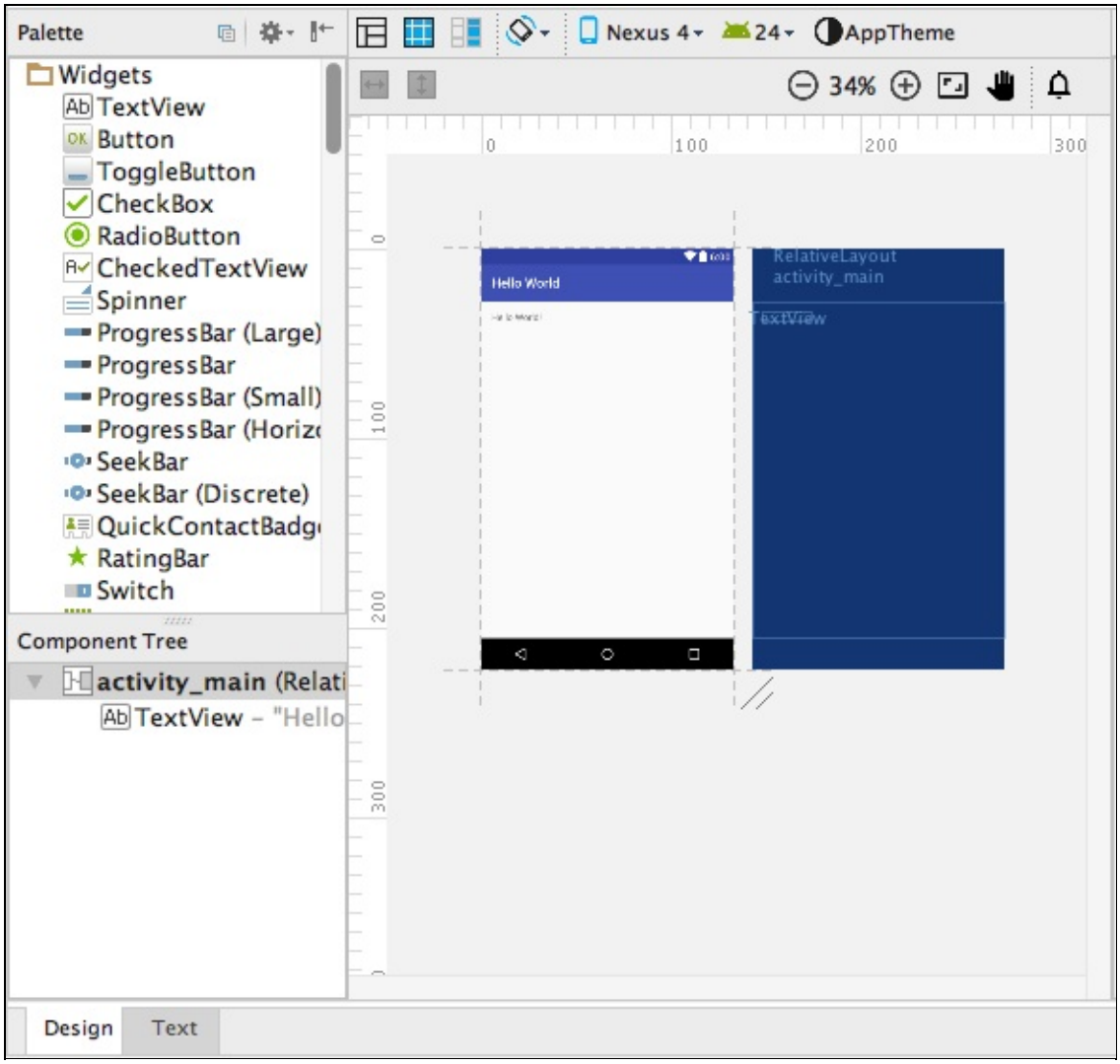
```
MainActivity kelas publik memperluas AppCompatActivity {
    ...
}
```

Seperti yang Anda pelajari terdahulu, sebelum sistem Android bisa memulai komponen aplikasi seperti aktivitas, sistem harus mengetahui bahwa aktivitas ada dengan membaca file AndroidManifest.xml aplikasi. Oleh karena itu, setiap aktivitas harus dicantumkan di file AndroidManifest.xml.

Menampilkan dan mengedit layout

Sumber daya layout ditulis dalam XML dan dicantumkan dalam folder **layout** di folder **res** dalam Project: Tampilan Android. Klik **res > layout**, kemudian klik dua kali **activity_main.xml** untuk melihat file layout di panel pengeditan.

Android Studio menampilkan tampilan Desain layout, seperti yang ditampilkan dalam gambar di bawah ini. Tampilan ini menyediakan panel Palette elemen antarmuka pengguna, dan petak yang menampilkan layout layar.



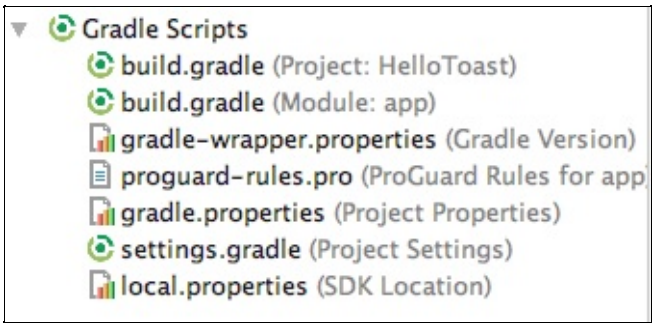
Memahami proses pembangunan

Android Application Package (APK) adalah format file paket untuk mendistribusikan dan memasang aplikasi seluler Android. Proses pembangunan melibatkan alat dan proses yang secara otomatis mengonversi setiap proyek menjadi APK.

Android Studio menggunakan Gradle sebagai dasar sistem pembangunan, dengan kemampuan khusus Android lebih banyak yang disediakan oleh Android Plugin for Gradle. Sistem pembangunan ini berjalan sebagai alat (bantu) terintegrasi dari menu Android Studio.

Memahami file build.gradle

Bila Anda membuat proyek, Android Studio secara otomatis menghasilkan file pembangunan penting di folder **Gradle Scripts** dalam Project: Tampilan Android. File pembangunan Android Studio diberi nama **build.gradle** seperti yang ditampilkan di bawah ini:



Setiap proyek memiliki hal berikut:

build.gradle (Project: *apptitle*)

Ini adalah file pembangunan tingkat atas untuk keseluruhan proyek, berada di akar direktori proyek, yang mendefinisikan konfigurasi pembangunan yang berlaku untuk semua modul di proyek Anda. File ini, yang dihasilkan oleh Android Studio, tidak boleh diedit untuk menyertakan dependensi aplikasi.

build.gradle (Module: app)

Android Studio membuat file `build.gradle (Module: app)` terpisah untuk setiap modul. Anda bisa mengedit setelan pembangunan guna menyediakan opsi pemaketan khusus untuk setiap modul, seperti tipe pembangunan tambahan dan ragam produk, dan untuk menggantikan setelan di file manifes atau build.gradle tingkat atas. File ini paling sering adalah file untuk mengedit ketika mengubah konfigurasi tingkat aplikasi, seperti mendeklarasikan dependensi di bagian `dependencies` . Yang berikut ini menampilkan isi file `build.gradle (Module: app)` proyek:

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

android {
    compileSdkVersion 24
    buildToolsVersion "24.0.1"
    defaultConfig {
        applicationId "com.example.android.helloworld2"
        minSdkVersion 15
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        rilis {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:24.2.1'
    testCompile 'junit:junit:4.12'
}
```

File build.gradle menggunakan sintaks Gradle. Gradle adalah Domain Specific Language (DSL) untuk menjelaskan dan memanipulasi logika pembangunan dengan menggunakan Groovy, yang merupakan bahasa dinamis untuk Java Virtual Machine (JVM). Anda tidak perlu mempelajari Groovy untuk melakukan perubahan, karena Android Plugin for Gradle

memperkenalkan sebagian besar elemen DSL yang Anda perlukan.

Tip: Untuk mengetahui selengkapnya tentang DSL plugin Android, baca Dokumentasi referensi DSL.

Blok Plugin dan Android.

Dalam file `build.gradle (Module: app)` di atas, pernyataan pertama menerapkan tugas pembangunan plug-in Gradle khusus Android:

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

android {
   ...
}
```

Blok `android { }` menetapkan hal berikut ini untuk versi:

- Versi SDK target untuk mengompilasi kode:

  ```
  compileSdkVersion 24
  ```

- Versi alat pembangunan untuk digunakan membangun aplikasi:

  ```
  buildToolsVersion "24.0.1"
  ```

Blok defaultConfig

Setelan inti dan entri untuk aplikasi ditetapkan di blok `defaultConfig { }` dalam `android { } block:`

```
   ...
   defaultConfig {
      applicationId "com.example.hello.helloworld"
      minSdkVersion 15
      targetSdkVersion 23
      versionCode 1
      versionName "1.0"
            testInstrumentationRunner
               "android.support.test.runner.AndroidJUnitRunner"
   }
...
```

Setelan `minSdkVersion` dan `targetSdkVersion` menggantikan setelan AndroidManifest.xml untuk versi SDK minimum dan versi SDK target. Lihat "Mendeklarasikan versi Android" sebelumnya di bagian ini untuk informasi latar belakang tentang setelan ini.

Pernyataan `testInstrumentationRunner` ini menambahkan dukungan instrumentasi untuk menguji antarmuka pengguna dengan Espresso dan UIAutomator. Hal ini dijelaskan dalam pelajaran terpisah.

Tipe pembangunan

Tipe pembangunan untuk aplikasi ditetapkan di blok `buildTypes { }` , yang mengontrol cara aplikasi dibangun dan dipaketkan.

```
...
buildTypes {
   rilis {
      minifyEnabled false
      proguardFiles getDefaultProguardFile('proguard-android.txt'),
                                                'proguard-rules.pro'
   }
}
...
```

Tipe pembangunan yang ditetapkan adalah `release` untuk rilis aplikasi. Tipe pembangunan umum lainnya adalah `debug` . Mengonfigurasi tipe pembangunan dijelaskan dalam pelajaran terpisah.

Dependensi

Dependensi untuk aplikasi didefinisikan di blok `dependencies { }` , yang merupakan bagian file build.gradle yang paling mungkin berubah saat Anda mulai mengembangkan kode yang bergantung pada pustaka. Blok adalah bagian dari Gradle API standar dan termasuk *di luar* blok `android { }` .

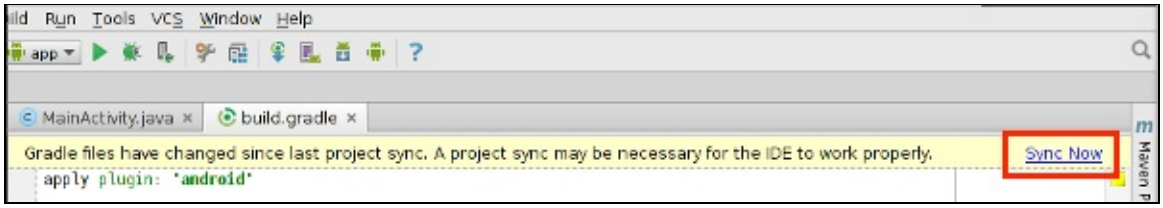
```
...
dependencies {
   compile fileTree(dir: 'libs', include: ['*.jar'])
   androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
exclude group: 'com.android.support', module: 'support-annotations'
   })
   compile 'com.android.support:appcompat-v7:24.2.0'
   testCompile 'junit:junit:4.12'
}
```

Dalam cuplikan di atas, pernyataan `compile fileTree(dir: 'libs', include: ['*.jar'])` menambahkan dependensi semua file ".jar" di dalam direktori `libs` . Konfigurasi `compile` akan mengompilasi aplikasi utama — segala sesuatu di dalamnya ditambahkan ke classpath kompilasi, dan juga dipaketkan ke dalam APK final.

Menyinkronkan proyek Anda

Bila Anda membuat perubahan pada file konfigurasi pembangunan dalam proyek, Android Studio akan mengharuskan Anda untuk melakukan *sinkronisasi* file proyek sehingga Android Studio bisa mengimpor perubahan konfigurasi pembangunan dan menjalankan beberapa pemeriksaan untuk memastikan konfigurasi tidak akan menimbulkan kesalahan pembangunan.

Untuk menyinkronkan file proyek, klik **Sync Now** di bilah notifikasi yang muncul saat membuat perubahan, atau klik **Sync Project** dari bilah menu. Jika Android Studio memperlihatkan kesalahan apa pun dengan konfigurasi — misalnya, jika kode sumber menggunakan fitur API yang hanya tersedia di API level yang lebih tinggi dari `compileSdkVersion` — jendela Messages muncul untuk menjelaskan masalah.



Menjalankan aplikasi pada emulator atau perangkat

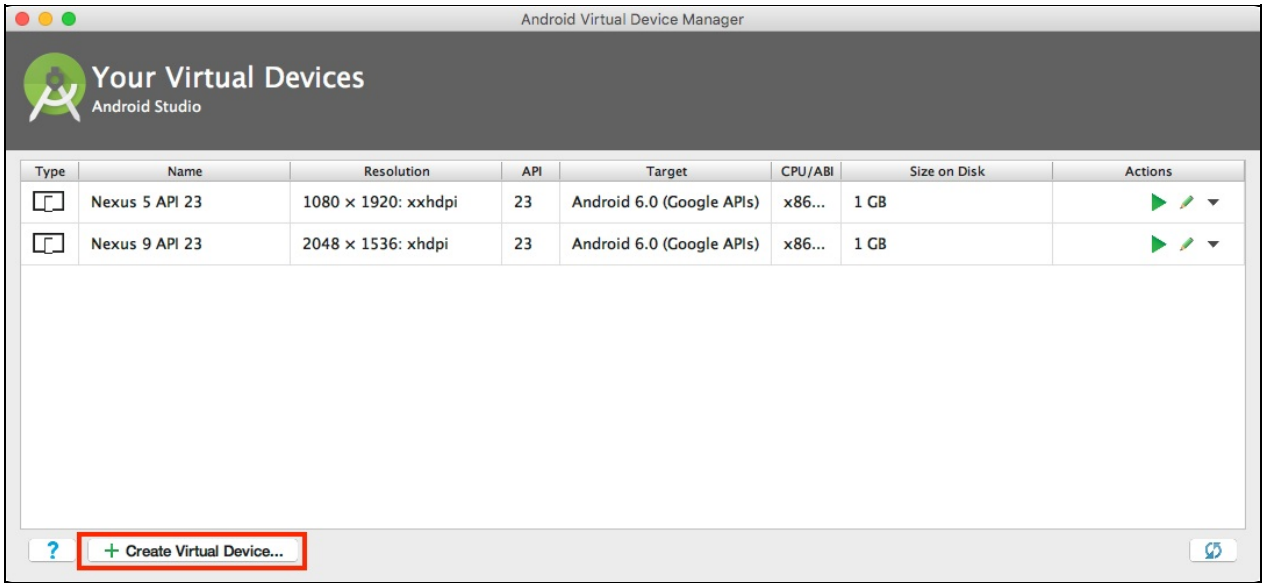
Dengan emulator perangkat virtual, Anda bisa menguji aplikasi pada perangkat yang berbeda seperti tablet atau ponsel cerdas — dengan API level yang berbeda untuk versi Android yang berbeda — untuk memastikan aplikasi terlihat bagus dan berfungsi untuk sebagian besar pengguna. Meskipun ini adalah ide yang bagus, Anda tidak harus bergantung pada memiliki perangkat fisik yang tersedia untuk development aplikasi.

Pengelola Android Virtual Device (AVD) membuat perangkat virtual atau emulator yang menyimulasikan konfigurasi untuk tipe perangkat Android tertentu. Gunakan AVD Manager untuk mendefinisikan karakteristik perangkat keras perangkat dan API levelnya, dan untuk menyimpannya sebagai konfigurasi perangkat virtual. Ketika Anda memulai emulator Android, emulator akan membaca konfigurasi yang ditetapkan dan membuat perangkat emulasi pada komputer yang berperilaku sama persis dengan versi fisik perangkat.

Membuat perangkat virtual

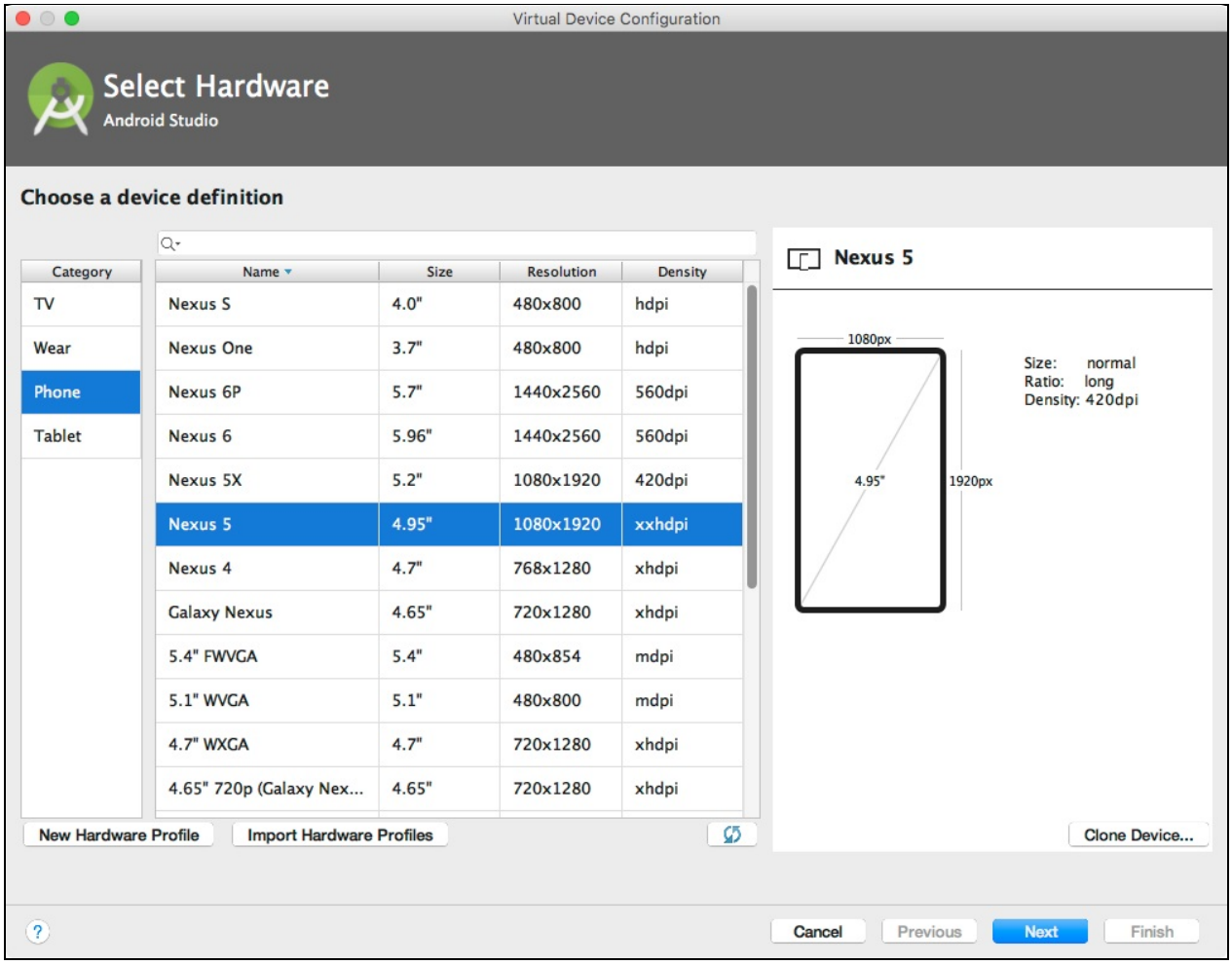
Untuk menjalankan emulator pada komputer, gunakan AVD Manager untuk membuat konfigurasi yang menjelaskan perangkat virtual. Pilih **Tools > Android > AVD Manager**, atau klik ikon AVD Manager  di bilah alat.

Layar "Your Virtual Devices" muncul menampilkan semua perangkat virtual yang dibuat sebelumnya. Klik tombol **+Create Virtual Device** untuk membuat perangkat virtual baru.



Anda bisa memilih perangkat dari daftar perangkat keras yang telah didefinisikan sebelumnya. Untuk setiap perangkat, tabel menampilkan ukuran tampilan diagonal (Size), resolusi layar dalam piksel (Resolution), dan kepadatan piksel (Density). Misalnya, kepadatan piksel perangkat Nexus 5 adalah `xxhdpi`, artinya aplikasi menggunakan ikon di folder

xxhdpi dari folder **mipmap**. Selain itu, aplikasi akan menggunakan layout dan sumber daya dapat digambar dari folder yang didefinisikan untuk kepadatan tersebut.



Anda juga memilih versi sistem Android untuk perangkat. Tab **Recommended** menampilkan sistem yang disarankan untuk perangkat. Lebih banyak versi tersedia di dalam tab **x86 Images** dan **Other Images**.

Menjalankan aplikasi pada perangkat virtual

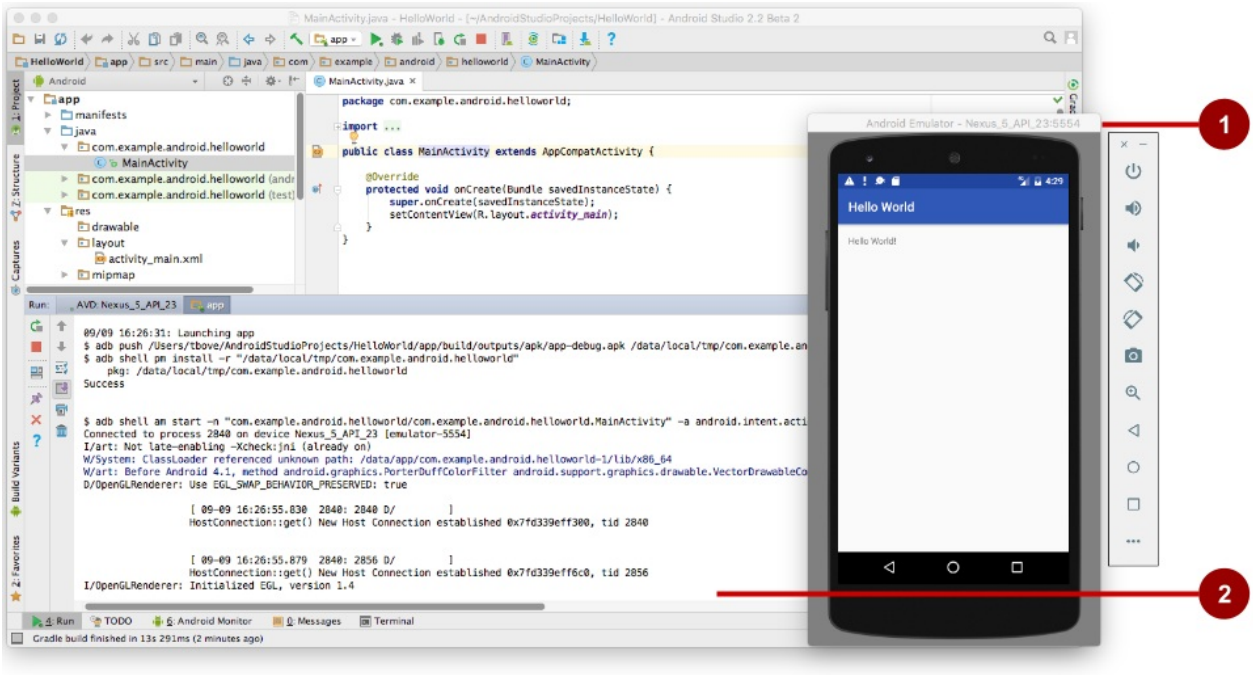
Untuk menjalankan aplikasi pada perangkat virtual yang Anda buat di bagian sebelumnya, ikuti langkah-langkah ini:

1. Di Android Studio, pilih **Run > Run app** atau klik **Run icon**  di bilah alat.
2. Di jendela Select Deployment Target, pada Emulator yang tersedia, pilih perangkat virtual yang Anda buat, dan klik **OK**.

Emulator memulai dan mem-booting seperti perangkat fisik. Ini memerlukan waktu beberapa saat, bergantung pada kecepatan komputer Anda. Aplikasi membangun, dan setelah emulator siap, Android Studio mengunggah aplikasi ke emulator dan menjalankannya.

Anda harus melihat aplikasi yang dibuat dari template Empty Activity ("Hello World") seperti yang ditampilkan dalam gambar berikut ini, yang juga menampilkan panel Run Android Studio yang menampilkan tindakan yang dilakukan untuk menjalankan aplikasi di emulator.

Catatan: Saat menguji pada emulator, praktik yang baik adalah memulainya sekali pada awal sesi, dan jangan menutupnya hingga selesai sehingga tidak perlu melalui proses booting kembali.



Dalam gambar di atas:

1. **Emulator** menjalankan aplikasi.
2. **Run Pane**. Ini menampilkan tindakan yang dilakukan untuk memasang dan menjalankan aplikasi.

Menjalankan aplikasi pada perangkat fisik

Selalu uji aplikasi Anda pada perangkat fisik, karena pengguna akan menggunakan aplikasi pada perangkat fisik. Meskipun emulator cukup bagus, emulator tidak bisa menampilkan semua kemungkinan keadaan perangkat, seperti yang terjadi jika ada panggilan masuk sewaktu aplikasi berjalan. Untuk menjalankan aplikasi pada perangkat fisik, Anda memerlukan hal berikut ini:

- Perangkat Android seperti ponsel cerdas atau tablet.
- Kabel data untuk menghubungkan perangkat Android ke komputer melalui porta USB.
- Jika Anda menggunakan Linux atau Windows, mungkin perlu melakukan langkah tambahan untuk menjalankan aplikasi pada perangkat keras. Periksa dokumentasi Menggunakan Perangkat Keras. Pada Windows, Anda mungkin perlu memasang driver USB yang sesuai untuk perangkat. Lihat Driver USB OEM.

Untuk memungkinkan Android Studio berkomunikasi dengan perangkat, aktifkan USB Debugging pada perangkat Android. Pada versi Android 4.2 dan yang lebih baru, layar opsi Developer disembunyikan secara default. Ikuti langkah-langkah ini untuk mengaktifkan USB Debugging:

1. Pada perangkat fisik, buka **Settings** dan pilih **About phone** di bagian bawah layar Settings.
2. Ketuk informasi **Build number** tujuh kali.

   Anda membacanya dengan benar: Ketuk *tujuh kali*.

3. Kembali ke layar sebelumnya (**Settings**). **Developer Options** saat ini muncul di bagian bawah layar. Ketuk **Developer options**.
4. Pilih **USB Debugging**.

Sekarang, hubungkan perangkat dan jalankan aplikasi dari Android Studio.

Pemecahan masalah koneksi perangkat

Jika Android Studio tidak mengenali perangkat, coba hal berikut:

1. Putuskan koneksi perangkat dari komputer, kemudian hubungkan kembali.

2. Mulai ulang Android Studio.
3. Jika komputer masih tidak menemukan perangkat atau mendeklarasikannya "tidak sah":
    i. Putuskan koneksi perangkat dari komputer.
    ii. Di perangkat, pilih **Settings > Developer Options**.
    iii. Ketuk **Revoke USB Debugging authorizations**.
    iv. Hubungkan kembali perangkat ke komputer.
    v. Bila dikonfirmasi, berikan otorisasi.
4. Anda mungkin perlu memasang driver USB yang sesuai untuk perangkat. Lihat dokumentasi Menggunakan Perangkat Keras.
5. Periksa dokumentasi terakhir, forum pemrograman, atau dapatkan bantuan dari instruktur Anda.

Menggunakan log

Log merupakan alat (bantu) debug andal yang bisa Anda gunakan untuk melihat nilai, jalur eksekusi, dan pengecualian. Setelah menambahkan pernyataan log ke aplikasi, pesan log Anda akan muncul bersama pesan log umum di tab logcat panel Android Monitor di Android Studio.

Untuk melihat panel Android Monitor, klik tombol **Android Monitor** di bagian bawah jendela utama Android Studio. Android Monitor menawarkan dua tab:

- Tab **logcat**. Tab **logcat** menampilkan pesan log tentang aplikasi saat aplikasi berjalan. Jika menambahkan pernyataan logging ke aplikasi, pesan log Anda dari pernyataan tersebut muncul dengan pesan log lain di bawah tab ini.
- Tab **Monitors**. Tab **Monitors** memantau kinerja aplikasi, yang bisa digunakan untuk men-debug dan menyesuaikan kode Anda.

Menambahkan pernyataan log ke aplikasi Anda

Pernyataan log menambahkan pesan apa pun yang Anda tetapkan ke log. Menambahkan pernyataan log di titik tertentu dalam kode memungkinkan developer melihat nilai, jalur eksekusi, dan pengecualian.

Misalnya, pernyataan log berikut menambahkan `"MainActivity"` dan `"Hello World"` ke log:

```
Log.d("MainActivity", "Hello World");
```

Berikut adalah elemen pernyataan ini:

- `Log` : Kelas Log adalah API untuk mengirim pesan log.
- `d` : Anda menetapkan *log level* sehingga Anda bisa memfilter pesan log menggunakan menu tarik-turun di bagian tengah panel tab **logcat**. Berikut adalah tingkat log yang bisa Anda tetapkan:
    - `d` : Pilih **Debug** atau **Verbose** untuk melihat pesan ini.
    - `e` : Pilih **Error** atau **Verbose** untuk melihat pesan ini.
    - `w` : Pilih **Warning** atau **Verbose** untuk melihat pesan ini.
    - `i` : Pilih **Info** atau **Verbose** untuk melihat pesan ini.
- `"MainActivity"` : Argumen pertama adalah *log tag* yang bisa digunakan untuk memfilter pesan pada tab **logcat**. Ini biasanya merupakan nama aktivitas tempat pesan berasal. Akan tetapi, Anda bisa membuatnya menjadi apa saja yang berguna bagi Anda untuk men-debug aplikasi. Praktik terbaik adalah untuk menggunakan konstanta sebagai tag log, seperti berikut:

1. Definisikan tag log sebagai konstanta sebelum menggunakannya dalam pernyataan log:

```
    private static final String LOG_TAG =
         MainActivity.class.getSimpleName();
```

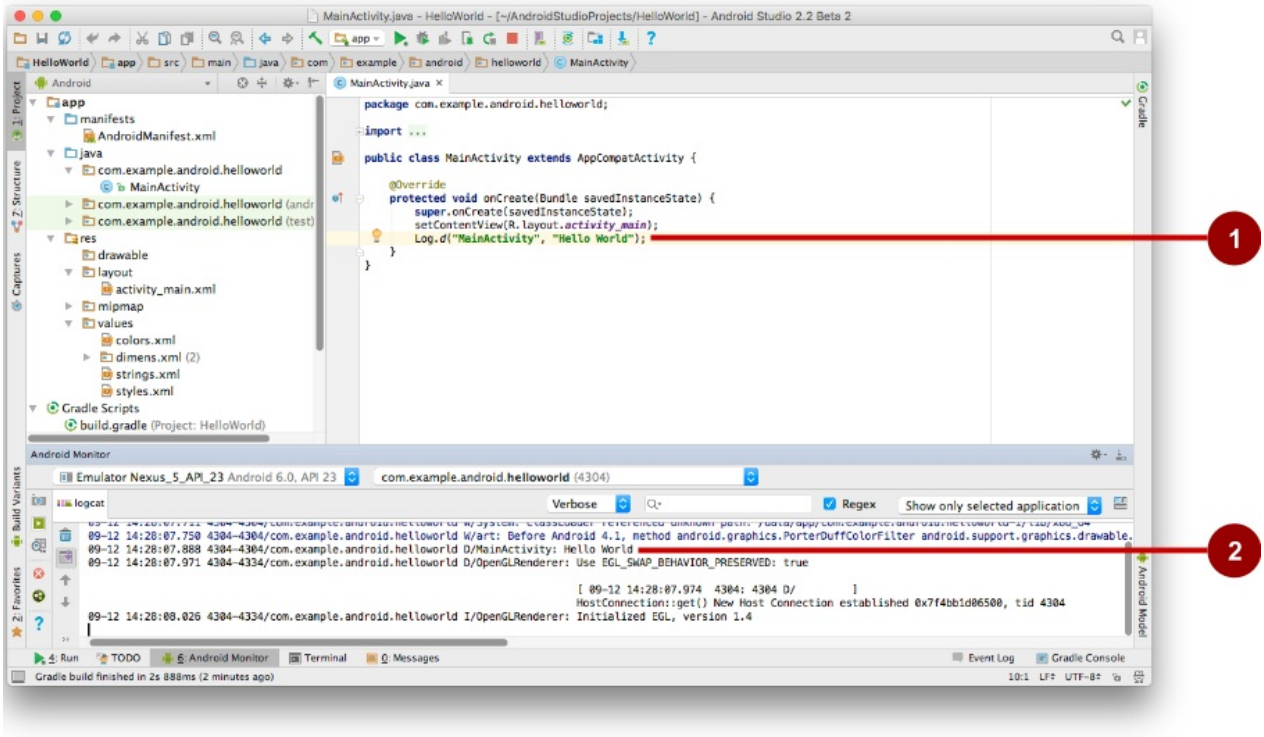
2. Gunakan konstanta dalam pernyataan log:

```
Log.d(LOG_TAG, "Hello World");
```

3. `"Hello World"` : Argumen kedua adalah pesan sebenarnya yang muncul setelah log level dan tag log pada tab **logcat**.

Menampilkan pesan log Anda

Panel Run muncul di tempat panel Android Monitor bila Anda menjalankan aplikasi di emulator atau perangkat. Setelah mulai menjalankan aplikasi, klik tombol **Android Monitor** di bagian bawah jendela utama, kemudian klik tab **logcat** di panel Android Monitor jika belum dipilih.



Dalam gambar di atas:

1. Pernyataan log di metode `onCreate()` dari `MainActivity` .
2. Panel Android Monitor yang menampilkan pesan log `logcat` , termasuk pesan dari pernyataan log.

Secara default, tampilan log disetel ke **Verbose** di menu tarik-turun di bagian atas tampilan **logcat** untuk menampilkan semua pesan. Anda bisa mengubahnya menjadi **Debug** untuk melihat pesan yang dimulai dengan `Log.d` , atau mengubahnya menjadi **Error** untuk melihat pesan yang dimulai dengan `Log.e` , dan seterusnya.

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Pasang Android Studio dan Jalankan "Hello World"

Ketahui selengkapnya

- Dokumentasi Android Studio:
  - Mengenal Android Studio
  - Laman unduhan Android Studio
  - Konfigurasi Varian Pembangunan
  - Buat dan Kelola Perangkat Virtual

    - Tandatangani Aplikasi Anda
    - Ciutkan Kode dan Sumber Daya Anda
- Panduan Android API, bagian "Kembangkan":
    - Pengantar Android
    - Arsitektur Platform
    - Ringkasan UI
    - Versi Platform
    - Mendukung Versi Platform Berbeda
    - Mendukung Beberapa Layar
- Lainnya:
    - Panduan Pengguna Android Studio: Image Asset Studio
    - Wikipedia: Rangkuman Riwayat Versi Android
    - Sintaks Groovy
    - Situs Gradle

1.2: Layout, Tampilan, dan Sumber Daya

Materi:

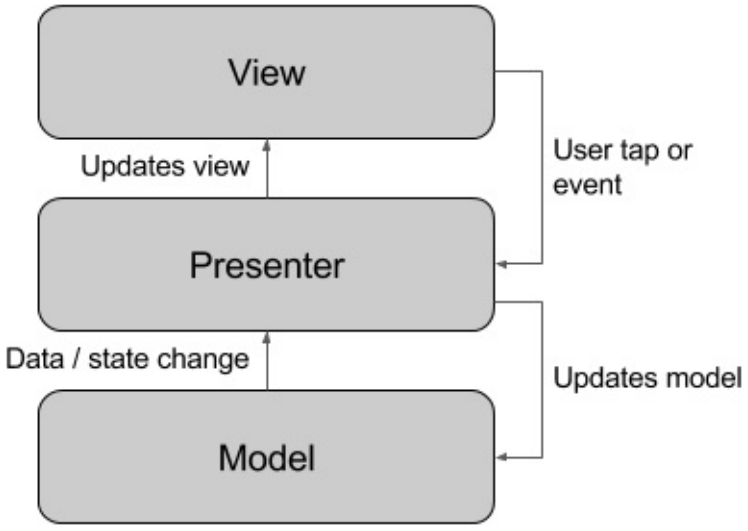
- Pola model-view-presenter
- Tampilan
- File sumber daya
- Merespons klik tampilan
- Praktik terkait
- Ketahui selengkapnya

Bab ini menjelaskan layout antarmuka pengguna di layar dan sumber daya lain yang Anda buat untuk aplikasi, dan kode yang akan digunakan untuk merespons ketukan pengguna pada elemen antarmuka pengguna.

Pola model-view-presenter

Menautkan aktivitas ke sumber daya layout adalah contoh dari bagian pola arsitektur *model-view-presenter* (MVP). Pola MVP adalah cara yang sudah umum digunakan untuk mengelompokkan fungsi aplikasi:

- **Tampilan.** Tampilan adalah elemen antarmuka pengguna yang menampilkan data dan respons terhadap tindakan pengguna. Setiap elemen layar adalah tampilan. Sistem Android menyediakan berbagai jenis tampilan.
- **Presenter.** Presenter menghubungkan tampilan aplikasi ke model. Presenter menyediakan tampilan dengan data sebagaimana ditetapkan oleh model, dan juga menyediakan masukan pengguna dari tampilan kepada model.
- **Model.** Model menetapkan struktur data aplikasi dan kode untuk mengakses dan memanipulasi data. Beberapa aplikasi yang Anda buat dalam pelajaran bisa digunakan bersama model untuk mengakses data. Aplikasi Hello Toast tidak menggunakan model data, namun Anda bisa memikirkan logikanya — menampilkan pesan, dan meningkatkan penghitung ketukan — sebagai model.



Tampilan

UI terdiri dari hierarki objek yang disebut *tampilan*, setiap elemen layar adalah *tampilan*. Kelas View menyatakan blok pembangunan dasar untuk semua komponen UI, dan kelas dasar untuk kelas yang menyediakan komponen UI interaktif seperti tombol, kotak centang, dan bidang entri teks.

Tampilan memiliki lokasi, yang dinyatakan sebagai pasangan koordinat kiri dan atas, dan dua dimensi, yang dinyatakan sebagai lebar dan tinggi. Unit untuk lokasi dan dimensi adalah piksel yang tidak tergantung perangkat (dp).

Sistem Android menyediakan ratusan tampilan yang telah didefinisikan sebelumnya, termasuk yang menampilkan:

- Teks (TextView)
- Bidang untuk memasukkan dan mengedit teks (EditText)
- Pengguna tombol bisa mengetuk (Button) dan komponen interaktif lainnya
- Teks yang bisa digulir (ScrollView) dan item yang bisa digulir (RecyclerView)
- Gambar (ImageView)

Anda bisa mendefinisikan tampilan untuk muncul di layar dan merespons ketukan pengguna. Tampilan juga bisa didefinisikan untuk menerima masukan teks, atau tidak terlihat hingga diperlukan.

Anda bisa menetapkan tampilan di file sumber daya layout XML. Sumber daya layout ditulis dalam XML dan dicantumkan dalam folder **layout** di folder **res** dalam Project: Tampilan Android.

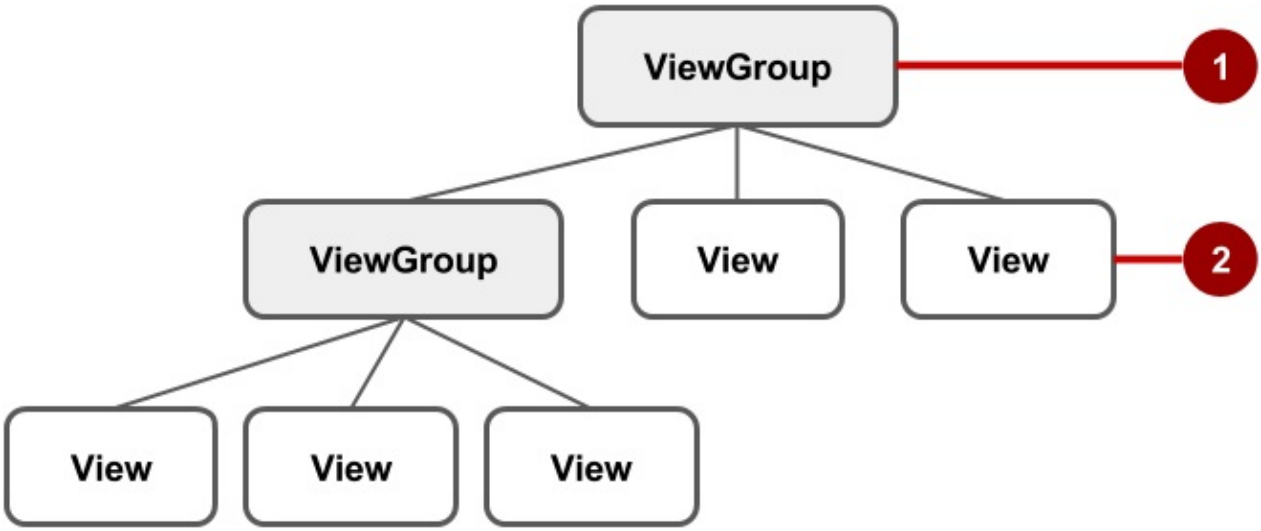
Grup tampilan

Tampilan bisa dikelompokkan bersama di dalam *grup tampilan* (ViewGroup), yang berfungsi sebagai kontainer tampilan. Hubungannya adalah induk-anak, dalam hal ini *induk* adalah grup tampilan, dan *anak* adalah tampilan atau grup tampilan dalam grup. Berikut ini adalah grup tampilan yang umum:

- ScrollView: Grup yang berisi satu tampilan anak lainnya dan memungkinkan pengguliran tampilan anak.
- RecyclerView: Grup yang berisi daftar tampilan atau grup tampilan lainnya dan memungkinkan penggulirannya dengan menambahkan dan membuang tampilan secara dinamis dari layar.

Grup tampilan layout

Tampilan untuk layar dikelola dalam hierarki. Di *akar* hierarki ini adalah ViewGroup yang berisi layout keseluruhan layar. Layar anak grup tampilan bisa menjadi tampilan atau grup tampilan lain seperti yang ditampilkan dalam gambar berikut.



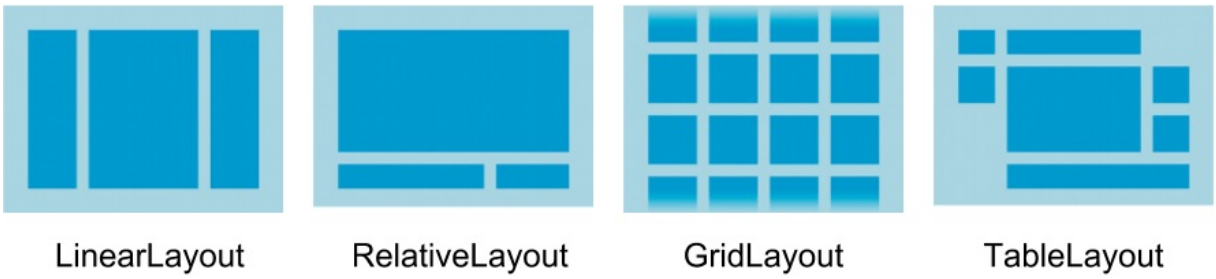
Dalam gambar di atas:

1. Grup tampilan *akar*.
2. Rangkaian tampilan anak dan grup tampilan pertama yang induknya adalah akar.

Beberapa grup tampilan ditandai sebagai *layout* karena grup tampilan tersebut mengelola tampilan anak dalam cara khusus dan umumnya digunakan sebagai grup tampilan akar. Beberapa contoh layout adalah:

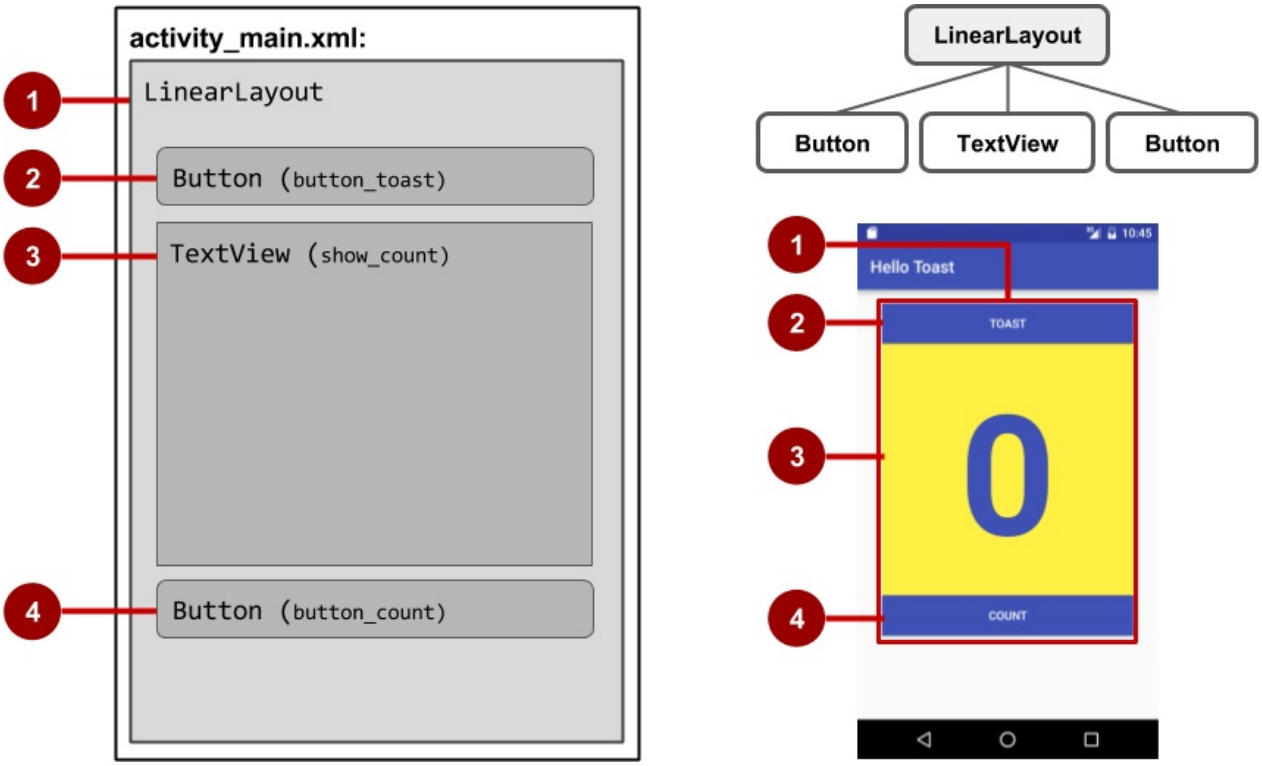
- LinearLayout: Grup tampilan anak yang diposisikan dan disejajarkan secara horizontal atau secara vertikal.
- RelativeLayout: Grup tampilan anak yang setiap tampilannya diposisikan dan disejajarkan relatif terhadap tampilan dalam grup tampilan. Dengan kata lain, posisi tampilan anak bisa dijelaskan dalam hubungan satu sama lain atau dengan grup tampilan induk.
- ConstraintLayout: Grup tampilan anak yang menggunakan titik jangkar, tepi, dan panduan untuk mengontrol cara memosisikan tampilan relatif terhadap elemen lain di layout. ConstraintLayout didesain untuk mempermudah saat menyeret dan melepaskan tampilan di editor layout.

- TableLayout: Grup tampilan anak yang disusun ke dalam baris dan kolom.
- AbsoluteLayout: Grup yang memungkinkan Anda menetapkan lokasi pasti (koordinat x/y) tampilan anaknya. Layout mutlak bersifat kurang fleksibel dan lebih sulit dikelola daripada tipe layout lainnya tanpa pemosisian mutlak.
- FrameLayout: Grup tampilan anak bertumpuk. FrameLayout didesain untuk memblokir area di layar guna menampilkan satu tampilan. Tampilan anak digambar bertumpuk, dengan anak yang baru saja ditambahkan di atas. Ukuran FrameLayout adalah ukuran tampilan anak terbesarnya.
- GridLayout: Grup yang menempatkan layar anaknya dalam kotak persegi panjang yang bisa digulir.



Tip: Ketahui selengkapnya tentang tipe layout yang berbeda di Objek Layout Umum.

Contoh sederhana dari layout dengan tampilan anak adalah aplikasi Hello Toast di salah satu pelajaran awal. Tampilan aplikasi Hello Toast muncul dalam gambar di bawah ini sebagai diagram file layout (activity_main.xml), bersama diagram hierarki (kanan atas) dan tangkapan layar layout yang benar-benar selesai (kanan bawah).



Dalam gambar di atas:

1. Layout akar LinearLayout, yang berisi semua tampilan anak, disetel ke orientasi vertikal.
2. Button (`button_toast`) tampilan anak. Sebagai tampilan anak pertama, muncul di bagian atas di layout linear.
3. TextView (`show_count`) tampilan anak. Sebagai tampilan anak kedua, muncul di bawah tampilan anak pertama di layout linear.
4. Button (`button_count`) tampilan anak. Sebagai tampilan anak ketiga, muncul di bawah tampilan anak kedua di layout linear.

Hierarki tampilan bisa tumbuh menjadi kompleks untuk aplikasi yang menampilkan banyak tampilan di layar. Penting untuk memahami hierarki tampilan, karena akan memengaruhi apakah tampilan terlihat dan apakah digambar secara efisien.

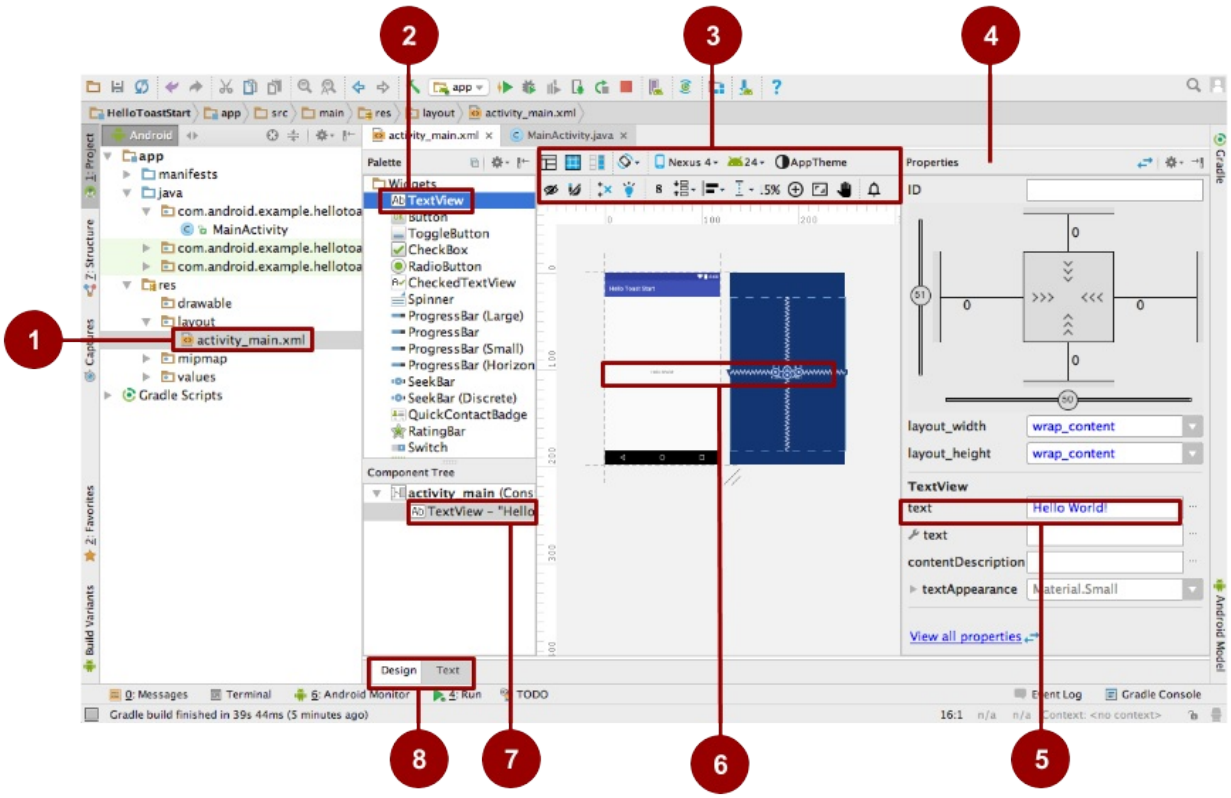
Tip: Anda bisa menjelajahi hierarki tampilan aplikasi menggunakan Hierarchy Viewer. Hierarchy Viewer menampilkan tampilan pohon hierarki dan memungkinkan Anda menganalisis kinerja tampilan di perangkat Android. Masalah kinerja dibahas di bab berikutnya.

Anda mendefinisikan tampilan di editor layout, atau dengan memasukkan kode XML. Editor layout menunjukkan representasi visual kode XML.

Menggunakan editor layout

Gunakan editor layout untuk mengedit file layout. Anda bisa menyeret dan melepas objek tampilan ke dalam panel grafik, serta menyusun, mengubah ukuran, dan menetapkan propertinya. Anda akan segera melihat efek perubahan yang dilakukan.

Untuk menggunakan editor layout, buka file layout XML. Editor layout muncul bersama tab **Design** di bagian bawah yang disorot. (Jika tab **Text** disorot dan Anda melihat kode XML, klik tab **Design**.) Untuk template Empty Activity, layout seperti yang ditampilkan dalam gambar di bawah ini.



Dalam gambar di atas:

1. File **XML layout**. File layout XML, biasanya diberi nama file **activiy_main.xml**. Klik dua kali untuk membuka editor layout.
2. **Palet elemen UI** (tampilan). Panel Palette menyediakan daftar elemen UI dan layout. Tambahkan elemen atau layout ke UI dengan menyeretnya ke panel desain.
3. **Bilah alat desain**. Bilah alat panel desain menyediakan tombol untuk mengonfigurasi penampilan layout dalam panel desain dan untuk mengedit properti layout. Lihat gambar di bawah ini untuk detail.

   Tip: Arahkan kursor ke atas setiap ikon untuk menampilkan keterangan alat yang merangkum fungsinya.

4. **Panel Properties**. Panel Properties menyediakan kontrol properti untuk tampilan yang dipilih.
5. **Kontrol properti**. Kontrol properti sesuai dengan atribut XML. Yang ditampilkan dalam gambar adalah properti `Text` dari TextView yang dipilih, yang disetel ke `Hello World!` .
6. **Panel desain**. Seret tampilan dari panel Palette ke panel desain untuk memosisikannya di layout.
7. **Component Tree**. Panel Component Tree menampilkan hierarki tampilan. Klik tampilan atau grup tampilan dalam panel ini untuk memilihnya. Gambar menampilkan TextView yang dipilih.

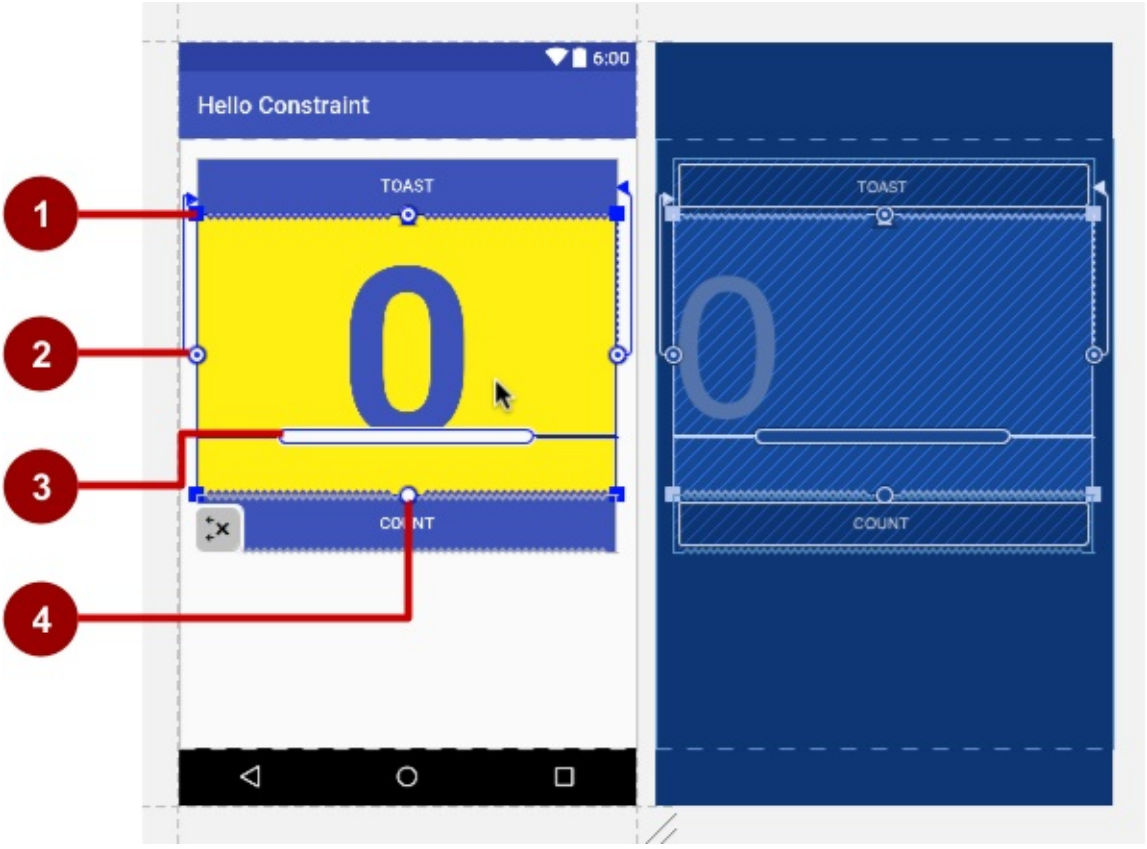
8. Tab **Design** dan **Text**. Klik **Design** untuk melihat editor layout, atau **Text** untuk melihat kode XML.

Bilah alat (bantu) desain editor layout menawarkan deretan tombol yang memungkinkan Anda mengonfigurasi penampilan layout:

Dalam gambar di atas:

1. **Design**, **Blueprint**, dan **Both**: Klik ikon **Design** (ikon pertama) untuk menampilkan pratinjau warna layout. Klik ikon **Blueprint** (ikon tengah) untuk hanya menampilkan ringkasan setiap tampilan. Anda bisa melihat *kedua* tampilan bersebelahan dengan mengeklik ikon ketiga.
2. **Orientasi layar**: Klik untuk memutar perangkat antara lanskap dan potret.
3. **Tipe dan ukuran perangkat**: Pilih tipe perangkat (ponsel/tablet, Android TV, atau Android Wear) dan konfigurasi layar (ukuran dan kepadatan).
4. **Versi API**: Pilih versi Android yang digunakan untuk pratinjau layout.
5. **Tema aplikasi**: Pilih tema UI yang akan diterapkan pada pratinjau.
6. **Bahasa**: Pilih bahasa untuk menampilkan string UI Anda. Daftar ini hanya menampilkan bahasa yang tersedia dalam sumber daya string.
7. **Varian Layout**: Alihkan ke salah satu layout alternatif untuk file ini, atau buat yang baru.

Editor layout menawarkan lebih banyak fitur di tab **Design** bila Anda menggunakan ConstraintLayout, termasuk tuas untuk mendefinisikan pembatas. *Pembatas* adalah koneksi atau penyejajaran ke tampilan lainnya, ke layout induk, atau ke panduan yang tidak terlihat. Setiap pembatas muncul sebagai garis yang membentang dari tuas melingkar. Setiap tampilan memiliki tuas pembatas melingkar di bagian tengah setiap sisi. Setelah memilih tampilan di panel Component Tree atau mengekliknya di layout, tampilan juga menampilkan tuas pengubah ukuran pada setiap sudut.



Dalam gambar di atas:

1. **Tuas pengubah ukuran**.

2. **Garis pembatas dan tuas**. Dalam gambar, pembatas menyejajarkan sisi kiri tampilan ke sisi kiri tombol.
3. **Tuas patokan**. Tuas patokan menyejajarkan patokan teks tampilan ke patokan tampilan lainnya.
4. **Tuas pembatas** tanpa garis pembatas.

Menggunakan XML

Terkadang lebih cepat dan lebih mudah mengedit kode XML secara langsung, terutama saat menyalin dan menempelkan kode untuk tampilan serupa.

Untuk menampilkan dan mengedit kode XML, buka file layout XML. Editor layout muncul bersama tab **Design** di bagian bawah yang disorot. Klik tab **Text** untuk melihat kode XML. Yang berikut ini menampilkan cuplikan kode XML untuk LinearLayout dengan Button dan TextView:

```
<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"
    ... >

    <Button
        android:id="@+id/button_toast"
        android:layout_width="@dimen/my_view_width"
        android:layout_height="wrap_content"
        ... />

 <TextView
        android:id="@+id/show_count"
        android:layout_width="@dimen/my_view_width"
        android:layout_height="@dimen/counter_height"
        ... />
    ...
</LinearLayout>
```

Atribut XML (properti tampilan)

Tampilan memiliki *properti* yang mendefinisikan tempat munculnya tampilan di layar, ukurannya, dan bagaimana kaitan tampilan dengan tampilan lainnya, dan cara tampilan merespons masukan pengguna. Saat mendefinisikan tampilan di XML, properti dirujuk sebagai *atribut*.

Misalnya, dalam keterangan XML TextView berikut, `android:id`, `android:layout_width`, `android:layout_height`, `android:background`, merupakan atribut XML yang diterjemahkan secara otomatis menjadi properti TextView:

```
<TextView
       android:id="@+id/show_count"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:background="@color/myBackgroundColor"
       android:textStyle="bold"
       android:text="@string/count_initial_value"
/>
```

Atribut biasanya berbentuk seperti ini:

```
android:attribute_name="value"
```

attribute_name adalah nama atribut. *value* adalah string dengan nilai untuk atribut. Misalnya:

```
android:textStyle="bold"
```

Jika *value* adalah sumber daya, seperti warna, simbol `@` menetapkan jenis sumber dayanya. Misalnya:

```
android:background="@color/myBackgroundColor"
```

Atribut latar belakang disetel ke sumber daya warna yang diidentifikasi sebagai `myBackgroundColor`, yang dideklarasikan sebagai `#FFF043`. Sumber daya warna dijelaskan dalam "Atribut terkait gaya" dalam bab ini.

Setiap tampilan dan grup tampilan mendukung berbagai atribut XML-nya sendiri. Beberapa atribut telah ditetapkan untuk tampilan (misalnya, TextView mendukung atribut `textSize`), namun atribut-atribut ini juga diwarisi oleh tampilan apa pun yang mungkin memperluas kelas TextView. Sebagian bersifat umum untuk semua tampilan, karena diwarisi dari kelas View akar (seperti atribut `android:id`). Untuk keterangan atribut khusus, lihat bagian ringkasan dokumentasi kelas View.

Mengidentifikasi tampilan

Untuk mengidentifikasi tampilan secara unik dan merujuknya dari kode, Anda harus memberikan ID. Atribut `android:id` memungkinkan Anda menetapkan `id` yang unik— yakni identifier sumber daya untuk tampilan.

Misalnya:

```
android:id="@+id/button_count"
```

Bagian `"@+id/button_count"` dari atribut di atas akan membuat `id` baru yang disebut `button_count` untuk tampilan. Anda menggunakan simbol plus (`+`) untuk menunjukkan bahwa Anda sedang membuat baru `id`.

Merujuk tampilan

Untuk merujuk ke identifier sumber daya yang ada, hilangkan simbol plus (`+`). Misalnya, untuk merujuk tampilan berdasarkan `id` -nya dalam atribut *lain*, misalnya `android:layout_toLeftOf` (yang dijelaskan di bagian berikutnya) untuk mengontrol posisi tampilan, Anda perlu menggunakan:

```
android:layout_toLeftOf="@id/show_count"
```

Dalam atribut di atas, `"@id/show_count"` merujuk ke tampilan dengan identifier sumber daya `show_count`. Atribut memosisikan tampilan agar "ke kiri dari" tampilan `show_count`.

Memosisikan tampilan

Beberapa atribut pemosisian yang terkait layout diperlukan untuk tampilan, dan secara otomatis muncul bila Anda menambahkan tampilan ke layout XML, siap untuk Anda tambahkan nilainya.

Pemosisian LinearLayout

Misalnya, LinearLayout diperlukan untuk menyetel atribut ini:

- android:layout_width
- android:layout_height
- android:orientation

Atribut `android:layout_width` dan `android:layout_height` bisa menggunakan salah satu dari tiga nilai ini:

- `match_parent` akan meluaskan tampilan untuk mengisi induknya dengan lebar dan tinggi. Bila LinearLayout adalah tampilan akar, ia akan meluaskan ke ukuran layar perangkat. Untuk tampilan dalam grup tampilan akar, ia akan meluaskan ke ukuran grup tampilan induk.
- `wrap_content` akan menciutkan dimensi tampilan yang cukup besar untuk menampung isinya. (Jika tidak ada isinya, tampilan menjadi tidak terlihat.)
- Gunakan `dp` (piksel yang tidak tergantung perangkat) berjumlah tetap untuk menetapkan ukuran tetap, yang disesuaikan untuk ukuran layar perangkat. Misalnya, `16dp` berarti 16 piksel yang tidak tergantung perangkat. Piksel

yang tidak tergantung perangkat dan dimensi lain dijelaskan di "Dimensi" dalam bab ini.

Atribut `android:orientation` bisa berupa:

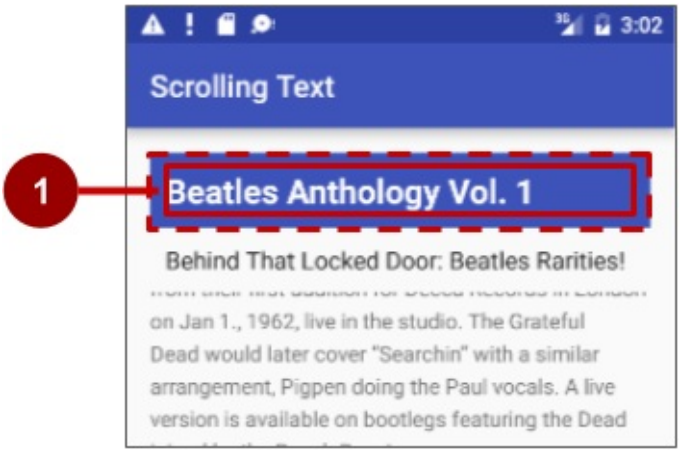
- `horizontal:` Tampilan disusun dari kiri ke kanan.
- `vertical:` Tampilan disusun dari atas ke bawah.

Atribut terkait layout lainnya antara lain:

- `Android:layout_gravity` : Atribut ini digunakan bersama tampilan untuk mengontrol tempat menyusun tampilan dalam grup tampilan induknya. Misalnya, atribut berikut memusatkan tampilan secara horizontal di layar:

```
android:layout_gravity="center_horizontal"
```

- Pengisi adalah ruang, yang diukur dalam piksel yang tidak tergantung perangkat, antara tepi tampilan dan isi tampilan, seperti yang ditampilkan dalam gambar di bawah ini.



Dalam gambar di atas:

1. Pengisi adalah ruang antara tepi tampilan (garis putus-putus) dan isi tampilan (garis utuh). Pengisi tidak sama dengan margin, yang merupakan ruang dari tepi tampilan ke induknya.

Ukuran tampilan menyertakan pengisinya. Berikut ini adalah atribut pengisi yang umum digunakan:

- `Android:padding` : Menyetel pengisi keempat tepi.
- `android:paddingTop` : Menyetel pengisi tepi atas.
- `android:paddingBottom` : Menyetel pengisi tepi bawah.
- `android:paddingLeft` : Menyetel pengisi tepi kiri.
- `android:paddingRight` : Menyetel pengisi tepi kanan.
- `android:paddingStart` : Menyetel pengisi awal tampilan; digunakan di tempat di atas, terutama bersama tampilan yang panjang dan sempit.
- `android:paddingEnd` : Menyetel pengisi, dalam piksel, tepi ujung; yang digunakan bersama `android:paddingStart` .

Tip: Untuk melihat semua atribut XML untuk LinearLayout, lihat bagian Rangkuman referensi LinearLayout di Panduan Developer. Layout akar lainnya, seperti RelativeLayout dan AbsoluteLayout, mencantumkan atribut XML-nya di bagian Rangkuman.

Pemosisian RelativeLayout

Grup tampilan berguna lainnya untuk layout adalah RelativeLayout, yang bisa Anda gunakan untuk memosisikan tampilan anak yang berhubungan satu sama lain atau dengan induk. Atribut yang bisa Anda gunakan bersama RelativeLayout antara lain berikut ini:

- android:layout_toLeftOf: Memosisikan tepi kanan tampilan ini ke kiri tampilan lainnya (yang diidentifikasi melalui `ID` -nya).
- android:layout_toRightOf: Memosisikan tepi kiri tampilan ini ke kanan tampilan lainnya (yang diidentifikasi melalui `ID` -

nya).
- android:layout_centerHorizontal: Memusatkan tampilan ini secara horizontal dalam induknya.
- android:layout_centerVertical: Memusatkan tampilan ini secara vertikal dalam induknya.
- android:layout_alignParentTop: Memosisikan tepi atas tampilan ini agar cocok dengan tepi atas induknya.
- android:layout_alignParentBottom: Memosisikan tepi bawah tampilan ini agar cocok dengan tepi bawah induknya.

Untuk daftar lengkap atribut tampilan dalam RelativeLayout, lihat RelativeLayout.LayoutParams.

Atribut terkait gaya

Anda menetapkan atribut gaya untuk menyesuaikan penampilan tampilan. Tampilan yang *tidak* memiliki atribut gaya, misalnya `android:textColor`, `android:textSize`, dan `android:background`, menggunakan gaya yang didefinisikan di tema aplikasi.

Berikut ini adalah atribut terkait gaya yang digunakan dalam contoh layout XML di bagian sebelumnya:

- `Android:background`: Menetapkan warna atau sumber daya dapat digambar untuk digunakan sebagai latar belakang.
- `android:text`: Menetapkan teks untuk ditampilkan di tampilan.
- `android:textColor`: Menetapkan warna teks.
- `android:textSize`: Menetapkan ukuran teks.
- `android:textStyle`: Menetapkan gaya teks, misalnya `bold`.

File sumber daya

File sumber daya adalah cara memisahkan nilai statis dari kode sehingga Anda tidak harus mengubah kode itu sendiri untuk mengubah nilai. Anda bisa menyimpan semua string, layout, dimensi, warna, gaya, dan teks menu secara terpisah di file sumber daya.

File sumber daya disimpan dalam folder yang berada di folder **res**, termasuk:

- **drawable**: Untuk gambar dan ikon
- **layout**: Untuk file sumber daya layout
- **menu**: Untuk item menu
- **mipmap**: Untuk kumpulan ikon aplikasi yang sudah dihitung dan dioptimalkan yang digunakan oleh Peluncur
- **values**: Untuk warna, dimensi, string, dan gaya (atribut tema)

Sintaks untuk merujuk sumber daya di layout XML adalah seperti berikut:

```
@package_name:resource_type/resource_name
```

- *package_name* adalah nama paket tempat sumber daya berada. Ini tidak diperlukan saat merujuk sumber daya dari paket yang sama — yakni, yang disimpan di folder **res** proyek Anda.
- *resource_type* adalah `R` subkelas untuk tipe sumber daya. Lihat Tipe Sumber Daya untuk informasi selengkapnya tentang setiap tipe sumber daya dan cara merujuknya.
- *resource_name* adalah nama file sumber daya tanpa ekstensi, atau nilai atribut `android:name` di elemen XML.

Misalnya, pernyataan layout XML berikut menyetel atribut `android:text` ke sumber daya `string`:

```
android:text="@string/button_label_toast"
```

- *resource_type* adalah `string`.
- resource_name adalah `button_label_toast.`
- *package_name* tidak diperlukan karena sumber daya disimpan di proyek (dalam file strings.xml).

Contoh lainnya: pernyataan layout XML ini menyetel atribut `android:background` ke sumber daya `color`, dan karena sumber daya didefinisikan di proyek (dalam file colors.xml), *package_name* tidak ditetapkan:

```
android:background="@color/colorPrimary"
```

Dalam contoh berikut, pernyataan layout XML menyetel atribut `android:textColor` ke sumber daya `color` . Akan tetapi, sumber daya tidak didefinisikan dalam proyek, melainkan disediakan oleh Android, sehingga *package_name* `android` juga harus ditetapkan, yang diikuti dengan titik dua:

```
android:textColor="@android:color/white"
```

Tip: Untuk informasi selengkapnya tentang mengakses sumber daya dari kode, lihat Mengakses Sumber Daya. Untuk konstanta warna Android, lihat sumber daya R.color standar Android.

File sumber daya nilai

Menyimpan nilai sebagai string dan warna dalam file sumber daya terpisah mempermudah pengelolaannya, terutama jika Anda menggunakannya lebih dari sekali di layout.

Misalnya, penting sekali menyimpan string dalam file sumber daya terpisah untuk menerjemahkan dan melokalkan aplikasi, sehingga Anda bisa membuat file sumber daya string untuk setiap bahasa tanpa mengubah kode. File sumber daya untuk gambar, warna, dimensi, dan atribut lainnya berguna untuk mengembangkan aplikasi bagi orientasi dan ukuran layar perangkat yang berbeda.

String

Sumber daya string berada dalam file **strings.xml** dalam folder **values** di dalam folder **res** saat menggunakan Project: Tampilan Android. Anda bisa mengedit file ini secara langsung dengan membukanya:

```
<resources>
    <string name="app_name">Hello Toast</string>
    <string name="button_label_count">Count</string>
    <string name="button_label_toast">Toast</string>
    <string name="count_initial_value">0</string>
</resources>
```

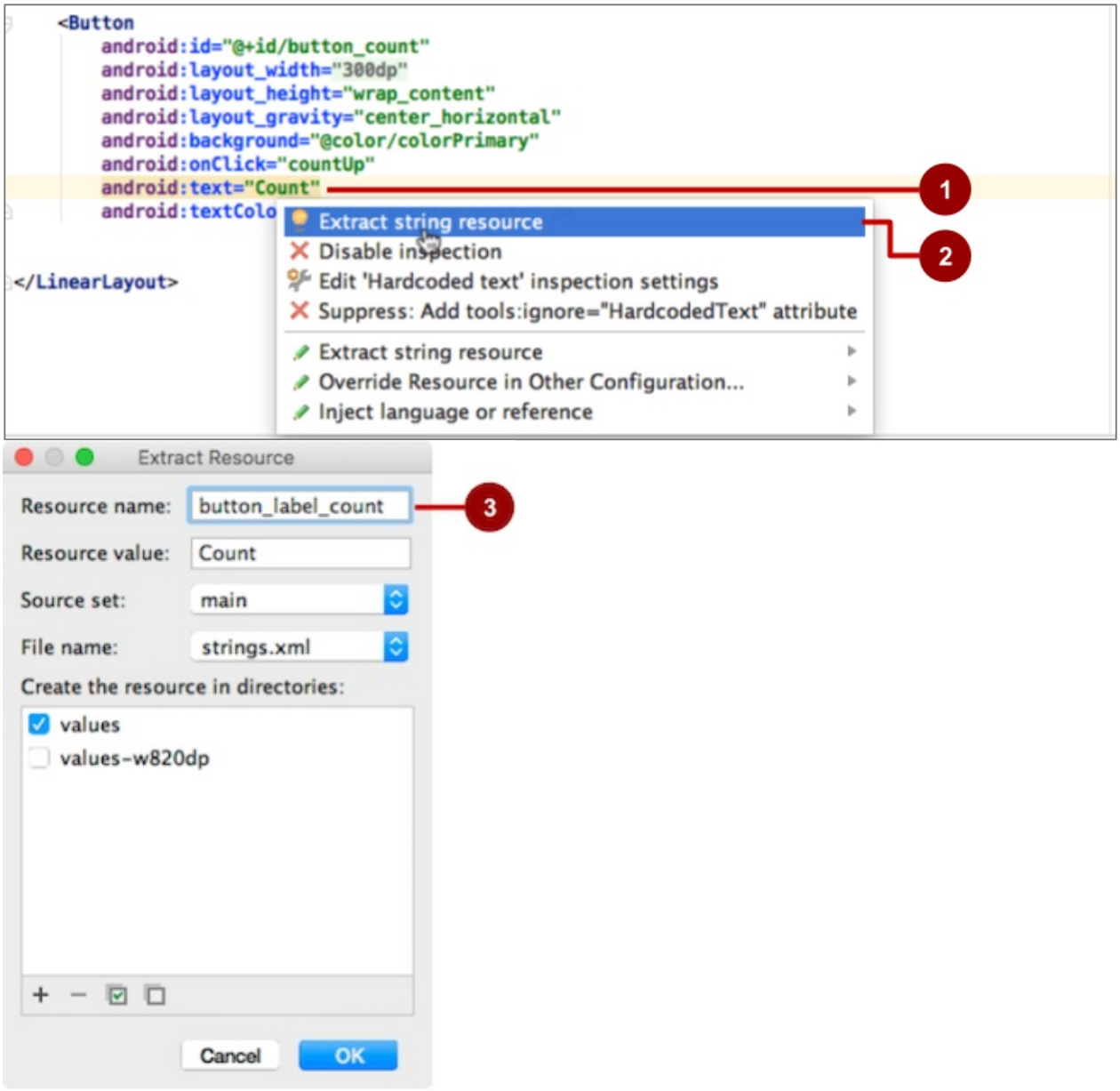
Di sini, `name` (misalnya, `button_label_count`) adalah nama sumber daya yang Anda gunakan di kode XML, seperti dalam atribut berikut:

```
android:text="@string/button_label_count"
```

Nilai string `name` ini adalah kata (`Count`) yang dimasukkan dalam tag `<string></string>` (Anda jangan menggunakan tanda kutip kecuali jika tanda kutip harus menjadi bagian dari nilai string.)

Mengekstrak string ke sumber daya

Anda juga harus *mengekstrak* string hard-code dalam file layout XML ke sumber daya string. Untuk mengekstrak string hard-code dalam layout XML, ikuti langkah-langkah ini (lihat gambar):



1. Klik string hard-code, dan tekan Alt-Enter di Windows, atau Option-Return pada Mac OS X.
2. Pilih **Extract string resource**.
3. Edit nama Sumber Daya untuk nilai string.

Selanjutnya Anda bisa menggunakan nama sumber daya di kode XML. Gunakan ekspresi `"@string/resource_name"` (termasuk tanda kutip) untuk merujuk sumber daya string:

```
android:text="@string/button_label_count"
```

Warna

Sumber daya warna berada di file **colors.xml** dalam folder **values** di dalam folder **res** saat menggunakan Project: Tampilan Android. Anda bisa mengedit file ini secara langsung:

```
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name="myBackgroundColor">#FFF043</color>
</resources>
```

Di sini, `name` (misalnya, `colorPrimary`) adalah nama sumber daya yang Anda gunakan dalam kode XML:

```
android:textColor="@color/colorPrimary"
```

Nilai warna `name` ini adalah nilai warna heksadesimal (`#3F51B5`) yang dimasukkan dalam tag `<color></color>` . Nilai heksadesimal menetapkan nilai merah, hijau, dan biru (RGB). Nilai selalu diawali dengan karakter pound (`#`), yang diikuti dengan informasi Alpha-Red-Green-Blue. Misalnya, nilai heksadesimal hitam adalah #000000, sedangkan nilai heksadesimal untuk varian biru langit adalah #559fe3. Nilai warna dasar dicantumkan dalam dokumentasi kelas Color.

Warna `colorPrimary` adalah salah satu warna dasar yang telah didefinisikan sebelumnya dan digunakan untuk bilah aplikasi. Di aplikasi produksi, Anda bisa, misalnya, menyesuaikannya agar sesuai dengan merek. Menggunakan warna dasar untuk elemen UI lainnya akan membuat UI seragam.

Tip: Untuk spesifikasi desain material warna Android, lihat Gaya dan Menggunakan Tema Material. Untuk nilai heksadesimal warna umum, lihat Kode Warna Heksadesimal. Untuk konstanta warna Android, lihat sumber daya R.color standar Android.

Anda bisa melihat blok kecil pilihan warna di margin kiri di sebelah deklarasi sumber daya warna di **colors.xml**, juga di margin kiri di sebelah atribut yang menggunakan nama sumber daya di file XML layout.

Tip: Untuk melihat warna di munculan, aktifkan fitur dokumentasi Munculan otomatis. Pilih **Android Studio > Preferences > Editor > General > Code Completion**, dan centang opsi "Autopopup documentation in (ms)". Selanjutnya Anda bisa mengarahkan kursor ke atas nama sumber daya warna untuk melihat warnanya.

Dimensi

Dimensi harus dipisahkan dari kode untuk mempermudah pengelolaannya, terutama jika Anda perlu menyesuaikan layout untuk resolusi perangkat yang berbeda. Selain itu juga memudahkan pengukuran yang konsisten untuk tampilan, dan untuk mengubah ukuran beberapa tampilan dengan mengubah satu sumber daya dimensi.

Sumber daya dimensi berada di file **dimens.xml** dalam folder **values** di dalam folder **res** saat menggunakan Project: Tampilan Android. **dimens.xml** yang ditampilkan di tampilan bisa menjadi folder yang memiliki lebih dari satu file **dimens.xml** untuk resolusi perangkat yang berbeda. Misalnya, aplikasi yang dibuat dari template Empty Activity menyediakan file **dimens.xml** kedua untuk 820 dp.

Anda bisa mengedit file ini secara langsung dengan membukanya:

```
<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
    <dimen name="my_view_width">300dp</dimen>
    <dimen name="count_text_size">200sp</dimen>
    <dimen name="counter_height">300dp</dimen>
</resources>
```

Di sini, `name` (misalnya, `activity_horizontal_margin`) adalah nama sumber daya yang Anda gunakan di kode XML:

```
android:paddingLeft="@dimen/activity_horizontal_margin"
```

Nilai `name` ini adalah pengukuran (`16dp`) yang dimasukkan dalam tag `<dimen></dimen>`.

Anda bisa mengekstrak dimensi dengan cara yang sama seperti string:

1. Klik dimensi hard-code, dan tekan Alt-Enter di Windows, atau tekan Option-Return di Mac OS X.
2. Pilih **Extract dimension resource**.
3. Edit nama Sumber Daya untuk nilai dimensi.

Piksel yang tidak tergantung perangkat (`dp`) tidak tergantung resolusi layar. Misalnya, `10px` (10 piksel tetap) akan terlihat sedikit lebih kecil pada layar yang beresolusi lebih tinggi, namun Android akan menskalakan 1 `0dp` (10 piksel yang tidak tergantung perangkat) untuk melihat langsung di layar resolusi berbeda. Ukuran teks juga bisa disetel untuk terlihat langsung di layar resolusi berbeda dengan menggunakan ukuran *piksel yang diskalakan* (`sp`).

Tip: Untuk informasi selengkapnya tentang unit `dp` dan `sp`, lihat Mendukung Kepadatan Berbeda.

Gaya

Gaya adalah sumber daya yang menetapkan atribut umum seperti tinggi, pengisi, warna font, ukuran font, dan warna latar belakang. Gaya ditujukan untuk atribut yang memodifikasi rupa tampilan.

Gaya didefinisikan di file **styles.xml** dalam folder **values** di dalam folder **res** saat menggunakan Project: Tampilan Android. Anda bisa mengedit file ini secara langsung. Gaya dibahas dalam bab berikutnya, bersama Spesifikasi Desain Material.

File sumber daya lainnya

Android Studio mendefinisikan sumber daya lainnya yang dibahas dalam bab lain:

- **Gambar dan ikon**. Folder **drawable** menyediakan sumber daya ikon dan gambar. Jika aplikasi Anda tidak memiliki folder drawable, Anda bisa membuatnya di dalam folder res secara manual. Untuk informasi selengkapnya tentang sumber daya dapat digambar, lihat Sumber Daya Dapat Digambar di bagian Sumber Daya Aplikasi dari Panduan Developer Android.
- **Ikon yang dioptimalkan**. Folder **mipmap** umumnya berisi kumpulan ikon aplikasi yang sudah dihitung dan dioptimalkan, yang digunakan oleh Peluncur. Luaskan folder untuk melihat apakah versi ikon disimpan sebagai sumber daya untuk kepadatan layar yang berbeda.
- **Menu**. Anda bisa menggunakan file sumber daya XML untuk mendefinisikan item menu dan menyimpannya di proyek dalam folder **menu**. Menu dijelaskan dalam bab berikutnya.

Merespons klik tampilan

Kejadian klik terjadi bila pengguna mengetuk atau mengeklik tampilan yang bisa diklik seperti Button, ImageButton, ImageView (mengetuk atau mengeklik gambar), atau FloatingActionButton.

Pola model-view-presenter berguna untuk memahami cara merespons klik tampilan. Bila kejadian terjadi bersama tampilan, kode *presenter* akan melakukan aksi yang memengaruhi kode *model*. Agar pola ini berfungsi, Anda harus:

- Menulis metode Java yang melakukan aksi khusus, yang ditentukan oleh logika kode *model* — yakni tindakan yang bergantung pada hal yang Anda inginkan untuk dilakukan aplikasi bila kejadian ini terjadi. Ini biasanya disebut sebagai *penangan kejadian*.
- Mengaitkan metode penangan kejadian ini ke *tampilan*, sehingga metode berjalan bila kejadian terjadi.

Atribut onClick

Android Studio menyediakan pintasan untuk mempersiapkan tampilan yang bisa diklik, dan untuk mengaitkan penangan kejadian dengan tampilan: gunakan atribut `android:onClick` bersama elemen tampilan yang bisa diklik di layout XML.

Misalnya, ekspresi XML berikut di file layout untuk serangkaian Button. `showToast()` sebagai penangan kejadian:

```
android:onClick="showToast"
```

Bila tombol `b` diketuk, atribut `android:onClick` -nya akan memanggil metode `showToast()` .

Tulis penangan kejadian, misalnya `showToast()` yang direferensikan dalam kode XML di atas, untuk memanggil metode lain yang mengimplementasikan logika *model* aplikasi:

```
public void showToast(View view) {
        // Do something in response to the button click.
}
```

Agar dapat digunakan bersama atribut `android:onClick` , metode `showToast()` harus berupa `public` , mengembalikan `void` , dan memerlukan parameter `view` agar dapat mengetahui tampilan mana yang memanggil metode.

Android Studio menyediakan pintasan untuk membuat *stub* penangan kejadian (placeholder untuk metode yang bisa Anda isi nanti) di kode Java untuk aktivitas yang terkait dengan layout XML. Ikuti langkah-langkah ini:

1. Di dalam file layout XML (misalnya activity_main.xml), klik nama metode dalam pernyataan atribut `android:onClick` .
2. Tekan Alt-Enter di Windows atau Option-Return di Mac OS X, dan pilih **Create onClick event handler**.
3. Pilih aktivitas yang terkait dengan file layout (misalnya **MainActivity**) dan klik **OK**. Ini akan membuat stub metode placeholder di MainActivity.java.

Memperbarui tampilan

Untuk memperbarui materi tampilan, misalnya mengganti teks di TextView, terlebih dahulu kode Anda harus membuat instance objek dari tampilan. Selanjutnya kode Anda bisa memperbarui objek, dengan demikian memperbarui tampilan.

Untuk merujuk tampilan dalam kode Anda, gunakan metode findViewById() kelas View, yang akan mencari tampilan berdasarkan `id` sumber daya. Misalnya, pernyataan berikut menyetel `mShowCount` menjadi TextView dengan `show_count` ID sumber daya:

```
mShowCount = (TextView) findViewById(R.id.show_count);
```

Dari poin ini, kode Anda bisa menggunakan `mShowCount` untuk menyatakan TextView, sehingga bila Anda memperbarui `mShowCount` , tampilan akan diperbarui.

Misalnya, ketika tombol berikut dengan atribut `android:onClick` diketuk, onClick akan memanggil metode `countUp()` :

```
android:onClick="countUp"
```

Anda bisa mengimplementasikan `countUp()` untuk menaikkan hitungan, mengubah hitungan ke string, dan menyetel string sebagai teks untuk objek `mShowCount` :

```
public void countUp(View view) {
        mCount++;
        if (mShowCount != null)
            mShowCount.setText(Integer.toString(mCount));
}
```

Karena Anda sudah mengaitkan `mShowCount` dengan TextView untuk menampilkan hitungan, metode `mShowCount.setText()` memperbarui tampilan teks di layar.

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Buat UI Interaktif Pertama Anda
- Menggunakan Layout

Ketahui selengkapnya

- Dokumentasi Android Studio:
  - Panduan Pengguna Android Studio
- Panduan Android API, bagian "Kembangkan":
  - Ringkasan UI
  - Objek Layout Umum
  - Definisi kelas Color
  - Sumber daya R.color standar Android
  - Mendukung Kepadatan Berbeda
- Desain Material:
  - Gaya
  - Menggunakan Tema Material
- Lainnya:
  - Panduan Pengguna Android Studio: Image Asset Studio
  - Pola arsitektur Model-View-Presenter (MVP)
  - Hierarchy Viewer untuk memvisualisasikan hierarki tampilan
  - Kode Warna Heksa Warna

1.3: Tampilan Bergulir dan Teks

Materi:

- TextView
- Tampilan bergulir
- Praktik terkait
- Ketahui selengkapnya

Bab ini menjelaskan salah satu tampilan yang paling sering digunakan di aplikasi: TextView, yang menampilkan materi tekstual pada layar. TextView bisa digunakan untuk menampilkan pesan, respons dari database, atau bahkan keseluruhan artikel bergaya majalah yang bisa digulir pengguna. Bab ini juga menampilkan cara membuat tampilan bergulir untuk teks dan elemen lainnya.

TextView

Satu tampilan yang mungkin sering Anda gunakan adalah kelas TextView, yang merupakan subkelas dari kelas View yang menampilkan teks pada layar. Anda bisa menggunakan TextView untuk tampilan berukuran apa pun, mulai dari karakter atau kata tunggal hingga teks selayar penuh. Anda bisa menambahkan `id` sumber daya ke TextView dan mengontrol cara teks muncul dengan menggunakan atribut di file layout XML.

Anda bisa merujuk tampilan TextView di kode Java dengan menggunakan `id` sumber dayanya, dan memperbarui teks dari kode Anda. Jika Anda ingin memungkinkan pengguna mengedit teks, gunakan EditText, subkelas TextView yang memungkinkan pengeditan dan masukan teks. Anda mempelajari semua tentang EditText di bab lainnya.

Atribut TextView

Anda bisa menggunakan atribut XML untuk mengontrol:

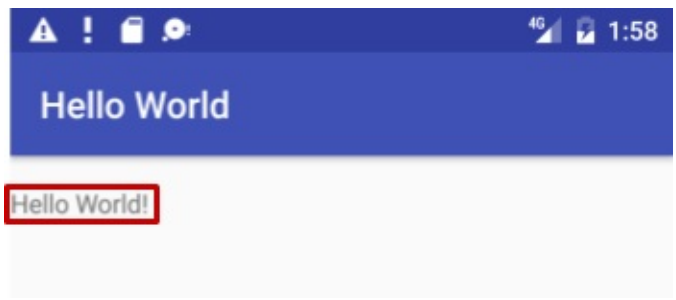
- Tempat memosisikan TextView di layout (seperti tampilan lainnya)
- Cara munculnya tampilan itu sendiri, misalnya dengan warna latar belakang
- Seperti apa teks terlihat dalam tampilan, misalnya teks awal dengan gaya, ukuran, dan warnanya

Misalnya, untuk menyetel lebar, tinggi, dan posisi dalam LinearLayout:

```
<TextView
   ...
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   … />
```

Untuk menyetel nilai teks awal tampilan, gunakan atribut `android:text` :

```
android:text="Hello World!"
```



Anda bisa mengekstrak string teks ke dalam sumber daya string (mungkin disebut `hello_world`) yang lebih mudah dikelola untuk versi aplikasi multibahasa, atau jika Anda perlu mengubah string di lain waktu. Setelah mengekstrak string, gunakan nama sumber daya string dengan `@string/` untuk menetapkan teks:

```
android:text="@string/hello_world"
```

Atribut yang paling sering digunakan bersama TextView adalah sebagai berikut:

- android:text: Menyetel teks yang akan ditampilkan.
- android:textColor: Menyetel warna teks. Anda bisa menyetel atribut ke suatu nilai warna, sumber daya yang telah didefinisikan sebelumnya, atau tema. Sumber daya dan tema warna dijelaskan dalam bab lainnya.
- android:textAppearance: Penampilan teks, termasuk warna, jenis huruf, gaya, dan ukurannya. Anda menyetel atribut ini untuk sumber daya gaya yang telah didefinisikan sebelumnya atau tema yang sudah mendefinisikan nilai-nilai ini.
- android:textSize: Menyetel ukuran teks (jika belum disetel melalui `android:textAppearance`). Gunakan ukuran `sp` (piksel yang diskalakan) seperti `20sp` atau `14.5sp`, atau menyetel atribut untuk sumber daya atau tema yang telah didefinisikan sebelumnya.
- android:textSize: Menyetel gaya teks (jika belum disetel melalui `android:textAppearance`). Gunakan `normal`, `bold`, `italic`, atau `bold|italic`.
- Android:typeface: Menyetel jenis huruf teks (jika belum disetel melalui `android:textAppearance`). Gunakan `normal`, `sans`, `serif`, atau `monospace`.
- android:lineSpacingExtra: Menyetel spasi tambahan antar baris teks. Gunakan ukuran `sp` (piksel yang diskalakan) atau dp (piksel yang tidak tergantung perangkat), atau setel atribut ke sumber daya atau tema yang telah didefinisikan sebelumnya.
- android:autoLink: Mengontrol apakah tautan seperti URL dan alamat email secara otomatis ditemukan dan diubah ke tautan yang dapat diklik (dapat disentuh). Gunakan salah satu dari berikut ini:
  - `none`: Tidak ada pola yang cocok (default).
  - `web`: Cocok dengan URL web.
  - `email`: Cocok dengan alamat email.
  - `phone`: Cocok dengan nomor telepon.
  - `map`: Cocok dengan alamat peta.
  - `all`: Cocok dengan semua pola (setara dengan web|email|telepon|peta).

  Misalnya, untuk menyetel atribut agar cocok dengan URL web, gunakan ini:

  ```
  android:autoLink="web"
  ```

Menggunakan tag yang disematkan dalam teks

Dalam aplikasi yang mengakses artikel majalah atau koran, artikel yang muncul mungkin berasal dari sumber online atau mungkin disimpan sebelumnya dalam database di perangkat. Anda juga bisa membuat teks sebagai satu string panjang di sumber daya strings.xml.

Dalam kasus lain, teks bisa berisi tag HTML yang disematkan atau kode pemformatan teks lainnya. Agar tampil dengan benar, tampilan teks harus diformat dengan aturan berikut:

- Jika ada tanda apostrof (') di teks, Anda harus *meloloskannya* dengan mengawalinya dengan backslash (**\'**). Jika menggunakan tanda kutip ganda dalam teks, Anda juga harus meloloskannya (**\"**). Anda juga harus meloloskan karakter non-ASCII lainnya. Lihat bagian "Memformat dan Menata Gaya" pada Sumber Daya String untuk detail selengkapnya.
- TextView mengabaikan semua tag HTML kecuali yang berikut ini:
  - Penggunaan tag HTML dan **** di sekitar kata yang harus ditulis tebal.
  - Penggunaan tag HTML ***dan*** di sekitar kata yang harus ditulis miring. Akan tetapi perhatikan, jika menggunakan apostrof keriting dalam frasa miring, Anda harus menggantinya dengan apostrof tegak.
  - Anda bisa mengombinasikan tebal dan miring dengan mengombinasikan tag, seperti dalam **...** kata...**</i>**.

Untuk membuat string teks panjang dalam file **strings.xml**, masukkan keseluruhan teks dalam `<string name="your_string_name"></string>` di file strings.xml (*your_string_name* adalah nama yang Anda berikan untuk sumber daya string, misalnya `article_text`).

Baris teks dalam file strings.xml tidak digulung ke baris berikutnya — melainkan memanjang melebihi margin kanan. Ini adalah perilaku yang benar. Setiap baris teks baru yang dimulai dari margin kiri menyatakan paragraf lengkap.

Masukkan **\n** untuk menyatakan akhir baris, dan **\n** lainnya untuk menyatakan baris kosong. Jika Anda tidak menambahkan karakter penutup baris, paragraf akan saling bertabrakan saat ditampilkan di layar.

Tip: Jika ingin melihat teks yang dibungkus dalam strings.xml, Anda bisa menekan Return atau Enter untuk memasukkan akhiran ganti baris, atau memformat teks terlebih dahulu dalam editor teks dengan akhiran ganti baris. Akhiran tidak akan ditampilkan di layar.

Merujuk TextView di kode

Untuk merujuk TextView di kode Java Anda, gunakan `id` sumber dayanya. Misalnya, untuk memperbarui TextView dengan teks baru, Anda perlu:

1. Mencari TextView (dengan id `show_count`) dan menetapkannya ke variabel. Anda menggunakan metode `findViewById()` kelas View, dan merujuk ke tampilan yang ingin dicari dengan menggunakan format ini:

   ```
   R.id.view_id
   ```

   Dalam hal ini, `view_id` adalah identifier sumber daya untuk tampilan:

   ```
   mShowCount = (TextView) findViewById(R.id.show_count);
   ```

2. Setelah mengambil tampilan sebagai variabel anggota TextView, selanjutnya Anda bisa menyetel teks untuk tampilan teks ke teks baru dengan menggunakan metode setText() kelas TextView:

   ```
   mShowCount.setText(mCount_text);
   ```

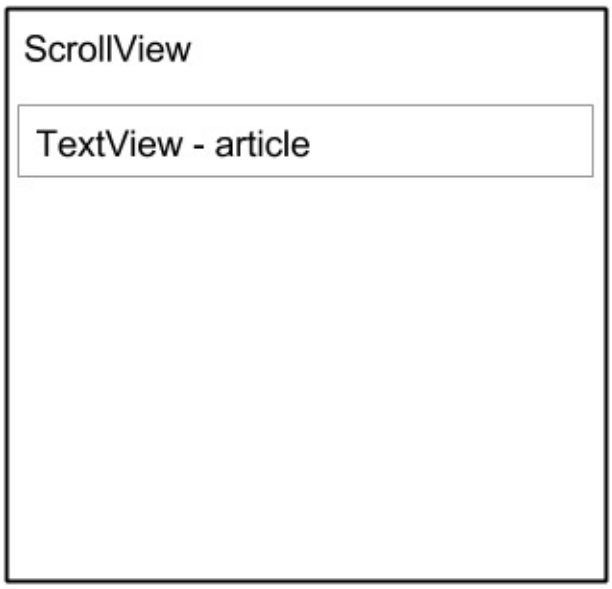
Tampilan bergulir

Jika informasi yang ingin ditampilkan di aplikasi Anda lebih besar dari tampilan perangkat, Anda bisa membuat *tampilan bergulir* yang bisa digulir pengguna secara vertikal dengan menggesek ke atas atau ke bawah, atau secara horizontal dengan menggesek ke kanan atau ke kiri.

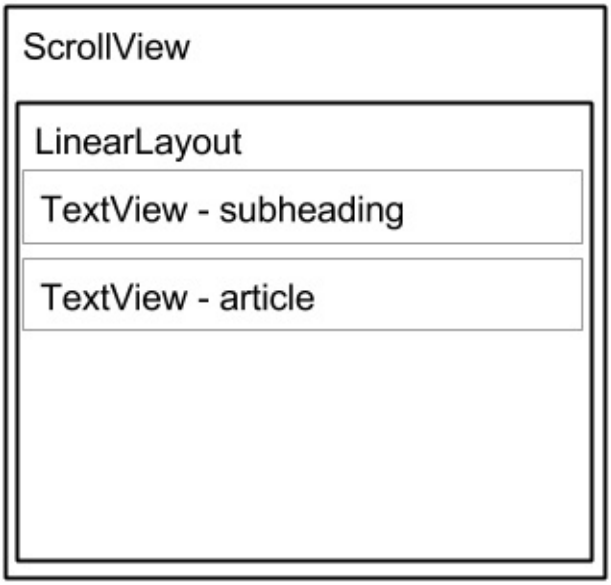
Anda biasanya akan menggunakan tampilan bergulir untuk kabar berita, artikel, atau teks panjang lainnya yang tidak muat pada tampilan. Anda juga bisa menggunakan tampilan bergulir untuk mengombinasikan tampilan (seperti TextView dan Button) dalam tampilan bergulir.

Membuat layout dengan ScrollView

Kelas ScrollView menyediakan layout untuk tampilan yang bergulir vertikal. (Untuk pengguliran horizontal, Anda perlu menggunakan HorizontalScrollView.) ScrollView adalah subkelas FrameLayout, yang berarti Anda hanya bisa menempatkan *satu* tampilan sebagai anak; anak tersebut berisi keseluruhan materi yang akan digulir.



Meskipun Anda hanya bisa menempatkan satu tampilan anak di dalam ScrollView, tampilan anak bisa menjadi grup tampilan dengan hierarki tampilan anak, seperti LinearLayout. Pilihan yang bagus untuk tampilan dalam ScrollView adalah LinearLayout yang disusun dalam orientasi vertikal.



ScrollView dan kinerja

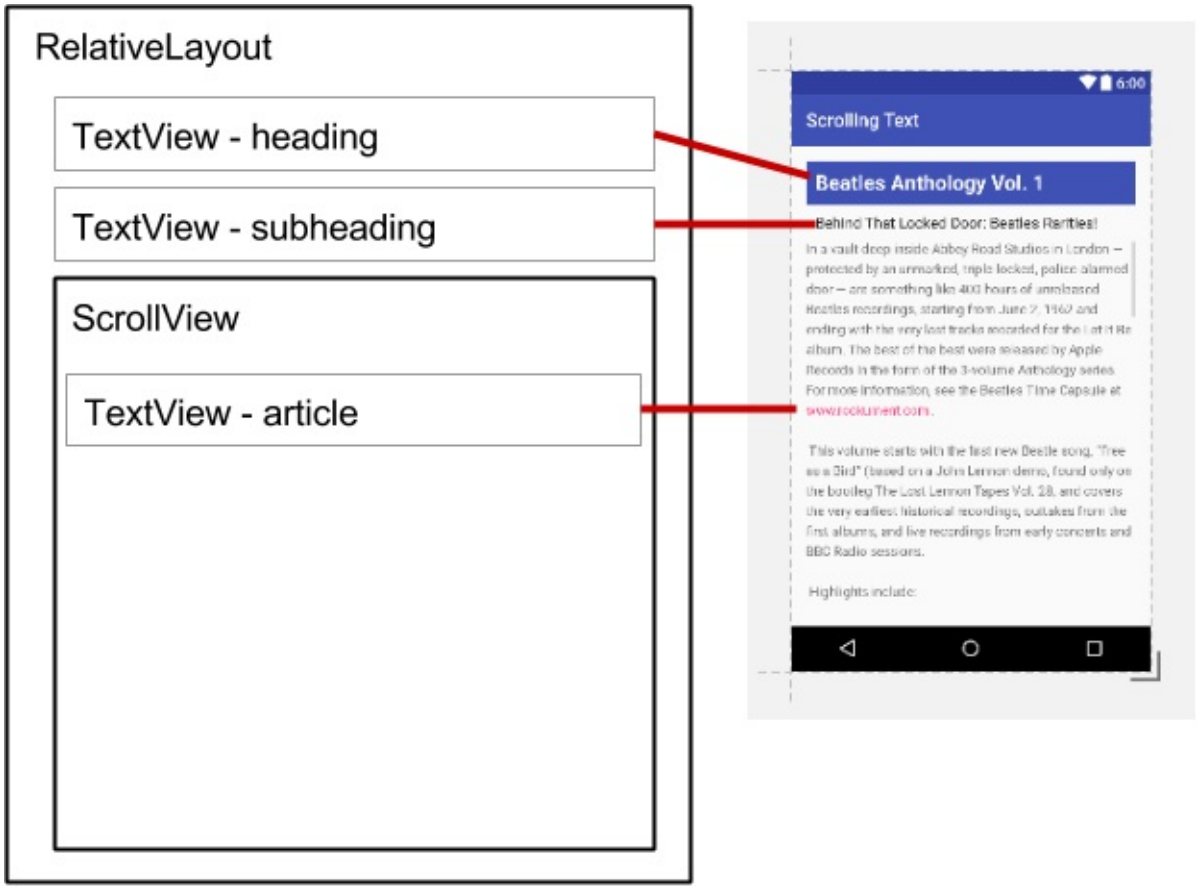
Dengan ScrollView, semua tampilan Anda berada dalam memori dan dalam hierarki tampilan sekalipun tidak ditampilkan di layar. Ini membuat ScrollView berguna untuk menggulir laman teks berbentuk bebas dengan lancar, karena teks sudah ada di memori. Akan tetapi, ScrollView bisa menggunakan banyak memori, yang bisa memengaruhi kinerja aplikasi lainnya.

Menggunakan instance LinearLayout yang disarangkan juga bisa mengakibatkan hierarki tampilan yang terlalu dalam, sehingga bisa memperlambat kinerja. Menyarangkan sejumlah instance LinearLayout yang menggunakan atribut `android:layout_weight` bisa menjadi sangat mahal karena setiap tampilan anak perlu diukur dua kali. Pertimbangkan penggunaan layout yang lebih datar seperti RelativeLayout atau GridLayout untuk meningkatkan kinerja.

Layout kompleks dengan ScrollView mungkin mengalami masalah kinerja, khususnya dengan tampilan anak seperti gambar. Kami menyarankan agar Anda *tidak* menggunakan gambar bersama ScrollView. Untuk menampilkan daftar panjang item, atau gambar, pertimbangkan untuk menggunakan RecyclerView. Selain itu, menggunakan AsyncTask akan menyediakan cara sederhana untuk melakukan pekerjaan di luar thread utama, seperti memuat gambar dalam thread latar belakang, kemudian menerapkannya ke UI setelah selesai. AsyncTask dibahas di bab lainnya.

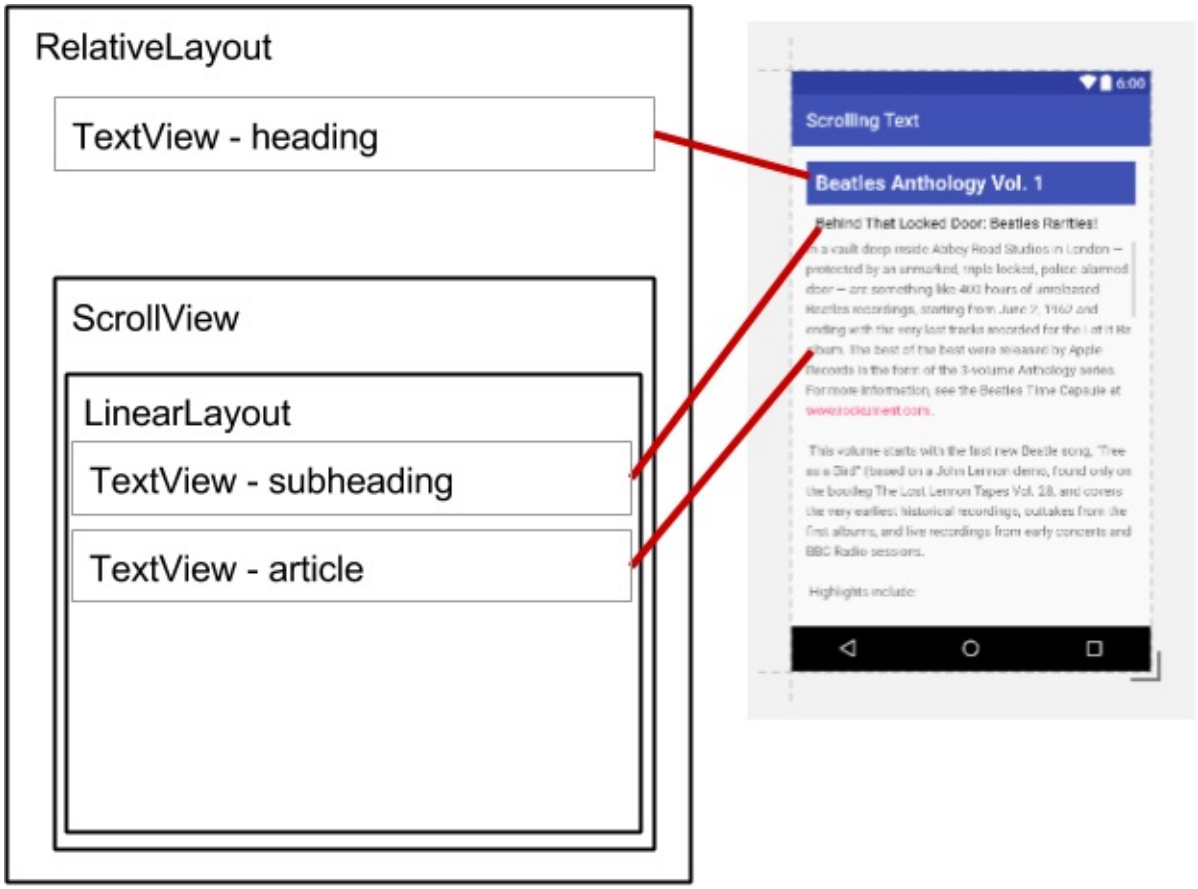
ScrollView dengan TextView

Untuk menampilkan artikel majalah yang dapat digulir pada layar, Anda dapat menggunakan RelativeLayout bagi layar yang menyertakan TextView terpisah untuk judul artikel, lainnya untuk subjudul artikel, dan TextView ketiga untuk teks artikel bergulir (lihat gambar di bawah ini), yang disetel dalam ScrollView. Satu-satunya bagian layar yang bergulir adalah ScrollView dengan teks artikel.



ScrollView dengan LinearLayout

Grup tampilan ScrollView hanya bisa berisi satu tampilan; akan tetapi, tampilan tersebut bisa berupa grup tampilan yang berisi beberapa tampilan, misalnya LinearLayout. Anda bisa *menyarangkan* grup tampilan seperti LinearLayout *dalam* grup tampilan ScrollView, sehingga menggulir apa pun yang ada di dalam LinearLayout.



Saat menambahkan LinearLayout di dalam ScrollView, gunakan `match_parent` untuk atribut `android:layout_width` LinearLayout guna mencocokkan dengan lebar grup tampilan induk (ScrollView), dan gunakan `wrap_content` untuk atribut `android:layout_height` LinearLayout guna membuat grup tampilan cukup besar untuk menampung materi dan pengisinya.

Karena ScrollView hanya mendukung pengguliran vertikal, Anda harus menyetel orientasi LinearLayout ke vertikal (dengan menggunakan atribut `android:orientation="vertical"`), sehingga keseluruhan LinearLayout akan bergulir secara vertikal. Misalnya, layout XML berikut menggulir TextView `article` bersama `article_subheading` TextView:

```
<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" />

 <TextView
         android:id="@+id/article"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:autoLink="web"
         android:lineSpacingExtra="@dimen/line_spacing"
         android:text="@string/article_text" />

 </LinearLayout>
</ScrollView>
```

```
<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" />

 <TextView
         android:id="@+id/article"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:autoLink="web"
         android:lineSpacingExtra="@dimen/line_spacing"
         android:text="@string/article_text" />

 </LinearLayout>
</ScrollView>
```

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Menggunakan Elemen TextView

Ketahui selengkapnya

- Dokumentasi Android Studio:
  - Mengenal Android Studio
  - Panduan Pengguna Android Studio
- Panduan Android API, bagian "Kembangkan":
  - TextView
  - ScrollView:
  - Sumber Daya String
  - Tampilan
  - Layout Relatif
- Desain Material:
  - Gaya
  - Menggunakan Tema Material
- Lainnya:
  - Blog Developer Android: Linkify Teks Anda!
  - Codepath: Bekerja dengan TextView

1.4: Sumber Daya untuk Membantu Anda Belajar

Materi:

- Menjelajahi dokumentasi developer Android
- Menonton video developer
- Menjelajahi contoh kode di Android SDK
- Menggunakan template aktivitas
- Menjelajahi blog developer Android
- Sumber informasi lain
- Praktik terkait

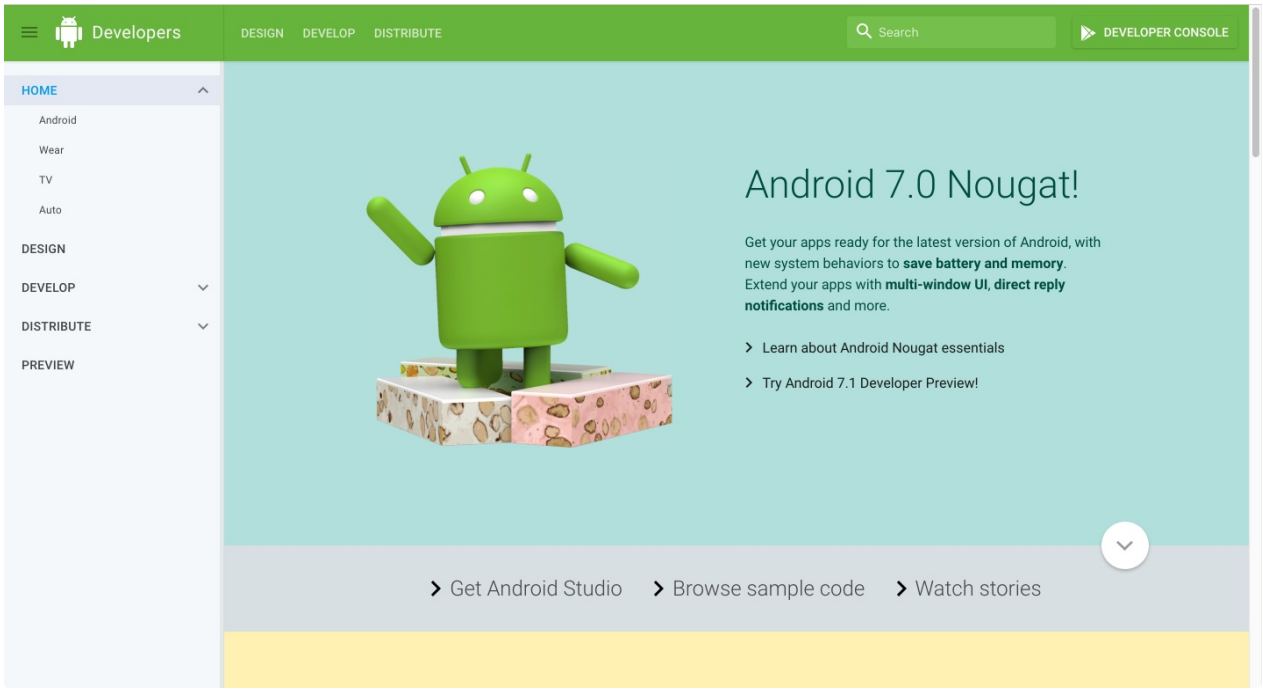
Bab ini menjelaskan sumber daya yang tersedia untuk developer Android, dan cara menggunakannya.

Menjelajahi dokumentasi developer Android

Tempat terbaik untuk mempelajari tentang development Android dan mengikuti informasi tentang alat pengembangan Android terbaru adalah dengan menjelajahi dokumentasi developer Android resmi.

developer.android.com

Laman beranda



Dokumentasi ini berisi banyak informasi yang terus diperbarui oleh Google. Untuk mulai menjelajah, klik tautan berikut pada laman beranda:

- **Dapatkan Android Studio**: Unduh Android Studio, lingkungan development terintegrasi (IDE) resmi untuk membangun aplikasi Android.
- **Jelajahi kode contoh**: Jelajahi pustaka kode contoh di GitHub untuk mengetahui cara membangun komponen yang berbeda bagi aplikasi Anda. Klik kategori di kolom kiri untuk menjelajahi contoh yang tersedia. Setiap contoh merupakan aplikasi Android yang berfungsi penuh. Anda bisa menjelajahi sumber daya dan file sumber, serta melihat struktur proyek keseluruhan. Salin dan tempel kode yang diperlukan, dan jika ingin berbagi tautan ke baris tertentu, Anda bisa mengeklik dua kali untuk mendapatkan URL. Untuk kode contoh lainnya, lihat "Menjelajahi contoh kode di

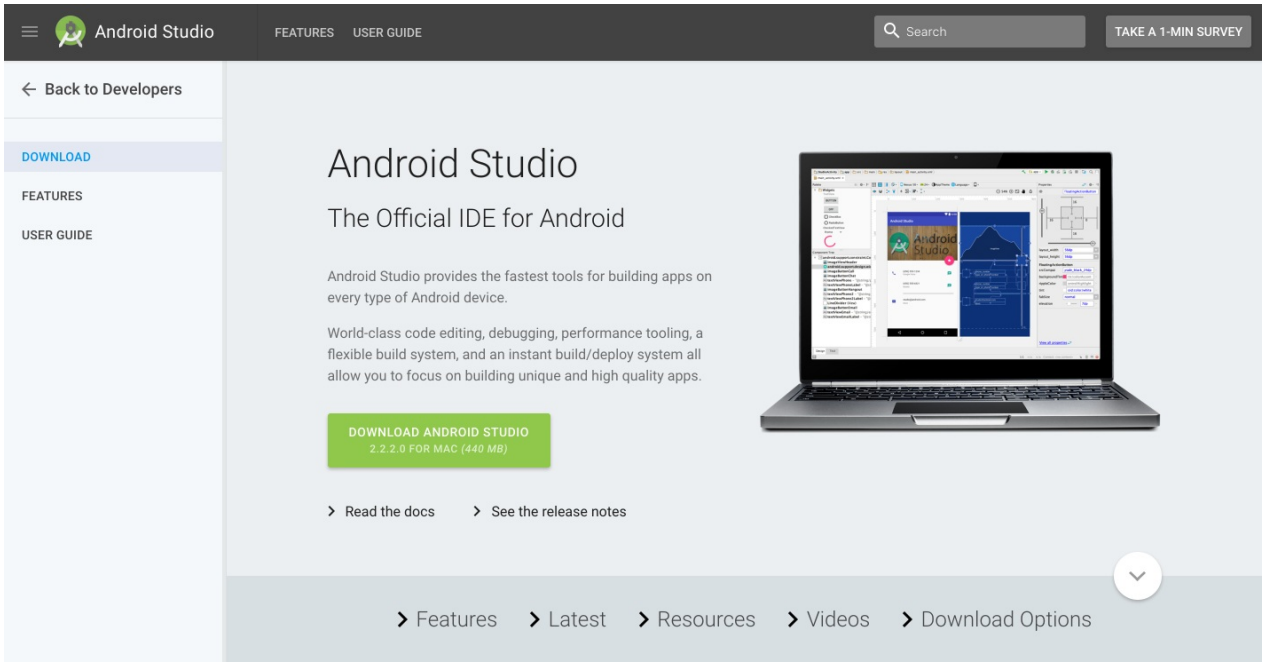
Android SDK" di bab ini.

- **Saksikan cerita**: Pelajari tentang developer Android lainnya, aplikasi, dan kesuksesan mereka bersama Android dan Google Play. Laman tersebut menawarkan video dan artikel berisi cerita terbaru tentang development Android, seperti cara developer memperbaiki pengalaman penggunanya, dan cara meningkatkan interaksi pengguna dengan aplikasi.

Laman beranda juga menawarkan tautan bagi developer Android untuk pratinjau aplikasi bagi versi Android terbaru, dan untuk bergabung dengan program developer Google Play:

- **Developer Console**: Google Play Store adalah sistem distribusi digital Google untuk aplikasi yang dikembangkan dengan Android SDK. Di laman Google Play Developer Console, Anda bisa menerima Persetujuan Developer, membayar biaya pendaftaran, dan melengkapi detail akun agar bisa bergabung dengan program developer Google Play.
- **Pratinjau**: Buka laman pratinjau untuk versi Android terbaru guna menguji kompatibilitas aplikasi Anda, dan memanfaatkan fitur baru seperti pintasan aplikasi, dukungan keyboard gambar, ikon melingkar, dan sebagainya.
- **Android**, **Wear**, **TV**, dan **Auto**: Pelajari tentang Android versi terbaru untuk ponsel cerdas dan tablet, perangkat yang dapat dikenakan, televisi, dan mobil otomatis.

Laman Android Studio



Setelah mengeklik **Dapatkan Android Studio** di laman beranda, laman Android Studio, yang ditampilkan di atas, akan muncul bersama tautan berguna berikut ini:

- **Unduh Android Studio**: Unduh Android Studio untuk sistem operasi komputer yang Anda gunakan saat ini.
- **Baca dokumentasi**: Jelajahi dokumentasi Android Studio.
- **Lihat catatan rilis**: Baca catatan rilis untuk Android Studio versi terbaru.
- **Fitur**: Pelajari tentang fitur Android Studio versi terbaru.
- **Terbaru**: Baca berita tentang Android Studio.
- **Sumber Daya**: Baca artikel tentang penggunaan Android Studio, termasuk pengantar dasar.
- **Video**: Tonton tutorial video tentang penggunaan Android Studio.
- **Opsi Unduhan**: Unduh versi Android Studio untuk sistem operasi berbeda dari yang Anda gunakan.

Dokumentasi Android Studio

Berikut ini adalah tautan ke dokumentasi Android Studio yang berguna untuk pelatihan ini:

- Mengenal Android Studio
- Dasar-Dasar Alur Kerja Developer

- Ringkasan Proyek
- Buat Ikon Aplikasi dengan Image Asset Studio
- Tambahkan Grafik Vektor Multi-Kepadatan
- Buat dan Kelola Perangkat Virtual
- Laman Android Monitor
- Debug Aplikasi Anda
- Konfigurasi Pembangunan Anda
- Tandatangani Aplikasi Anda

Desain, Kembangkan, Distribusikan, dan Pratinjau

Dokumentasi Android bisa diakses melalui tautan berikut dari laman beranda:

- **Desain**: Bagian ini mencakup Desain Material yang merupakan filosofi desain konseptual yang membahas bagaimana seharusnya tampilan dan cara kerja aplikasi pada perangkat seluler. Gunakan tautan berikut untuk mengetahui selengkapnya:
  - **Memperkenalkan desain material**: Pengantar filosofi desain material.
  - **Unduhan untuk desainer**: Unduh palet warna untuk kompatibilitas dengan spesifikasi desain material.
  - **Artikel**: Baca artikel dan berita tentang desain Android.
  - Gulir ke bawah laman Design untuk tautan ke sumber daya seperti video, template, font, dan palet warna.
  - Berikut ini adalah tautan ke bagian Design yang berguna untuk pelatihan ini:
    - Panduan Desain Material
    - Gaya
    - Menggunakan Tema Material
    - Komponen - Tombol
    - Panduan desain dialog
    - Panduan desain isyarat
    - Panduan Desain Notifikasi
    - Ikon dan sumber daya lain yang bisa diunduh
    - Desain - Pola - Navigasi
    - Panduan Sumber Daya Dapat Digambar
    - Panduan Gaya dan Tema
    - Setelan
    - Material Palette Generator
- **Kembangkan**: Di bagian inilah Anda bisa menemukan informasi antarmuka pemrograman aplikasi (API), dokumentasi referensi, tutorial, panduan alat (bantu), dan contoh kode, serta mendapatkan wawasan mengenai alat dan pustaka Android untuk mempercepat development. Anda bisa menggunakan tautan navigasi situs di kolom kiri, atau telusuri untuk menemukan hal yang dibutuhkan. Berikut ini adalah tautan populer ke bagian Kembangkan yang berguna untuk pelatihan ini:
  - Ringkasan:
    - Pengantar Android
    - Daftar Istilah Kosakata
    - Arsitektur Platform
    - Dasar-Dasar Aplikasi Android
    - Ringkasan UI
    - Versi Platform
    - Pustaka Dukungan Android
    - Bekerja dengan Izin Sistem
  - Praktik development:
    - Mendukung Versi Platform Berbeda
    - Mendukung Beberapa Layar
    - Mendukung Kepadatan Berbeda
    - Praktik Terbaik untuk Interaksi dan Keterlibatan
    - Praktik Terbaik untuk Antarmuka Pengguna

        - Praktik Terbaik untuk Pengujian
        - Menyediakan Sumber Daya
        - Mengoptimalkan Unduhan untuk Panduan Akses Jaringan Efisien
        - Praktik Terbaik untuk Izin Aplikasi
    - Artikel dan panduan pelatihan:
        - Memulai Aktivitas Lain
        - Menetapkan Tipe Metode Masukan
        - Menangani Masukan Keyboard
        - Menambahkan Bilah Aplikasi
        - Menggunakan Isyarat Sentuh
        - Membuat Daftar dan Kartu
        - Memulai Pengujian
        - Mengelola Daur Hidup Aktivitas
        - Menghubungkan ke Jaringan
        - Mengelola Penggunaan Jaringan
        - Memanipulasi Penerima Siaran Sesuai Kebutuhan
        - Menjadwalkan Alarm Berulang
        - Mentransfer Data Tanpa Menguras Baterai
        - Menyimpan File
        - Menyimpan Rangkaian Nilai-Kunci
        - Menyimpan Data di Database SQL
        - Mengonfigurasi Auto Backup for Apps
        - Bekerja dengan Izin Sistem
    - Topik umum
        - Gaya dan Tema
        - Layout
        - Menu
        - Maksud dan Filter Maksud
        - Proses dan thread
        - Loader
        - Layanan
        - Notifikasi
        - Opsi Storage
        - Melokalkan dengan Sumber Daya
        - Penyedia Materi
        - Kursor
        - Mencadangkan Data Aplikasi ke Cloud
        - Setelan
        - Izin Sistem
    - Informasi referensi:
        - Pustaka Dukungan
        - Sumber daya R.color standar Android
        - Sumber Daya Aplikasi
- **Distribusikan**: Bagian ini menyediakan informasi tentang segala sesuatu yang terjadi setelah Anda menulis aplikasi: menaruhnya di Play Store, menumbuhkan basis pengguna, dan menghasilkan uang.
    - Google Play
    - Hal-hal Penting untuk Keberhasilan Aplikasi
    - Daftar Periksa Peluncuran

Memasang dokumentasi offline

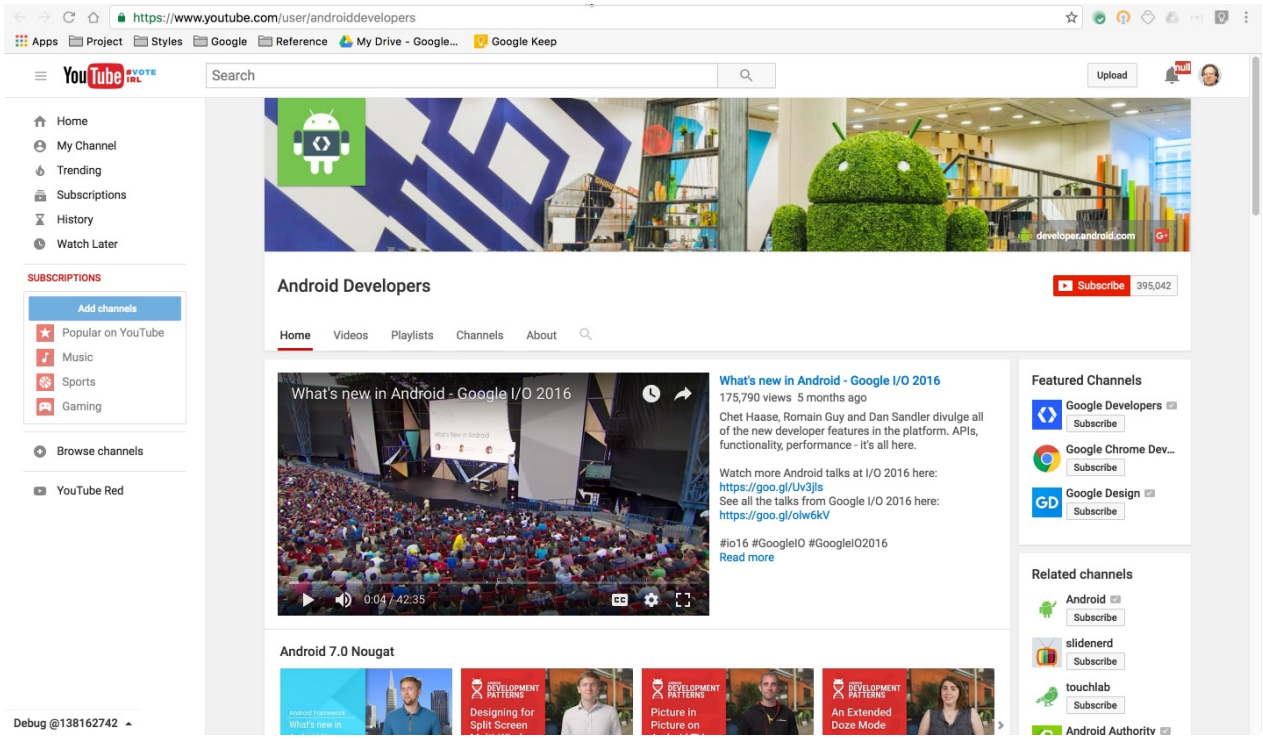
Untuk mengakses dokumentasi bahkan saat Anda tidak terhubung ke internet, pasang dokumentasi Software Development Kit (SDK) dengan menggunakan SDK Manager. Ikuti langkah-langkah ini:

1. Pilih **Tools > Android > SDK Manager**.
2. Di kolom kiri, klik **Android SDK**.
3. Pilih dan salin jalur untuk Android SDK Location di bagian atas layar, karena Anda akan memerlukannya untuk menemukan dokumentasi di komputer:

4. Klik tab **SDK Tools**. Anda bisa memasang SDK Tools tambahan yang tidak dipasang secara default, serta versi offline dokumentasi developer Android.
5. Klik kotak centang untuk "Documentation for Android SDK" jika belum dipasang, dan klik **Apply**.
6. Bila pemasangan selesai, klik **Finish**.
7. Arahkan ke direktori **sdk** yang disalin di atas, dan buka direktori **docs**.
8. Cari **index.html** dan buka.

Menonton video developer

Sebagai tambahan untuk dokumentasi Android, saluran YouTube Developer Android adalah sumber tutorial dan tip terbaik. Anda bisa berlangganan ke saluran untuk menerima notifikasi video baru melalui email. Untuk berlangganan, klik tombol **Subscribe** merah di sudut kanan atas seperti yang ditampilkan di bawah ini.



Berikut ini adalah video populer yang dirujuk dalam pelatihan ini:

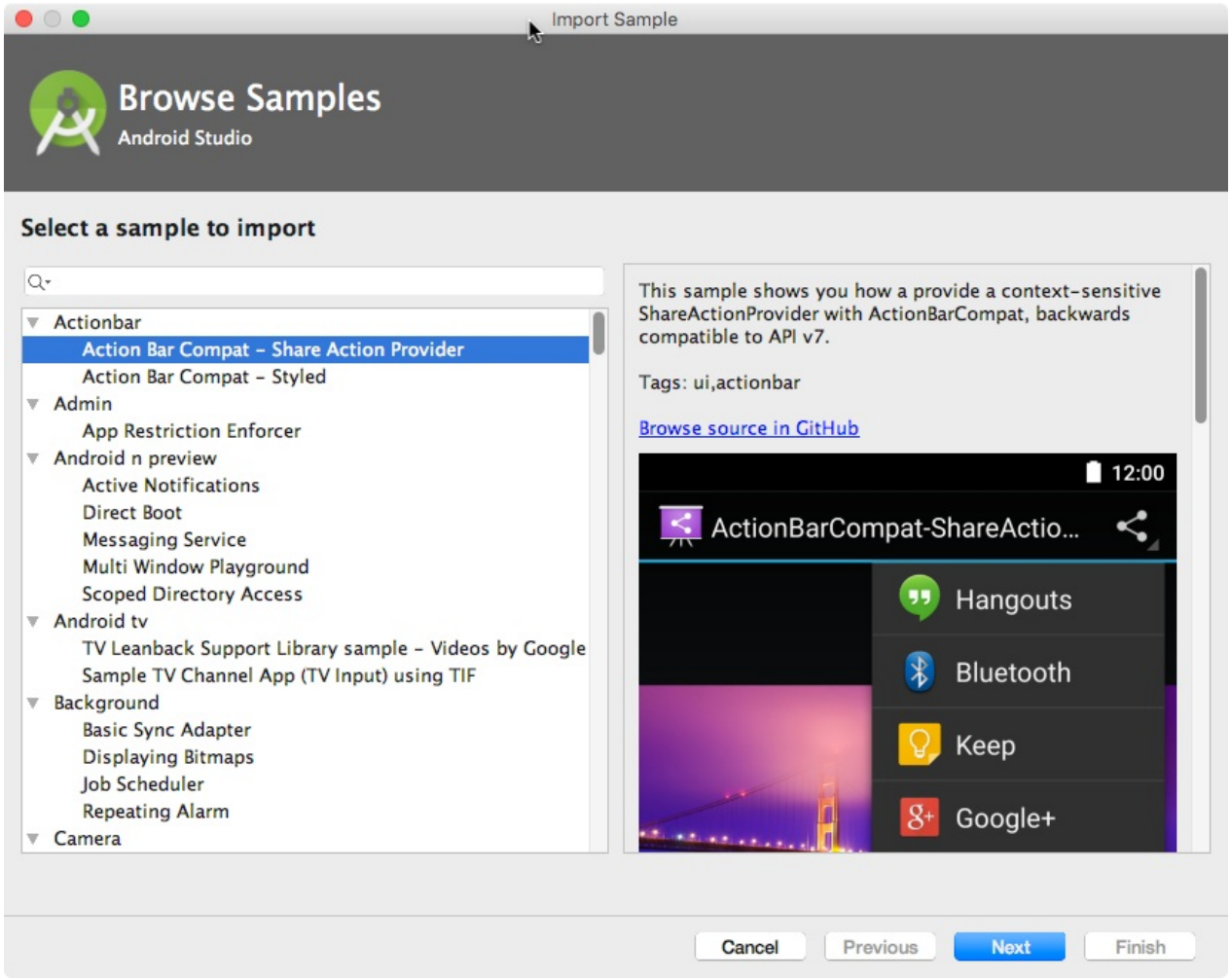
- Men-debug dan Menguji di Android Studio
- Dukungan Pengujian Android - Pola Pengujian Android No. 1
- Dukungan Pengujian Android - Pola Pengujian Android No. 2
- Dukungan Pengujian Android - Pola Pengujian Android No. 3
- Threading Kinerja 101
- Mencari AsyncTask yang Baik
- Menjadwalkan Presentasi Alarm
- Animasi RecyclerView dan di Balik Layar (Android Dev Summit 2015)
- Arsitektur Aplikasi Android: Miliaran Pengguna Berikutnya

- Daftar Putar Pola Kinerja Android

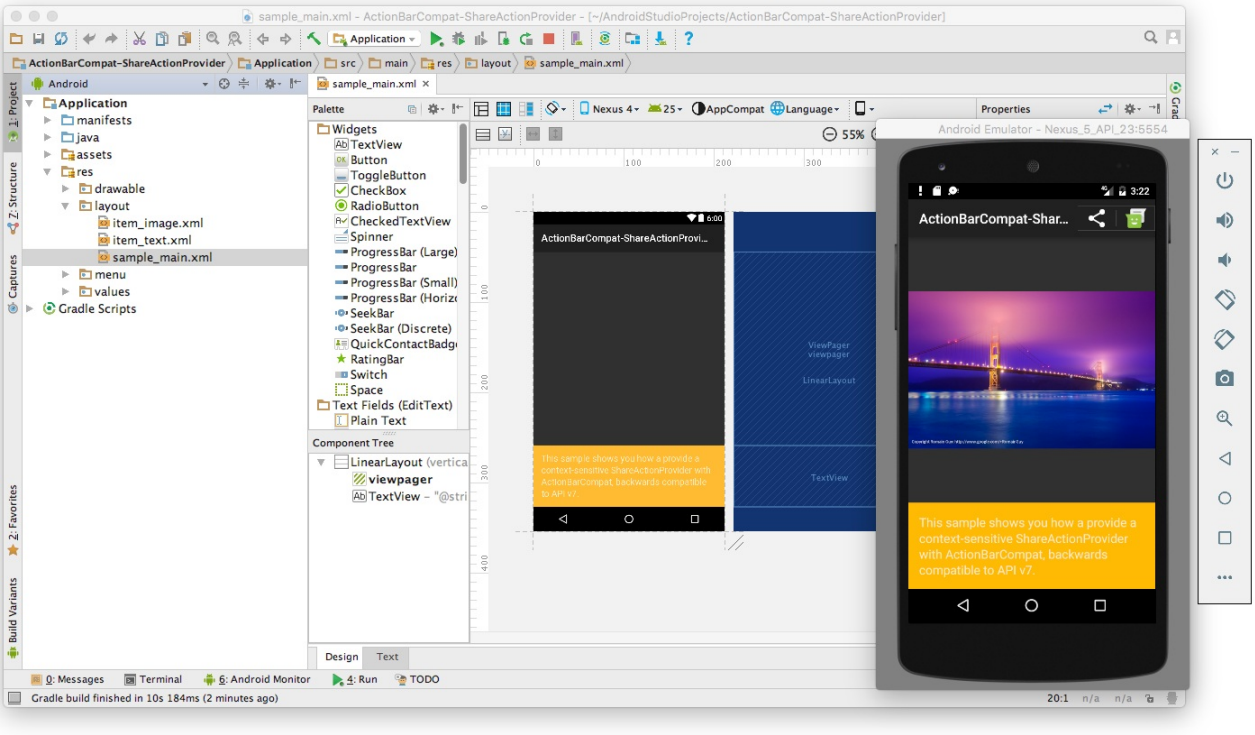
Selain itu, Udacity menawarkan kursus development Android online.

Menjelajahi contoh kode di Android SDK

Anda bisa menjelajahi ratusan contoh kode secara langsung di Android Studio. Pilih **Import an Android code sample** dari layar selamat datang Android Studio, atau pilih **File > New > Import Sample** jika proyek sudah dibuka. Jendela Browse Sample muncul seperti yang ditampilkan di bawah ini.



Pilih contoh, dan klik **Next**. Terima atau edit Application name serta Project location, dan klik **Finish**. Proyek aplikasi akan muncul seperti yang ditampilkan di bawah ini, dan Anda bisa menjalankan aplikasi di emulator yang disediakan bersama Android Studio, atau di perangkat yang terhubung.

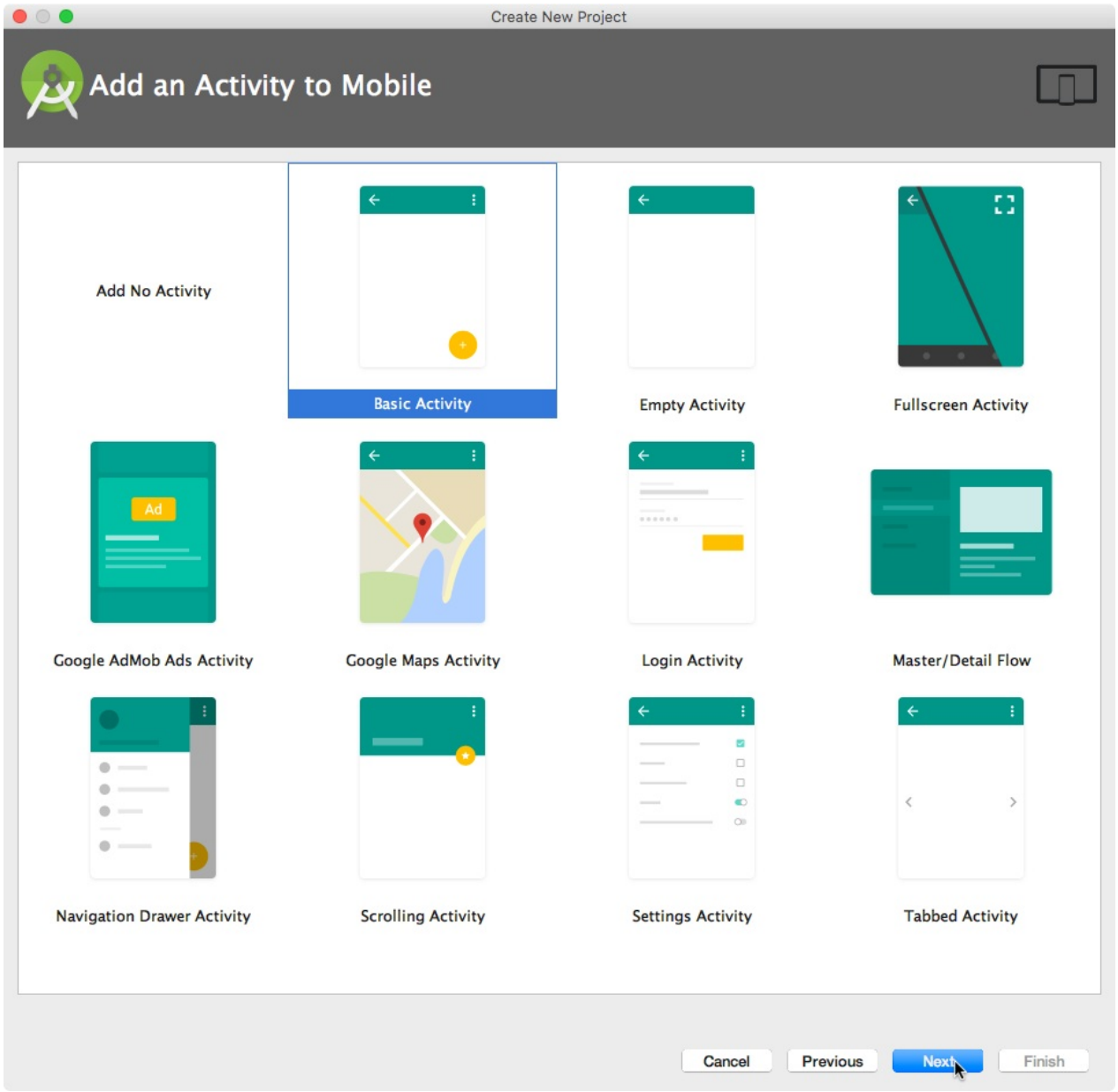


Catatan: Contoh ini dimaksudkan sebagai titik mulai development lebih jauh. Kami mendorong Anda untuk mendesain dan membangun ide sendiri.

Menggunakan template aktivitas

Android Studio menyediakan template untuk desain aktivitas umum dan yang disarankan. Penggunaan template akan menghemat waktu, dan membantu Anda mengikuti praktik terbaik untuk mengembangkan aktivitas.

Setiap template akan memasukkan aktivitas kerangka dan antarmuka pengguna. Pilih template aktivitas untuk aktivitas utama saat memulai proyek aplikasi. Anda juga bisa menambahkan template aktivitas ke proyek yang ada. Klik-kanan folder **java** di Project: Tampilan Android dan pilih **New > Activity > Gallery**.



Menjelajahi blog developer Android

Blog Developer Android menyediakan banyak artikel mengenai development Android.

Berikut ini adalah entri blog populer:

- Android Studio 2.2
- Mengamankan Android: Penyempurnaan keamanan di Nougat
- Menghubungkan aplikasi Anda ke Perangkat Wi-Fi
- Linkify Teks Anda!
- Holo Di Mana-mana
- Tips untuk membantu Anda tetap mematuhi kebijakan Google Play
- 5 Tips untuk membantu Anda meningkatkan monetisasi game-sebagai-layanan

Sumber informasi lainnya

Google dan pihak ketiga menawarkan berbagai tip dan teknik untuk development Android. Berikut ini adalah sumber informasi yang direferensikan oleh pelatihan ini:

- **Pelatihan Developer Google**: Bila Anda masih baru dalam hal pemrograman atau seorang developer berpengalaman, Google menawarkan beragam kursus online untuk mengajarkan development Android, mulai dari awal hingga mengoptimalkan kinerja aplikasi. Klik tab **Android** di bagian atas laman.
- **Google I/O Codelabs**: Google Developers Codelabs menyediakan pengalaman pengkodean langsung yang dipandu pada sejumlah topik. Sebagian besar codelab akan memandu Anda dalam proses pembangunan aplikasi kecil, atau menambahkan fitur baru ke aplikasi yang ada. Pilih **Android** dari menu tarik-turun Category di sisi kanan laman.
- **Codelab Pengujian Android**: Codelab ini menampilkan cara memulai pengujian untuk Android, termasuk menguji integrasi di Android Studio, pengujian unit, pengujian hermetik, pengujian antarmuka pengguna fungsional, dan kerangka kerja pengujian Espresso.
- **Blog Pengujian Google**: Blog ini difokuskan pada pengujian kode. Entri blog yang dirujuk dalam pelatihan ini antara lain:
  - Pengujian Otomatis untuk UI Android
  - Ukuran Pengujian
- **Stack Overflow**: Stack Overflow adalah komunitas jutaan programmer yang saling membantu. Jika Anda mengalami masalah, kemungkinan orang lain sudah mengeposkan jawabannya pada forum ini. Contoh yang dirujuk dalam pelatihan ini antara lain:
  - Bagaimana menyatakan di dalam RecyclerView di Espresso?
  - Bagaimana Menambahkan Fragmen ke Template Panel Samping Navigasi Khusus?
  - Bagaimana membuat Aktivitas Preferensi dan Fragmen Preferensi di Android?
  - Bagaimana menggunakan SharedPreferences di Android untuk menyimpan, mengambil, dan mengedit nilai
  - Bagaimana mengisi AlertDialog dari Arraylist?
  - onSavedInstanceState vs. SharedPreferences
  - Glide vs. Picasso
- **Google di GitHub**: GitHub adalah layanan hosting repositori. Layanan ini menawarkan semua kontrol versi yang didistribusikan dan fungsionalitas pengelolaan kode sumber (SCM) Git serta menambahkan fiturnya sendiri. Git adalah sistem kontrol versi yang digunakan secara luas untuk development perangkat lunak. Yang berikut ini ditampung dalam GitHub dan dirujuk di pelatihan ini:
  - Contoh Pengujian Android
  - Android Testing Support Library: Dasar-Dasar Espresso
  - Android Testing Support Library: Rujukan ringkas Espresso
  - Android Asset Studio oleh Roman Nurik
  - Kode sumber untuk latihan di GitHub
- Beragam sumber informasi yang dirujuk dalam pelatihan ini:
  - Codepath: Bekerja dengan TextView
  - SQLite.org: Keterangan lengkap Bahasa Kueri
  - Atomic Object: "Espresso – Menguji RecyclerViews pada Posisi Tertentu"
- **Penelusuran Google**: Masukkan pertanyaan ke dalam kotak telusur Google, didahului dengan "Android" untuk mempersempit penelusuran Anda. Mesin telusur Google akan mengumpulkan hasil yang relevan dari semua sumber daya. Misalnya:
  - *"Versi OS Android apa yang paling populer di India?"* Pertanyaan ini akan mengumpulkan hasil tentang pangsa pasar Android, termasuk laman Dasbor yang menyediakan ringkasan karakteristik dan versi platform perangkat yang aktif di ekosistem Android.
  - *"Android Settings Activity"* mengumpulkan berbagai artikel tentang Settings Activity termasuk laman topik Setelan, kelas PreferenceActivity, dan [Bagaimana membuat Aktivitas Preferensi dan Fragmen Preferensi di Android?] Stack Overflow(http://stackoverflow.com/questions/23523806/how-do-you-create-preference-activity-and-preference-fragment-on-android)
  - *"TextView Android"* mengumpulkan informasi tentang tampilan teks termasuk laman topik kelas TextView, kelas View, Layout, dan contoh kode dari beragam sumber.
  - Awali setiap penelusuran dengan **Android** untuk mempersempit penelusuran topik terkait Android. Misalnya, Anda bisa menelusuri keterangan kelas Android, seperti *"Android TextView"* atau *"aktivitas Android"*.

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Mempelajari Tentang Sumber Daya yang Tersedia

2.1: Memahami Aktivitas dan Maksud

Materi:

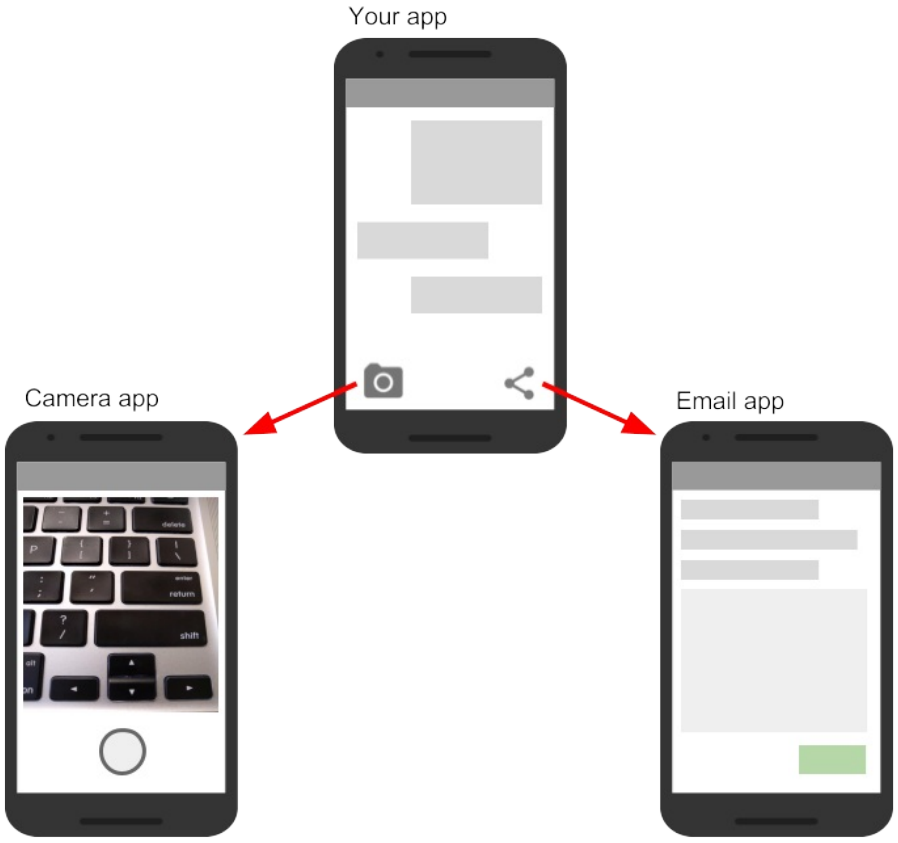
Dalam bab ini Anda akan mempelajari tentang *aktivitas*, blok pembangunan utama antarmuka pengguna aplikasi, serta penggunaan *maksud* untuk berkomunikasi di antara aktivitas.

Tentang aktivitas

Aktivitas menyatakan layar tunggal di aplikasi Anda dengan antarmuka yang bisa digunakan pengguna untuk berinteraksi. Misalnya, aplikasi email mungkin memiliki satu aktivitas yang menampilkan daftar email baru, aktivitas lain untuk menulis email, dan aktivitas lainnya lagi untuk membaca pesan satu per satu. Aplikasi Anda adalah koleksi aktivitas yang dibuat sendiri atau yang digunakan kembali dari aplikasi lain.

Meskipun aktivitas di aplikasi Anda bekerja sama membentuk pengalaman pengguna yang kohesif di aplikasi, setiap aplikasi tidak saling bergantung. Proses ini memungkinkan aplikasi memulai aktivitas di aplikasi lainnya, dan aplikasi lainnya bisa memulai aktivitas Anda (jika aplikasi mengizinkannya). Misalnya, aplikasi perpesanan yang Anda tulis bisa

memulai aktivitas di aplikasi kamera untuk mengambil gambar, kemudian memulai aktivitas di aplikasi email untuk memungkinkan pengguna berbagi gambar itu di email.



Umumnya, satu aktivitas di aplikasi ditetapkan sebagai aktivitas "utama", yang disajikan kepada pengguna saat membuka aplikasi untuk pertama kali. Kemudian setiap aktivitas bisa memulai aktivitas lainnya untuk melakukan tindakan yang berbeda.

Setiap kali aktivitas baru dimulai, aktivitas sebelumnya akan dihentikan, namun sistem mempertahankan aktivitas dalam tumpukan ("back-stack"). Bila pengguna selesai dengan aktivitas saat ini dan menekan tombol Kembali, aktivitas akan muncul dari tumpukan (dan dimusnahkan) dan aktivitas sebelumnya dilanjutkan.

Bila aktivitas dihentikan karena aktivitas baru dimulai, aktivitas pertama akan diberi tahu tentang perubahan tersebut dengan metode callback daur hidup aktivitas. Daur hidup Aktivitas adalah serangkaian keadaan aktivitas, mulai dari pertama kali dibuat, hingga setiap kali dihentikan atau dilanjutkan, hingga bila sistem memusnahkannya. Anda akan mengetahui selengkapnya tentang daur hidup aktivitas di bab berikutnya.

Membuat aktivitas

Untuk mengimplementasikan aktivitas di aplikasi Anda, lakukan yang berikut ini:

- Buat kelas Java aktivitas.
- Implementasikan antarmuka pengguna untuk aktivitas itu.
- Deklarasikan aktivitas baru itu di manifes aplikasi.

Bila Anda membuat proyek baru untuk aplikasi atau menambahkan aktivitas baru ke aplikasi, di Android Studio (dengan File > New > Activity), kode template untuk setiap tugas ini akan disediakan untuk Anda.

Buat kelas aktivitas

Aktivitas adalah subkelas dari kelas Activity atau salah satu dari subkelasnya. Jika Anda membuat proyek baru di Android Studio, aktivitas tersebut secara default menjadi subkelas dari kelas AppCompatActivity. Kelas AppCompatActivity adalah subkelas Activity yang memungkinkan Anda menggunakan fitur aplikasi Android terbaru seperti bilah aksi dan desain material, sementara tetap memungkinkan aplikasi tersebut kompatibel dengan perangkat yang menjalankan Android versi lama.

Inilah subkelas kerangka dari aktivitas AppCompatActivity:

```
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
   }
}
```

Tugas pertama Anda di subkelas aktivitas adalah mengimplementasikan metode callback daur hidup aktivitas standar (seperti OnCreate()) untuk menangani perubahan keadaan aktivitas. Perubahan keadaan ini meliputi hal-hal seperti kapan aktivitas dibuat, dihentikan, dilanjutkan, atau dimusnahkan. Anda akan mengetahui selengkapnya tentang daur hidup aktivitas dan callback daur hidup di bab berikutnya.

Salah satu callback yang diperlukan dan harus diimplementasikan oleh aplikasi Anda adalah metode onCreate(). Sistem memanggil metode ini bila membuat aktivitas, dan semua komponen penting aktivitas Anda harus melakukan diinisialisasi di sini. Yang paling penting, metode OnCreate() memanggil setContentView() untuk membuat layout utama aktivitas.

Anda biasanya mendefinisikan antarmuka pengguna untuk aktivitas di satu atau beberapa file layout XML. Bila metode setContentView() dipanggil dengan jalur ke file layout, sistem akan membuat semua tampilan awal dari layout yang ditetapkan dan menambahkannya ke aktivitas Anda. Hal ini sering kali disebut sebagai *memekarkan* layout.

Sering kali Anda mungkin juga ingin mengimplementasikan metode onPause() di kelas aktivitas. Sistem memanggil metode ini sebagai indikasi pertama bahwa pengguna meninggalkan aktivitas Anda (walaupun tidak selalu berarti aktivitas akan dimusnahkan). Di sinilah biasanya Anda harus mengikat perubahan yang harus dipertahankan di luar sesi pengguna saat ini (karena pengguna mungkin tidak akan kembali). Anda akan mengetahui selengkapnya tentang callback onPause() dan semua callback daur hidup lainnya di bab berikutnya.

Selain daur hidup callback, Anda juga bisa mengimplementasikan metode di aktivitas untuk menangani perilaku lainnya seperti masukan pengguna atau klik tombol.

Implementasikan antarmuka pengguna

Antarmuka pengguna untuk aktivitas disediakan menurut hierarki tampilan, yang mengontrol ruang tertentu dalam jendela aktivitas dan bisa merespons interaksi pengguna.

Cara yang paling umum untuk mendefinisikan antarmuka pengguna yang menggunakan tampilan adalah dengan file layout XML yang disimpan sebagai bagian dari sumber daya aplikasi Anda. Mendefinisikan layout di XML memungkinkan Anda untuk mengelola desain antarmuka pengguna secara terpisah dari kode sumber yang mendefinisikan perilaku aktivitas.

Anda juga bisa membuat tampilan baru secara langsung di kode aktivitas Anda dengan menyisipkan objek tampilan baru ke dalam ViewGroup, kemudian meneruskan ViewGroup akar ke setContentView(). Setelah layout dimekarkan -- apa pun sumbernya -- Anda bisa menambahkan lebih banyak tampilan di Java di mana saja dalam hierarki tampilan.

Deklarasikan aktivitas di manifes

Setiap aktivitas di aplikasi Anda harus dideklarasikan di manifes aplikasi Android bersama elemen `<activity>` , di dalam `<application>` . Bila Anda membuat proyek baru atau menambahkan aktivitas baru ke proyek di Android Studio, manifes akan dibuat atau diperbarui untuk menyertakan deklarasi aktivitas kerangka bagi setiap aktivitas. Inilah deklarasi untuk aktivitas utama.

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

Elemen `<activity>` menyertakan sejumlah atribut untuk mendefinisikan properti aktivitas seperti label, ikon, atau tema. Satu-satunya atribut yang diperlukan adalah android:name, yang menetapkan nama kelas aktivitas (seperti "MainActivity"). Lihat referensi elemen `<activity>` untuk informasi selengkapnya tentang deklarasi aktivitas.

Elemen `<activity>` juga bisa menyertakan deklarasi untuk filter maksud. Filter maksud menetapkan jenis maksud yang akan diterima aktivitas Anda.

```
 <intent-filter>
   <action android:name="android.intent.action.MAIN" />
   <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

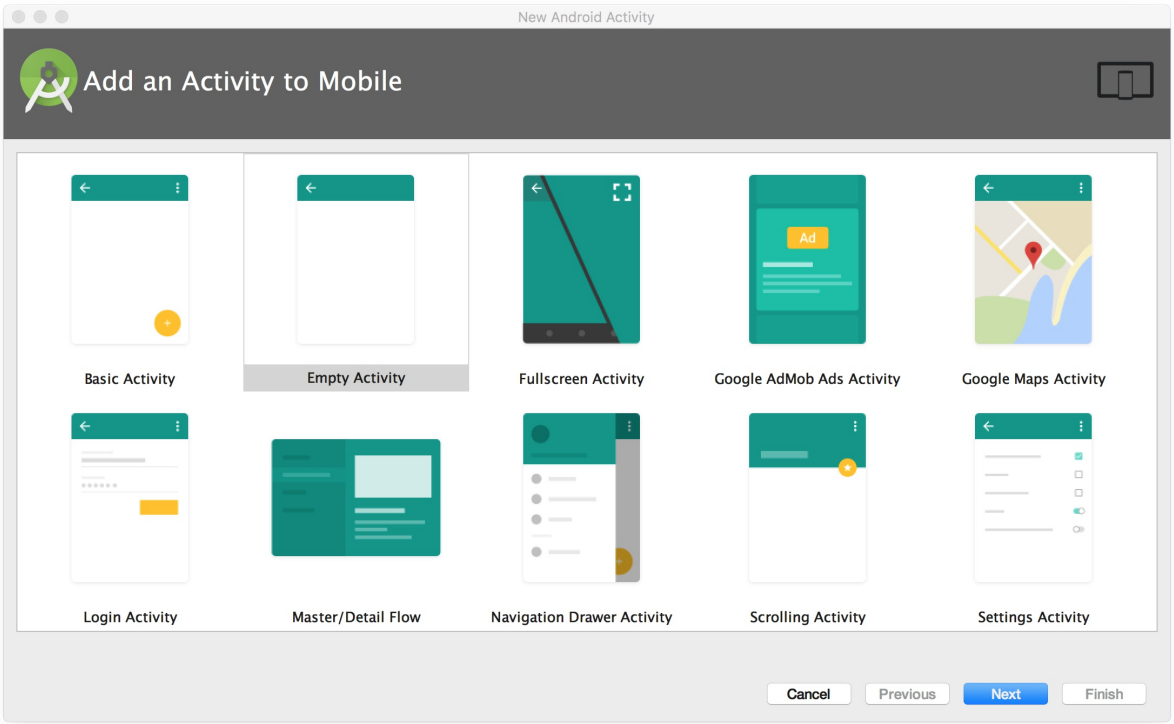
Filter maksud harus menyertakan setidaknya satu elemen, dan juga bisa menyertakan sebuah dan opsional. Aktivitas utama untuk aplikasi Anda memerlukan filter maksud yang mendefinisikan aksi "main" dan kategori "launcher" agar sistem bisa meluncurkan aplikasi. Android Studio membuat filter maksud ini untuk aktivitas utama di proyek Anda:

Elemen menetapkan bahwa ini adalah titik masuk "utama" ke aplikasi. Elemen menetapkan bahwa aktivitas ini harus tercantum dalam peluncur aplikasi sistem (untuk memungkinkan pengguna meluncurkan aktivitas ini).

Aktivitas lain di aplikasi Anda juga bisa mendeklarasikan filter maksud, namun hanya aktivitas utama yang harus menyertakan aksi "utama". Anda akan mengetahui selengkapnya tentang maksud implisit dan filter maksud dalam bab berikutnya.

Tambahkan lebih banyak aktivitas ke proyek Anda

Aktivitas utama untuk aplikasi Anda dan file layout terkait disertakan bersama proyek bila Anda membuatnya. Anda bisa menambahkan aktivitas baru ke proyek di Android Studio dengan menu **File > New > Activity**. Pilih template aktivitas yang ingin Anda gunakan, atau buka Gallery untuk melihat semua template yang tersedia.



Bila memilih sebuah template aktivitas, Anda akan melihat serangkaian layar yang sama untuk membuat aktivitas baru yang dilakukan saat membuat proyek di awal. Android Studio menyediakan tiga hal ini untuk setiap aktivitas baru di aplikasi Anda:

- File Java untuk aktivitas baru dengan definisi kelas kerangka dan metode onCreate(). Aktivitas baru, seperti aktivitas utama, adalah subkelas AppCompatActivity.
- File XML yang berisi layout untuk aktivitas baru. Perhatikan, metode setContentView() di kelas aktivitas akan memerkarkan layout baru ini.
- Elemen `<activity>` tambahan di manifes Android yang menetapkan aktivitas baru. Definisi aktivitas kedua tidak menyertakan filter maksud apa pun. Jika Anda ingin menggunakan aktivitas ini hanya dalam aplikasi (dan tidak memungkinkan aktivitas tersebut dimulai oleh aplikasi lain), maka tidak perlu menambahkan filter.

Tentang maksud

Semua aktivitas Android dimulai atau diaktifkan dengan *maksud*. Maksud adalah objek pesan yang membuat permintaan yang akan digunakan oleh waktu proses Android untuk memulai aktivitas atau komponen aplikasi lainnya di aplikasi Anda atau di beberapa aplikasi lainnya. Anda tidak bisa memulai aktivitas itu sendiri;

Bila aplikasi pertama kali dimulai dari layar utama perangkat, waktu proses Android akan mengirimkan maksud ke aplikasi Anda untuk memulai aktivitas utama aplikasi (yang didefinisikan dengan aksi MAIN dan kategori LAUNCHER di Manifes Android). Untuk memulai aktivitas lain di aplikasi Anda, atau meminta tindakan untuk dilakukan oleh beberapa aktivitas lain yang tersedia di perangkat, bangunlah maksud sendiri dengan kelas Intent dan panggil metode startActivity() untuk mengirim maksud itu.

Selain untuk memulai aktivitas, maksud juga digunakan untuk meneruskan data di antara aktivitas. Bila membuat maksud untuk memulai aktivitas baru, Anda bisa menyertakan informasi tentang data yang diinginkan untuk mengoperasikan aktivitas baru itu. Jadi, misalnya, aktivitas email yang menampilkan daftar pesan bisa mengirim maksud ke aktivitas yang menampilkan pesan itu. Aktivitas tampilan memerlukan data tentang pesan yang akan ditampilkan, dan Anda bisa menyertakan data itu di maksud.

Dalam bab ini, Anda akan mempelajari tentang penggunaan maksud bersama aktivitas, namun maksud juga digunakan untuk memulai layanan dan penerima siaran. Anda akan mempelajari tentang kedua komponen aplikasi itu nanti di buku ini.

Tipe maksud

Ada dua tipe maksud di Android:

- *Maksud eksplisit* menetapkan aktivitas penerima (atau komponen lainnya) melalui nama kelas yang benar-benar memenuhi syarat di aktivitas tersebut. Gunakan maksud eksplisit untuk memulai komponen di aplikasi milik sendiri (misalnya, untuk beralih layar di antarmuka pengguna), karena Anda sudah mengetahui paket dan nama kelas komponen itu.
- *Maksud implisit* tidak menetapkan aktivitas tertentu atau komponen lainnya untuk menerima maksud. Sebagai gantinya, Anda mendeklarasikan aksi umum yang dilakukan di maksud tersebut. Sistem Android mencocokkan permintaan Anda dengan aktivitas atau komponen lainnya yang bisa menangani aksi permintaan tersebut. Anda akan mengetahui selengkapnya tentang maksud implisit dalam bab berikutnya.

Objek dan bidang Intent

Objek Intent adalah instance kelas Intent. Untuk maksud eksplisit, bidang kunci suatu maksud menyertakan yang berikut ini:

- *Kelas* aktivitas (untuk maksud eksplisit). Ini adalah nama kelas aktivitas atau komponen lainnya yang akan menerima maksud, misalnya, com.example.SampleActivity.class. Gunakan konstruktor maksud atau metode setComponent(), setComponentName() atau setClassName() maksud untuk menetapkan kelas.
- *Data* maksud. Bidang data maksud berisi referensi ke data yang Anda inginkan untuk mengoperasikan aktivitas penerima, sebagai objek Uri.
- *Ekstra* maksud. Ini adalah pasangan nilai-kunci yang membawa informasi yang diperlukan aktivitas penerima untuk melakukan aksi yang diminta.
- *Flag* maksud. Ini adalah bit metadata tambahan, yang didefinisikan oleh kelas Intent. Flag dapat menginstruksikan sistem Android tentang cara meluncurkan aktivitas atau cara memperlakukan aktivitas setelah diluncurkan.

Untuk maksud implisit, Anda juga mungkin perlu mendefinisikan kategori dan aksi maksud. Anda akan mengetahui selengkapnya tentang kategori dan aksi maksud di bagian 2.3.

Memulai aktivitas dengan maksud eksplisit

Untuk memulai aktivitas tertentu dari aktivitas lain, gunakan maksud eksplisit dan metode startActivity(). Maksud eksplisit menyertakan nama kelas yang benar-benar memenuhi syarat untuk aktivitas atau komponen lainnya di objek Intent. Semua bidang maksud lainnya bersifat opsional, dan nol secara default.

Misalnya, jika Anda ingin memulai ShowMessageActivity untuk menampilkan pesan tertentu di aplikasi email, gunakan kode seperti ini.

```
Intent messageIntent = new Intent(this, ShowMessageActivity.class);
startActivity(messageIntent);
```

Konstruktor Intent menggunakan dua argumen untuk maksud eksplisit.

- Konteks aplikasi. Dalam contoh ini, kelas aktivitas menyediakan materi (di sini, `this`).
- Komponen tertentu untuk dimulai (`ShowMessageActivity.class`).

Gunakan metode startActivity() bersama objek maksud baru sebagai satu-satunya argumen. Metode startActivity() mengirim maksud ke sistem Android, yang meluncurkan kelas ShowMessageActivity atas nama aplikasi Anda. Aktivitas baru muncul pada layar dan aktivitas pembuatnya dihentikan sementara.

Aktivitas yang dimulai tetap pada layar hingga pengguna mengetuk tombol kembali pada perangkat, pada saat itu aktivitas ditutup dan diklaim kembali oleh sistem, dan aktivitas yang menghasilkannya akan dilanjutkan. Anda juga bisa menutup aktivitas yang dimulai secara manual sebagai respons terhadap aksi pengguna (seperti klik tombol) dengan metode finish():

```
public void closeActivity (View view) {
    finish();
}
```

Meneruskan data di antara aktivitas dengan maksud

Selain untuk memulai satu aktivitas dari aktivitas lain, Anda juga menggunakan maksud untuk meneruskan informasi di antara aktivitas. Objek maksud yang Anda gunakan untuk memulai aktivitas bisa menyertakan *data* maksud (URI objek untuk bertindak), atau *ekstra* maksud, yang merupakan bit data tambahan yang mungkin diperlukan aktivitas.

Di aktivitas (pengirim) pertama:

1. Buat objek Intent.
2. Masukkan data atau ekstra ke dalam maksud itu.
3. Mulailah aktivitas baru dengan startActivity().

Di aktivitas (penerima) kedua:

1. Dapatkan objek maksud yang digunakan memulai aktivitas.
2. Ambil data atau ekstra dari objek Intent.

Waktu untuk menggunakan data maksud atau ekstra maksud

Anda bisa menggunakan data maksud dan ekstra maksud untuk meneruskan data di antara aktivitas. Ada sejumlah perbedaan utama antara data dan ekstra yang menentukan mana yang harus Anda gunakan.

Data maksud hanya bisa menyimpan satu bagian informasi. URI yang menyatakan lokasi data yang ingin Anda gunakan untuk mengoperasikan. URI tersebut bisa berupa URL laman web (http://), nomor telepon (tel://), lokasi geografis (geo://), atau URI khusus lainnya yang Anda definisikan.

Gunakan bidang maksud data:

- Bila Anda hanya memiliki satu bagian informasi yang perlu dikirim ke aktivitas yang telah dimulai.
- Bila informasi itu adalah lokasi data yang bisa dinyatakan dengan URI.

Ekstra maksud adalah untuk data arbitrer lainnya yang ingin Anda teruskan ke aktivitas yang telah dimulai. Ekstra maksud disimpan di objek Bundle sebagai pasangan kunci dan nilai. Bundle adalah peta, yang dioptimalkan untuk Android, dengan tombol berupa string, dan nilai-nilainya bisa berupa tipe objek atau primitif apa pun (objek harus mengimplementasikan antarmuka Parcelable). Untuk memasukkan data ke dalam ekstra maksud, Anda bisa menggunakan salah satu metode putExtra() kelas Intent, atau membuat bundel sendiri dan memasukkannya ke dalam maksud dengan putExtras().

Gunakan ekstra maksud:

- Jika Anda ingin meneruskan lebih dari satu bagian informasi ke aktivitas yang telah dimulai.
- Jika informasi yang ingin Anda teruskan tidak bisa dinyatakan melalui URI.

Data dan ekstra maksud tidak eksklusif; Anda bisa menggunakan data URI dan ekstra untuk informasi tambahan yang diperlukan aktivitas yang dimulai untuk memproses data dalam URI itu.

Tambahkan data ke maksud

Untuk menambahkan data ke maksud eksplisit dari aktivitas pembuatnya, buat objek maksud seperti yang Anda lakukan sebelumnya:

```
Intent messageIntent = new Intent(this, ShowMessageActivity.class);
```

Gunakan metode setData() bersama objek Uri untuk menambahkan URI itu ke maksud. Beberapa contoh penggunaan setData() bersama URI:

```
// A web page URL
messageIntent.setData(Uri.parse("http://www.google.com"));
// a Sample file URI
messageIntent.setData(Uri.fromFile(new File("/sdcard/sample.jpg")));
// A sample content: URI for your app's data model
messageIntent.setData(Uri.parse("content://mysample.provider/data"));
// Custom URI
messageIntent.setData(Uri.parse("custom:" + dataID + buttonId));
```

Perlu diingat bahwa bidang data hanya bisa berisi URI tunggal; jika Anda memanggil setData() beberapa kali, hanya nilai terakhir yang akan digunakan. Gunakan ekstra maksud untuk menyertakan informasi tambahan (termasuk URI.)

Setelah menambahkan data, Anda bisa memulai aktivitas dengan maksud seperti biasanya.

```
startActivity(messageIntent);
```

Tambahkan ekstra ke maksud

Untuk menambahkan ekstra maksud ke maksud eksplisit dari aktivitas pembuatnya:

1. Tentukan kunci yang akan digunakan untuk informasi yang ingin dimasukkan ke dalam ekstra, atau definisikan sendiri. Setiap bagian informasi memerlukan kunci unik.
2. Gunakan metode putExtra() untuk menambahkan pasangan kunci/nilai ke ekstra maksud. Secara opsional Anda bisa membuat objek Bundle, menambahkan data ke bundel, kemudian menambahkan bundel ke maksud.

Kelas Intent menyertakan sejumlah kunci ekstra maksud yang bisa digunakan, didefinisikan sebagai konstanta yang dimulai dengan kata EXTRA_. Misalnya, Anda bisa menggunakan Intent.EXTRA_EMAIL untuk menunjukkan larik alamat email (sebagai string), atau Intent.EXTRA_REFERRER untuk menetapkan informasi tentang aktivitas pembuat yang mengirim maksud tersebut.

Anda juga bisa mendefinisikan kunci ekstra maksud milik sendiri. Secara konvensional Anda mendefinisikan kunci ekstra maksud sebagai variabel-variabel statis dengan nama yang dimulai kata EXTRA_. Untuk menjamin kunci tersebut unik, nilai string kunci itu sendiri harus diawali dengan nama kelas yang benar-benar memenuhi syarat aplikasi. Misalnya:

```
public final static String EXTRA_MESSAGE = "com.example.mysampleapp.MESSAGE";
public final static String EXTRA_POSITION_X = "com.example.mysampleapp.X";
public final static String EXTRA_POSITION_Y = "com.example.mysampleapp.Y";
```

Buat objek maksud (jika belum ada):

```
Intent messageIntent = new Intent(this, ShowMessageActivity.class);
```

Gunakan metode putExtra() bersama kunci untuk memasukkan data ke dalam ekstra maksud. Kelas Intent mendefinisikan banyak metode putExtra() untuk jenis data yang berbeda:

```
messageIntent.putExtra(EXTRA_MESSAGE, "this is my message");
messageIntent.putExtra(EXTRA_POSITION_X, 100);
messageIntent.putExtra(EXTRA_POSITION_Y, 500);
```

Atau, Anda bisa membuat bundel baru dan mengisinya dengan ekstra maksud. Bundel mendefinisikan banyak metode "put" untuk jenis data primitif yang berbeda serta objek yang mengimplementasikan antarmuka Parcelable Android atau Serializable Java.

```
Bundle extras = new Bundle();
extras.putString(EXTRA_MESSAGE, "this is my message");
extras.putInt(EXTRA_POSITION_X, 100);
extras.putInt(EXTRA_POSITION_Y, 500);
```

Setelah Anda mengisi bundel tersebut, tambahkan ke maksud dengan metode putExtras() (perhatikan "s" di Extras):

```
messageIntent.putExtras(extras);
```

Mulai aktivitas dengan maksud seperti biasa:

```
startActivity(messageIntent);
```

Ambil data dari maksud di aktivitas yang dimulai

Bila Anda memulai aktivitas bersama maksud, aktivitas yang telah dimulai akan memiliki akses ke maksud dan data yang dimuatnya.

Untuk mengambil maksud yang digunakan untuk memulai aktivitas (atau komponen lain), gunakan metode getIntent():

```
Intent intent = getIntent();
```

Gunakan getData() untuk mendapatkan URI dari maksud itu:

```
Uri locationUri = getData();
```

Untuk mendapatkan ekstra dari maksud, Anda perlu mengetahui kunci untuk pasangan kunci/nilai. Anda bisa menggunakan ekstra Intent standar jika telah menggunakannya, atau bisa menggunakan kunci yang didefinisikan di aktivitas pembuatnya (jika didefinisikan sebagai publik.)

Gunakan salah satu metode getExtra() untuk mengekstrak data ekstra dari objek maksud:

```
String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
int positionX = intent.getIntExtra(MainActivity.EXTRA_POSITION_X);
int positionY = intent.getIntExtra(MainActivity.EXTRA_POSITION_Y);
```

Atau Anda bisa mendapatkan seluruh bundel ekstra dari maksud dan mengekstrak nilai dengan beragam metode Bundle:

```
Bundle extras = intent.getExtras();
String message = extras.getString(MainActivity.EXTRA_MESSAGE);
```

Mendapatkan data kembali dari aktivitas

Bila Anda memulai aktivitas bersama sebuah maksud, aktivitas pembuatnya akan dihentikan sementara, dan aktivitas baru tetap di layar hingga pengguna mengeklik tombol kembali, atau panggil metode finish() di handler klik atau fungsi lainnya yang mengakhiri keterlibatan pengguna pada aktivitas ini.

Kadang-kadang bila mengirim data ke aktivitas bersama sebuah maksud, Anda juga ingin mendapatkan kembali data dari maksud itu. Misalnya, Anda mungkin memulai aktivitas galeri foto yang memungkinkan pengguna memilih foto. Dalam hal ini aktivitas asli Anda perlu menerima informasi tentang foto yang dipilih pengguna dari aktivitas yang diluncurkan.

Untuk meluncurkan aktivitas baru dan mendapatkan kembali hasilnya, lakukan langkah-langkah berikut di aktivitas pembuatnya:

1. Sebagai ganti meluncurkan aktivitas bersama startActivity(), panggil startActivityForResult() bersama maksud dan

kode permintaan.
2. Buat maksud baru di aktivitas yang diluncurkan dan tambahkan data yang dikembalikan ke maksud itu.
3. Implementasikan onActivityResult() di aktivitas pembuatnya untuk memproses data yang dikembalikan.

Anda akan mempelajari tentang setiap langkah di bagian berikut ini.

Gunakan startActivityForResult() untuk meluncurkan aktivitas

Untuk mendapatkan kembali data dari aktivitas yang diluncurkan, mulailah aktivitas itu bersama metode startActivityForResult() sebagai ganti startActivity().

```
startActivityForResult(messageIntent, TEXT_REQUEST);
```

Metode startActivityForResult(), seperti startActivity(), mengambil argumen maksud yang berisi informasi tentang aktivitas yang diluncurkan dan data yang dikirim ke aktivitas itu. Akan tetapi, metode startActivityForResult() juga memerlukan kode permintaan.

Kode permintaan adalah integer yang mengidentifikasi permintaan dan bisa digunakan untuk membedakan hasil bila Anda memproses data yang dikembalikan. Misalnya, jika meluncurkan satu aktivitas untuk mengambil foto dan aktivitas lain untuk memilih foto dari galeri, Anda akan memerlukan kode permintaan yang berbeda untuk mengidentifikasi permintaan yang menjadi pemilik data yang dikembalikan.

Secara konvensional, Anda mendefinisikan kode permintaan sebagai variabel-variabel integer statis dengan nama yang menyertakan REQUEST. Gunakan integer yang berbeda untuk setiap kode. Misalnya:

```
public static final int PHOTO_REQUEST = 1;
public static final int PHOTO_PICK_REQUEST = 2;
public static final int TEXT_REQUEST = 3;
```

Kembalikan respons dari aktivitas yang diluncurkan

Data respons dari aktivitas yang diluncurkan kembali ke aktivitas pembuatnya akan dikirim dalam maksud, baik dalam data maupun ekstra. Anda membentuk maksud yang dikembalikan ini dan memasukkan data ke dalamnya menggunakan cara yang sangat mirip dengan yang Anda lakukan untuk maksud yang mengirimnya. Biasanya aktivitas yang diluncurkan akan memiliki metode onClick atau metode callback masukan pengguna lain yang Anda gunakan untuk memproses aksi pengguna dan menutup aktivitas. Di sini juga Anda membentuk respons.

Untuk mengembalikan data dari aktivitas yang diluncurkan, buat objek maksud kosong yang baru.

```
Intent returnIntent = new Intent();
```

Catatan: Untuk menghindari kebingungan atas data yang dikirim dengan data yang dikembalikan, gunakan objek maksud baru, bukan menggunakan kembali objek maksud pengirim asal.
Maksud hasil yang dikembalikan tidak memerlukan kelas atau nama komponen berakhir di tempat yang tepat. Sistem Android akan mengarahkan respons kembali ke aktivitas pembuatnya untuk Anda.

Tambahkan data atau ekstra ke maksud dengan cara sama seperti yang Anda lakukan pada maksud asal. Anda mungkin perlu mendefinisikan kunci untuk ekstra maksud yang dikembalikan pada awal kelas.

```
public final static String EXTRA_RETURN_MESSAGE =
    "com.example.mysampleapp.RETURN_MESSAGE";
```

Selanjutnya masukkan data yang dikembalikan ke maksud seperti biasa. Di sini, pesan yang dikembalikan adalah ekstra maksud dengan kunci EXTRA_RETURN_MESSAGE.

```
messageIntent.putExtra(EXTRA_RETURN_MESSAGE, mMessage);
```

Gunakan metode setResult() bersama kode respons dan maksud dengan data respons:

```
setResult(RESULT_OK,replyIntent);
```

Kode respons didefinisikan oleh kelas Activity, dan bisa berupa

- RESULT_OK: permintaan berhasil.
- RESULT_CANCELED: pengguna membatalkan operasi.
- RESULT_FIRST_USER: untuk mendefinisikan kode hasil milik Anda.

Anda akan menggunakan kode hasil di aktivitas pembuatnya.

Terakhir, panggil finish() untuk menutup aktivitas dan melanjutkan aktivitas pembuatnya:

```
finish();
```

Baca data respons di onActivityResult()

Karena sekarang aktivitas yang diluncurkan telah mengirimkan data kembali ke aktivitas pembuatnya bersama maksud, aktivitas pertama itu harus menangani data tersebut. Untuk menangani data yang dikembalikan di aktivitas pembuatnya, implementasikan metode callback onActivityResult(). Inilah sebuah contoh sederhana.

```
public void onActivityResult(int requestCode, int resultCode,  Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == TEXT_REQUEST) {
        if (resultCode == RESULT_OK) {
            String reply =
                data.getStringExtra(SecondActivity.EXTRA_RETURN_MESSAGE);
                // process data
        }
    }
}
```

Tiga argumen untuk onActivityResult() berisi semua informasi yang Anda perlukan untuk menangani data yang dikembalikan.

- **Kode permintaan.** Kode permintaan yang Anda setel saat meluncurkan aktivitas bersama startActivityForResult(). Jika Anda meluncurkan aktivitas yang berbeda untuk melakukan operasi yang berbeda, gunakan kode ini untuk mengidentifikasi data tertentu yang Anda dapatkan kembali.
- **Kode hasil:** kode hasil yang disetel di aktivitas yang diluncurkan, biasanya salah satu dari RESULT_OK atau RESULT_CANCELED.
- **Data maksud**. maksud berisi data yang dikembalikan dari aktivitas peluncur.

Metode contoh yang ditampilkan di atas menampilkan logika tipikal untuk menangani kode permintaan dan kode respons. Pengujian pertama adalah untuk permintaan TEXT_REQUEST, dan bahwa hasilnya adalah berhasil. Di dalam isi pengujian itu, ekstrak informasi yang dikembalikan dari maksud. Gunakan getData() untuk mendapatkan data maksud, atau getExtra() untuk mengambil nilai dari ekstra maksud bersama kunci tertentu.

Navigasi aktivitas

Setiap aplikasi dengan kompleksitas apa pun yang Anda bangun akan menyertakan beberapa aktivitas, baik yang didesain dan diimplementasikan oleh Anda, maupun yang berpotensi di aplikasi lain. Karena pengguna Anda berpindah-pindah dalam aplikasi dan di antara aktivitas, navigasi yang konsisten menjadi semakin penting untuk pengalaman pengguna aplikasi. Beberapa hal lebih mengganggu pengguna daripada navigasi dasar yang berperilaku tidak konsisten dan tak terduga. Mendesain navigasi aplikasi Anda dengan bijak akan membuat penggunaan aplikasi tersebut bisa diprediksi dan bisa diandalkan pengguna.

Sistem Android mendukung dua bentuk strategi navigasi yang berbeda untuk aplikasi Anda.

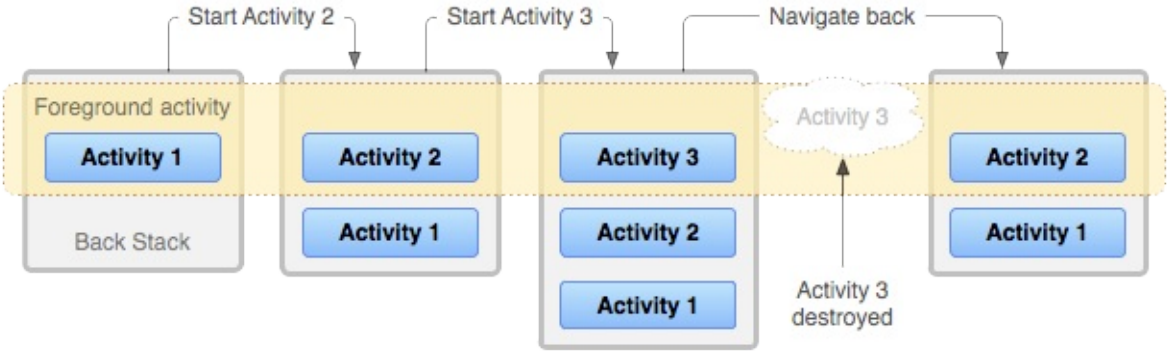
- Navigasi Sementara atau Kembali, yang disediakan melalui tombol kembali di perangkat, dan back-stack.
- Navigasi leluhur, atau Naik, yang disediakan oleh Anda sebagai opsi di bilah aksi aplikasi.

Navigasi kembali, tugas, dan back-stack

Navigasi Kembali memungkinkan pengguna Anda untuk kembali ke aktivitas sebelumnya dengan mengetuk tombol kembali di perangkat  . Navigasi kembali juga disebut navigasi *sementara* karena tombol kembali menyusuri riwayat layar yang baru saja ditampilkan, dalam urutan kronologi terbalik.

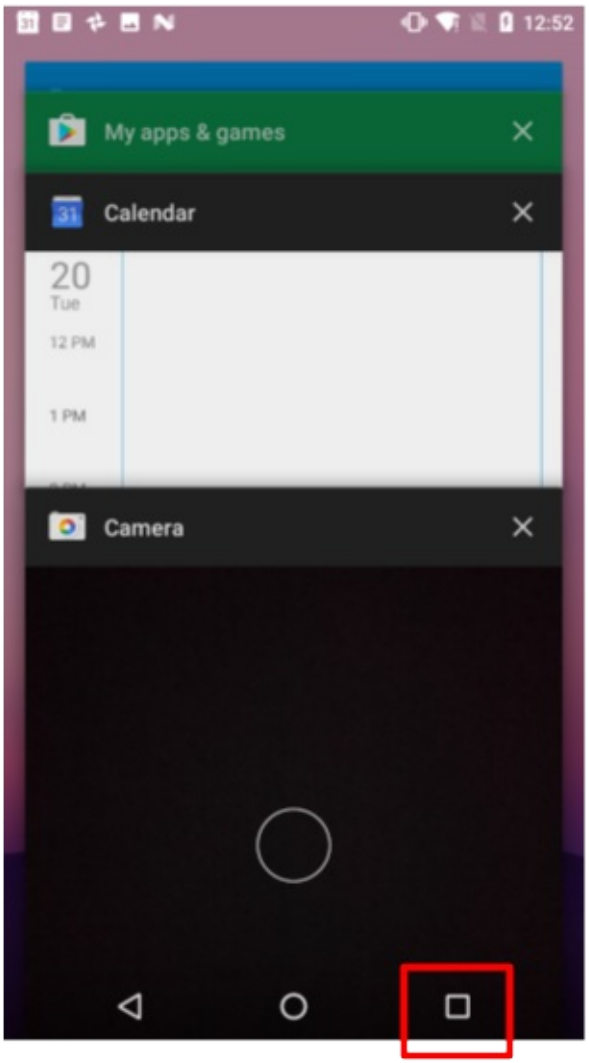
Back-stack adalah serangkaian aktivitas yang telah dikunjungi pengguna dan bisa dikunjungi kembali oleh pengguna dengan tombol kembali. Setiap kali aktivitas baru dimulai, aktivitas akan didorong ke back-stack dan mengambil fokus pengguna. Aktivitas sebelumnya akan dihentikan, namun tetap tersedia di back-stack. Back-stack beroperasi berdasarkan mekanisme "masuk terakhir, keluar pertama", jadi bila pengguna selesai dengan aktivitas saat ini dan menekan tombol Kembali, aktivitas tersebut akan dimunculkan dari tumpukan (serta dimusnahkan) dan aktivitas sebelumnya dilanjutkan.

Karena aplikasi bisa memulai aktivitas baik di dalam maupun di luar satu aplikasi, back-stack berisi semua aktivitas yang telah diluncurkan oleh pengguna dalam urutan terbalik. Setiap kali pengguna menekan tombol Kembali, setiap aktivitas di tumpukan akan dimunculkan untuk membuka aktivitas sebelumnya, hingga pengguna kembali ke layar Utama.



Android menyediakan back-stack untuk setiap *tugas*. Tugas adalah konsep penyusunan semua aktivitas yang berinteraksi dengan pengguna saat melakukan operasi, baik di dalam aplikasi maupun di beberapa aplikasi. Sebagian besar tugas dimulai dari layar utama Android, dan mengetuk ikon aplikasi akan memulai tugas (serta back-stack baru) untuk aplikasi itu. Jika pengguna menggunakan aplikasi sebentar, mengetuk beranda, dan memulai aplikasi, aplikasi baru itu akan diluncurkan di tugasnya sendiri dan memiliki back-stack sendiri. Jika pengguna kembali ke aplikasi pertama, back-stack tugas pertama akan kembali. Menyusuri dengan tombol kembali hanya akan mengembalikan ke aktivitas di tugas saat ini,

bukan untuk semua tugas yang berjalan pada perangkat. Android memungkinkan pengguna menyusuri berbagai tugas bersama ringkasan atau layar tugas saat ini, yang bisa diakses dengan tombol segi empat di sudut kanan bawah perangkat  .



Dalam sebagian besar kasus, Anda tidak perlu khawatir tentang pengelolaan tugas atau back-stack untuk aplikasi—sistem terus melacak hal ini untuk Anda, dan tombol kembali selalu tersedia pada perangkat.

Akan tetapi, mungkin ada saatnya Anda ingin mengganti perilaku default untuk tugas atau untuk back-stack. Misalnya, jika layar Anda berisi browser web yang disematkan yang memungkinkan pengguna menyusuri laman web, Anda mungkin ingin menggunakan perilaku kembali default di browser bila pengguna menekan tombol *Kembali* di perangkat, daripada mengembalikan ke aktivitas sebelumnya. Anda mungkin juga perlu mengubah perilaku default aplikasi dalam kasus khusus lainnya seperti pada notifikasi atau widget, sehingga aktivitas yang berada jauh di dalam aplikasi Anda mungkin diluncurkan sebagai tugasnya sendiri, tanpa back-stack sama sekali. Anda akan mengetahui selengkapnya tentang pengelolaan tugas dan back-stack di bagian berikutnya.

Navigasi naik

Misalnya, jika aktivitas utama di aplikasi email adalah daftar semua pesan, memilih sebuah pesan akan meluncurkan aktivitas kedua untuk menampilkan satu email itu. Dalam hal ini, aktivitas pesan akan menyediakan tombol Naik yang mengembalikan ke daftar pesan.

Perilaku tombol Naik didefinisikan oleh Anda di setiap aktivitas berdasarkan cara mendesain navigasi aplikasi. Dalam banyak kasus, navigasi Naik dan Kembali mungkin menyediakan perilaku yang sama: cuma mengembalikan ke aktivitas sebelumnya. Misalnya, aktivitas Setelan mungkin tersedia dari aktivitas apa pun di aplikasi Anda, sehingga "naik" sama dengan kembali -- cuma mengembalikan pengguna ke tempat sebelumnya di hierarki.

Menyediakan perilaku Naik untuk aplikasi Anda adalah hal yang opsional, namun praktik desain yang baik, adalah menyediakan navigasi yang konsisten untuk berbagai aktivitas di aplikasi Anda.

Implementasikan navigasi naik bersama aktivitas induk

Dengan proyek template standar di Android Studio, mudah untuk mengimplementasikan navigasi Naik. Jika satu aktivitas adalah anak aktivitas lain di hierarki aktivitas aplikasi, tetapkan induk aktivitas itu di Manifes Android.

Mulai di Android 4.1 (API level 16), deklarasikan induk logis setiap aktivitas dengan menetapkan atribut android:parentActivityName di elemen `<activity>` . Untuk mendukung Android versi lama, sertakan informasi `<meta-data>` untuk mendefinisikan aktivitas induk secara eksplisit. Gunakan kedua metode agar kompatibel mundur dengan semua versi Android.

Inilah definisi kerangka untuk kedua aktivitas utama (induk) dan aktivitas kedua (anak):

```
<application ... >
    <!-- The main/home activity (it has no parent activity) -->
         <activity
        android:name=".MainActivity" ...>
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>

    </activity>
    <!-- A child of the main activity -->
    <activity android:name=".SecondActivity"
        android:label="@string/activity2_name"
        android:parentActivityName=".MainActivity">
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.android.twoactivities.MainActivity" />
    </activity>
</application>
```

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Buat dan Mulai Aktivitas

Ketahui selengkapnya

- Dasar-Dasar Aplikasi Android
- Memulai Aktivitas Lain
- Aktivitas (Panduan API)
- Aktivitas (Referensi API)
- Maksud dan Filter Maksud (Panduan API)
- Maksud (Referensi API)

2.2: Daur Hidup Aktivitas dan Mengelola Keadaan

Materi:

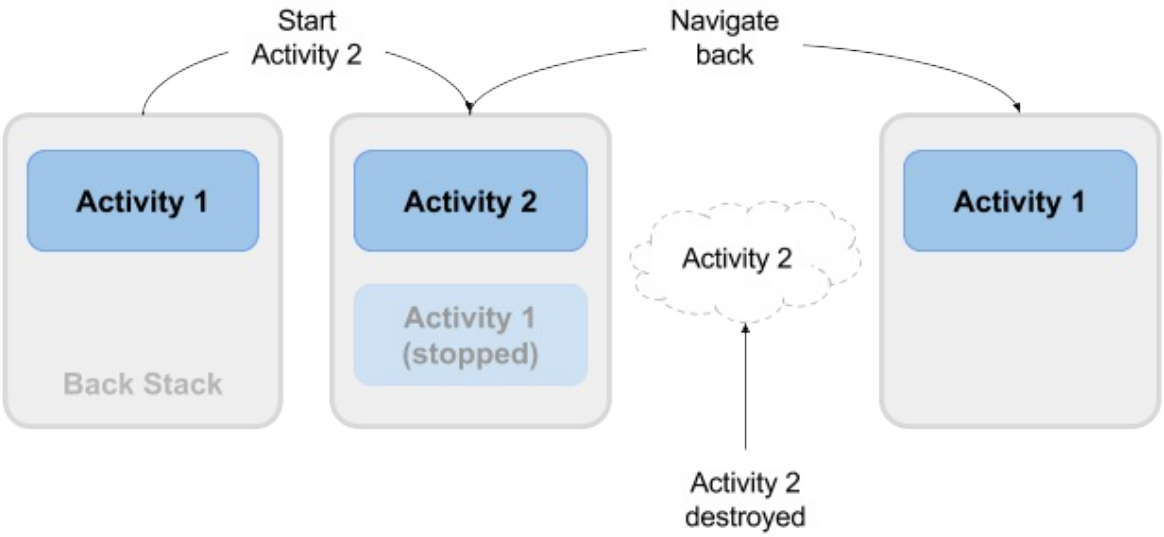
- Pengantar
- Tentang daur hidup aktivitas
- Keadaan aktivitas dan metode callback daur hidup
- Perubahan konfigurasi dan keadaan aktivitas
- Praktik Terkait
- Ketahui selengkapnya

Dalam bab ini, Anda akan mempelajari tentang daur hidup aktivitas, kejadian callback yang bisa Anda implementasikan untuk melakukan tugas di setiap tahap daur hidup, dan cara menangani keadaan instance aktivitas di seluruh daur hidup aktivitas.

Tentang daur hidup aktivitas

Daur hidup aktivitas adalah serangkaian keadaan aktivitas yang terjadi selama masa pakai keseluruhan, dari pertama kali dibuat hingga ketika dimusnahkan dan sistem mengklaim kembali sumber daya aktivitas tersebut. Karena pengguna berinteraksi dengan aplikasi Anda dan aplikasi lain pada perangkat, aktivitas yang berbeda akan beralih ke keadaan yang berbeda.

Misalnya, bila Anda memulai aplikasi, aktivitas utama aplikasi (Aktivitas 1) akan dimulai, muncul ke latar depan, dan menerima fokus pengguna. Bila Anda memulai aktivitas kedua (Aktivitas 2), aktivitas baru tersebut juga akan dibuat dan dimulai, dan aktivitas utama akan dihentikan. Bila Anda selesai dengan aktivitas kedua dan mengarah kembali, aktivitas utama akan dilanjutkan. Aktivitas kedua berhenti dan tidak lagi diperlukan; jika pengguna tidak melanjutkan aktivitas kedua, aktivitas akhirnya akan dimusnahkan oleh sistem.

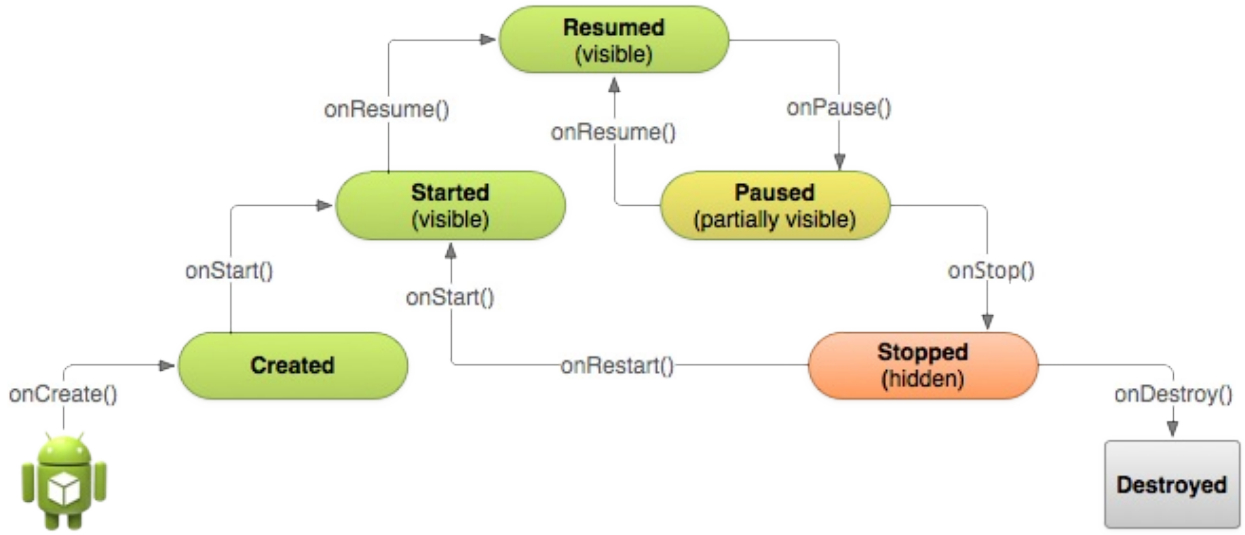


Keadaan aktivitas dan metode callback daur hidup

Bila aktivitas bertransisi masuk dan keluar dari keadaan daur hidup berbeda saat berjalan, sistem Android akan memanggil sejumlah metode callback daur hidup pada setiap tahap. Semua metode callback adalah kaitan yang bisa Anda ganti di setiap kelas Activity untuk mendefinisikan perilaku aktivitas bila pengguna meninggalkan dan memasuki aktivitas kembali.

Perlu diingat bahwa keadaan daur hidup (dan callback) adalah per aktivitas, bukan per aplikasi, dan Anda bisa mengimplementasikan perilaku berbeda di titik yang berbeda dalam daur hidup aktivitas yang berbeda di aplikasi.

Gambar ini menampilkan setiap keadaan aktivitas dan metode callback yang terjadi saat transisi aktivitas di antara keadaan yang berbeda:



Bergantung pada kompleksitas aktivitas, mungkin tidak semua metode callback daur hidup diimplementasikan di aktivitas. Akan tetapi, Anda perlu memahami satu per satu dan mengimplementasikan metode tersebut untuk memastikan aplikasi berperilaku seperti harapan pengguna. Mengelola daur hidup aktivitas Anda dengan mengimplementasikan metode callback sangat penting untuk mengembangkan aplikasi yang kuat dan fleksibel.

Aktivitas Dibuat (metode onCreate())

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // The activity is being created.
}
```

Aktivitas Anda masuk ke keadaan yang dibuat bila dimulai untuk pertama kali. Bila aktivitas dibuat pertama kali, sistem akan memanggil metode onCreate() untuk melakukan inisialisasi aktivitas itu. Misalnya, bila pengguna mengetuk ikon aplikasi dari layar utama atau beranda untuk memulai aplikasi itu, sistem akan memanggil metode onCreate() untuk aktivitas dalam aplikasi yang Anda deklarasikan sebagai aktivitas "peluncur" atau "utama". Dalam hal ini metode onCreate() aktivitas utama analog dengan metode main() di program lainnya.

Demikian juga, jika aplikasi Anda memulai aktivitas lain dengan maksud (baik eksplisit maupun implisit), sistem akan mencocokkan permintaan maksud dengan aktivitas dan memanggil onCreate() untuk aktivitas baru itu.

Metode onCreate() adalah satu-satunya callback yang harus Anda implementasikan di kelas aktivitas. Di metode onCreate(), Anda menjalankan logika startup aplikasi dasar yang hanya boleh terjadi sekali, seperti mempersiapkan antarmuka pengguna, menetapkan variabel-variabel cakupan kelas, atau mempersiapkan tugas latar belakang.

Dibuat adalah keadaan sementara; aktivitas tetap berada dalam keadaan dibuat hanya selama dibutuhkan untuk menjalankan onCreate(), kemudian aktivitas beralih ke keadaan dimulai.

Aktivitas Dimulai (metode onStart())

```
@Override
protected void onStart() {
    super.onStart();
    // The activity is about to become visible.
}
```

Setelah aktivitas Anda diinisialisasi dengan onCreate(), sistem akan memanggil metode onStart(), dan aktivitas berada dalam keadaan dimulai. Metode onStart() juga dipanggil jika aktivitas yang dihentikan kembali ke latar depan, seperti bila pengguna mengeklik tombol kembali atau naik untuk mengarah ke layar sebelumnya. Meskipun OnCreate() hanya dipanggil sekali bila aktivitas dibuat, metode onStart() bisa digunakan berkali-kali selama daur hidup aktivitas saat pengguna menyusuri aplikasi Anda.

Bila aktivitas berada dalam keadaan dimulai dan terlihat di layar, pengguna tidak bisa berinteraksi dengan aktivitas tersebut hingga onResume() dipanggil, aktivitas berjalan, dan aktivitas berada di latar depan.

Biasanya Anda mengimplementasikan onStart() di aktivitas sebagai pasangan untuk metode OnStop(). Misalnya, jika Anda melepas sumber daya perangkat keras (seperti GPS atau sensor) bila aktivitas dihentikan, Anda bisa mendaftarkan ulang sumber daya tersebut di metode onStart().

Dimulai, seperti halnya dibuat, adalah keadaan sementara. Setelah memulai, aktivitas berpindah ke keadaan dilanjutkan (berjalan).

Aktivitas dilanjutkan/berjalan (metode onResume())

```
@Override
protected void onResume() {
    super.onResume();
    // The activity has become visible (it is now "resumed").
}
```

Aktivitas Anda berada dalam keadaan dilanjutkan saat diinisialisasi, terlihat pada layar, dan siap digunakan. Keadaan dilanjutkan biasanya disebut keadaan berjalan, karena dalam keadaan ini pengguna benar-benar berinteraksi dengan aplikasi Anda.

Saat pertama aktivitas dimulai, sistem akan memanggil metode onResume() persis setelah onStart(). Metode onResume() juga mungkin dipanggil beberapa kali, setiap kali aplikasi kembali dari keadaan dihentikan sementara.

Sebagaimana dengan metode onStart() dan OnStop(), yang diimplementasikan berpasangan, Anda biasanya hanya mengimplementasikan onResume() sebagai pasangan untuk onPause(). Misalnya, jika dalam metode onPause() Anda menghentikan setiap animasi pada layar, Anda akan memulai lagi animasi tersebut di onResume().

Aktivitas ini tetap berada dalam keadaan dilanjutkan selama aktivitas berada di latar depan dan pengguna berinteraksi dengannya. Dari keadaan dilanjutkan, aktivitas bisa beralih ke keadaan dihentikan sementara.

Aktivitas dihentikan sementara (metode onPause())

```
@Override
protected void onPause() {
    super.onPause();
    // Another activity is taking focus
    // (this activity is about to be "paused").
}
```

Keadaan dihentikan sementara bisa terjadi dalam sejumlah situasi:

- Aktivitas akan masuk ke latar belakang, namun belum sepenuhnya berhenti. Ini merupakan indikasi pertama bahwa pengguna meninggalkan aktivitas Anda.
- Aktivitas hanya terlihat sebagian di layar, karena dialog atau aktivitas transparan lainnya dihamparkan di atasnya.

- Dalam mode multi-jendela atau layar terpisah (API 24), aktivitas ini ditampilkan pada layar, namun beberapa aktivitas lain memiliki fokus pengguna.

Sistem memanggil metode onPause() bila aktivitas beralih ke keadaan dihentikan sementara. Karena metode onPause() adalah indikasi pertama yang didapat bahwa pengguna mungkin meninggalkan aktivitas, Anda bisa menggunakan onPause() untuk menghentikan pemutaran animasi atau video, melepas sumber daya yang mengandalkan perangkat keras, atau mengikat perubahan aktivitas yang belum disimpan (seperti draf email).

Metode onPause() akan segera dijalankan. Jangan menggunakan onPause() untuk operasi yang banyak menggunakan CPU seperti menulis data persisten ke database. Aplikasi mungkin masih terlihat di layar saat diteruskan ke keadaan dihentikan sementara, dan penundaan apa pun dalam eksekusi onPause() bisa memperlambat transisi pengguna ke aktivitas berikutnya. Implementasikan operasi beban-berat bila aplikasi berada dalam keadaan dihentikan.

Perhatikan, dalam mode multi-jendela (API 24), aktivitas yang dihentikan sementara masih bisa terlihat pada layar. Dalam hal ini, Anda tidak perlu menghentikan sementara pemutaran animasi atau video seperti yang diinginkan untuk aktivitas yang terlihat sebagian. Anda bisa menggunakan metode inMultiWindowMode() di kelas Activity untuk menguji apakah aplikasi sedang berjalan dalam mode multi-jendela.

Aktivitas Anda bisa berpindah dari keadaan dihentikan sementara ke keadaan dilanjutkan (jika pengguna kembali ke aktivitas tersebut) atau ke keadaan dihentikan (jika pengguna meninggalkan aktivitas secara bersamaan).

Aktivitas dihentikan (metode onStop())

```
@Override
protected void onStop() {
    super.onStop();
    // The activity is no longer visible (it is now "stopped")
}
```

Aktivitas berada dalam keadaan dihentikan bila tidak lagi terlihat pada layar. Hal ini biasanya karena pengguna memulai aktivitas lain, atau kembali ke layar utama. Sistem mempertahankan instance aktivitas di back-stack, dan jika pengguna kembali ke aktivitas tersebut, maka akan dimulai lagi. Aktivitas yang dihentikan mungkin dimatikan secara bersamaan oleh sistem Android jika sumber daya minim.

Sistem akan memanggil metode onStop() bila aktivitas berhenti. Implementasikan metode OnStop() untuk menyimpan data persisten dan melepas sisa sumber daya yang belum dirilis di onPause(), termasuk operasi yang mungkin terlalu berat untuk onPause().

Aktivitas dimusnahkan (metode onDestroy())

```
@Override
protected void onDestroy() {
    super.onDestroy();
    // The activity is about to be destroyed.
}
```

Bila dimusnahkan, aktivitas akan dimatikan sepenuhnya, dan instance Aktivitas diklaim ulang oleh sistem. Hal ini bisa terjadi dalam sejumlah kasus:

- Anda memanggil finish() di aktivitas untuk mematikannya secara manual.
- Pengguna kembali ke aktivitas sebelumnya.
- Perangkat dalam situasi minim memori sehingga sistem mengklaim kembali aktivitas yang dihentikan untuk membebaskan lebih banyak sumber daya.
- Terjadi perubahan konfigurasi perangkat. Anda akan mengetahui selengkapnya tentang perubahan konfigurasi nanti dalam bab ini.

Gunakan onDestroy() untuk menghapus sepenuhnya setelah aktivitas Anda sehingga tidak ada komponen (seperti thread) yang berjalan setelah aktivitas ini dimusnahkan.

Perhatikan, ada kalanya sistem akan benar-benar mematikan proses hosting aktivitas tanpa memanggil metode ini (atau metode lainnya), sehingga Anda tidak boleh mengandalkan onDestroy() untuk menyimpan data atau keadaan aktivitas yang diperlukan. Sebagai gantinya, gunakan onPause() atau onStop().

Aktivitas dimulai kembali (metode onRestart())

```
@Override
protected void onRestart() {
    super.onRestart();
    // The activity is about to be restarted.
}
```

Keadaan dimulai kembali adalah keadaan sementara yang hanya terjadi jika aktivitas yang dihentikan dimulai kembali. Jika metode onRestart() dipanggil di antara onStop() dan onStart(). Jika memiliki sumber daya yang perlu dihentikan atau dimulai, Anda biasanya mengimplementasikan perilaku tersebut di OnStop() atau onStart(), bukan onRestart().

Perubahan konfigurasi dan keadaan aktivitas

Sebelumnya di bagian onDestroy() Anda telah mengetahui bahwa aktivitas mungkin dimusnahkan bila pengguna mengarah kembali, oleh Anda dengan metode finish(), atau oleh sistem bila perlu membebaskan sumber daya. Aktivitas Anda dimusnahkan keempat kalinya bila perangkat mengalami *perubahan konfigurasi*.

Perubahan konfigurasi terjadi di perangkat, dalam waktu proses, dan membatalkan validasi layout saat ini atau sumber daya lainnya di aktivitas Anda Bentuk yang paling umum dari perubahan konfigurasi adalah bila perangkat diputar. Bila perangkat diputar dari potret ke lanskap, atau sebaliknya, layout aplikasi Anda juga perlu diubah. Sistem membuat ulang aktivitas untuk membantu aktivitas tersebut beradaptasi dengan konfigurasi baru dengan memuat sumber daya alternatif (seperti layout khusus lanskap).

Perubahan konfigurasi lainnya bisa meliputi perubahan di lokal (pengguna memilih bahasa sistem yang berbeda), atau pengguna masuk ke mode multi-jendela (Android 7). Dalam mode multi-jendela, jika Anda telah mengonfigurasi aplikasi agar bisa diubah ukurannya, Android akan membuat ulang aktivitas tersebut untuk menggunakan definisi layout bagi ukuran aktivitas baru yang lebih kecil.

Bila terjadi perubahan konfigurasi, sistem Android akan menutup aktivitas Anda (memanggil onPause(), OnStop(), dan onDestroy()), kemudian memulainya dari awal (memanggil onCreate(), onStart(), dan onResume()).

Keadaan instance aktivitas

Bila aktivitas dimusnahkan dan dibuat ulang, ada implikasi untuk keadaan waktu proses aktivitas itu. Bila aktivitas dihentikan sementara atau dihentikan sepenuhnya, keadaan aktivitas akan dipertahankan karena aktivitas tersebut masih ditahan dalam memori. Bila aktivitas dibuat ulang, keadaan aktivitas dan kemajuan pengguna di aktivitas tersebut akan hilang, dengan pengecualian ini:

- Sebagian informasi keadaan aktivitas secara otomatis disimpan secara default. Keadaan tampilan di layout Anda dengan ID unik (seperti yang didefinisikan oleh atribut android:id di layout) disimpan dan dipulihkan bila aktivitas dibuat ulang. Dalam hal ini, nilai-nilai yang dimasukkan pengguna di tampilan EditText biasanya dipertahankan bila aktivitas tersebut dibuat ulang.
- Maksud yang digunakan untuk memulai aktivitas, dan informasi yang tersimpan di data atau ekstra maksud, tetap tersedia untuk aktivitas itu bila dibuat ulang.

Keadaan aktivitas disimpan sebagai rangkaian pasangan kunci/nilai di objek Bundle yang disebut *keadaan instance aktivitas*. Sistem akan menyimpan informasi keadaan default ke bundel keadaan instance tepat sebelum aktivitas dihentikan, dan meneruskan bundel itu ke instance aktivitas baru untuk dipulihkan.

Anda bisa menambahkan data instance sendiri ke bundel keadaan instance dengan mengganti callback onSaveInstanceState(). Bundel keadaan diteruskan ke metode onCreate(), sehingga Anda bisa memulihkan data keadaan instance tersebut bila aktivitas dibuat. Ada juga callback onRestoreInstanceState() yang bisa Anda gunakan untuk memulihkan data keadaan.

Karena rotasi perangkat adalah kasus penggunaan umum untuk aplikasi, pastikan Anda menguji bahwa aktivitas berperilaku dengan benar sebagai respons terhadap perubahan konfigurasi ini, dan implementasikan instance keadaan jika diperlukan.

Catatan: Keadaan instance aktivitas bersifat khusus untuk intance aktivitas tertentu, yang berjalan di satu tugas. Jika pengguna menghentikan aplikasi secara paksa, maka boot ulang perangkat, atau jika sistem Android menutup seluruh proses aplikasi untuk menjaga memori, keadaan instance aktivitas akan hilang. Untuk mempertahankan perubahan keadaan di seluruh instance aplikasi dan mem-boot ulang perangkat, Anda perlu menulis data itu ke preferensi bersama. Anda akan mengetahui selengkapnya tentang preferensi bersama dalam bab berikutnya.

Menyimpan keadaan instance aktivitas

Untuk menyimpan informasi ke bundel keadaan instance, gunakan callback onSaveInstanceState(). Ini bukan metode callback daur hidup, namun metode ini dipanggil bila pengguna meninggalkan aktivitas Anda (kadang-kadang sebelum metode OnStop()).

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    super.onSaveInstanceState(savedInstanceState);
    // save your state data to the instance state bundle
}
```

Di metode onSaveInstanceState() diteruskan objek Bundle (kumpulan pasangan kunci/nilai) bila dipanggil. Ini adalah bundel keadaan instance yang nanti Anda tambahkan informasi keadaan aktivitas sendiri.

Anda telah mempelajari tentang berbagai bundel di bab sebelumnya saat menambahkan kunci dan nilai untuk ekstra maksud. Tambahkan informasi ke bundel keadaan instance dengan cara yang sama, dengan kunci yang Anda definisikan dan beragam metode "put" yang didefinisikan dalam kelas Bundle:

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    super.onSaveInstanceState(savedInstanceState);

    // Save the user's current game state
    savedInstanceState.putInt("score", mCurrentScore);
    savedInstanceState.putInt("level", mCurrentLevel);
}
```

Jangan lupa memanggil melalui kelas super, untuk memastikan keadaan hierarki tampilan juga disimpan ke bundel.

Memulihkan keadaan instance aktivitas

Setelah menyimpan keadaan instance aktivitas, Anda juga perlu memulihkannya bila aktivitas tersebut dibuat ulang. Anda bisa melakukannya di salah satu dari dua tempat:

- Metode callback onCreate(), yang dipanggil bersama bundel keadaan instance bila aktivitas dibuat.
- Callback onRestoreInstanceState(), yang dipanggil setelah onStart(), setelah aktivitas dibuat.

Sering kali, tempat yang lebih baik untuk memulihkan keadaan aktivitas adalah di onCreate(), untuk memastikan antarmuka pengguna Anda yang menyertakan keadaan tersedia sesegera mungkin.

Untuk memulihkan keadaan instance yang disimpan di onCreate(), uji keberadaan bundel keadaan sebelum Anda mencoba mendapatkan datanya. Bila aktivitas Anda dimulai untuk pertama kali, keadaan tidak akan ada dan bundel akan nol.

```
@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("score");
        mCurrentLevel = savedInstanceState.getInt("level");
    } else {
        // Probably initialize members with default values for a new instance
    }
    ...
}
```

Praktik Terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Daur Hidup Aktivitas dan Keadaan Instance

Ketahui Selengkapnya

- Aktivitas (Panduan API)
- Aktivitas (Referensi API)
- Mengelola Daur Hidup Aktivitas
- Menghentikan Sementara dan Melanjutkan Aktivitas
- Menghentikan dan Memulai Ulang Aktivitas
- Membuat Ulang Aktivitas
- Menangani Perubahan Waktu Proses
- Bundel (Referensi API)

2.3: Aktivitas dan Maksud Implisit

Materi:

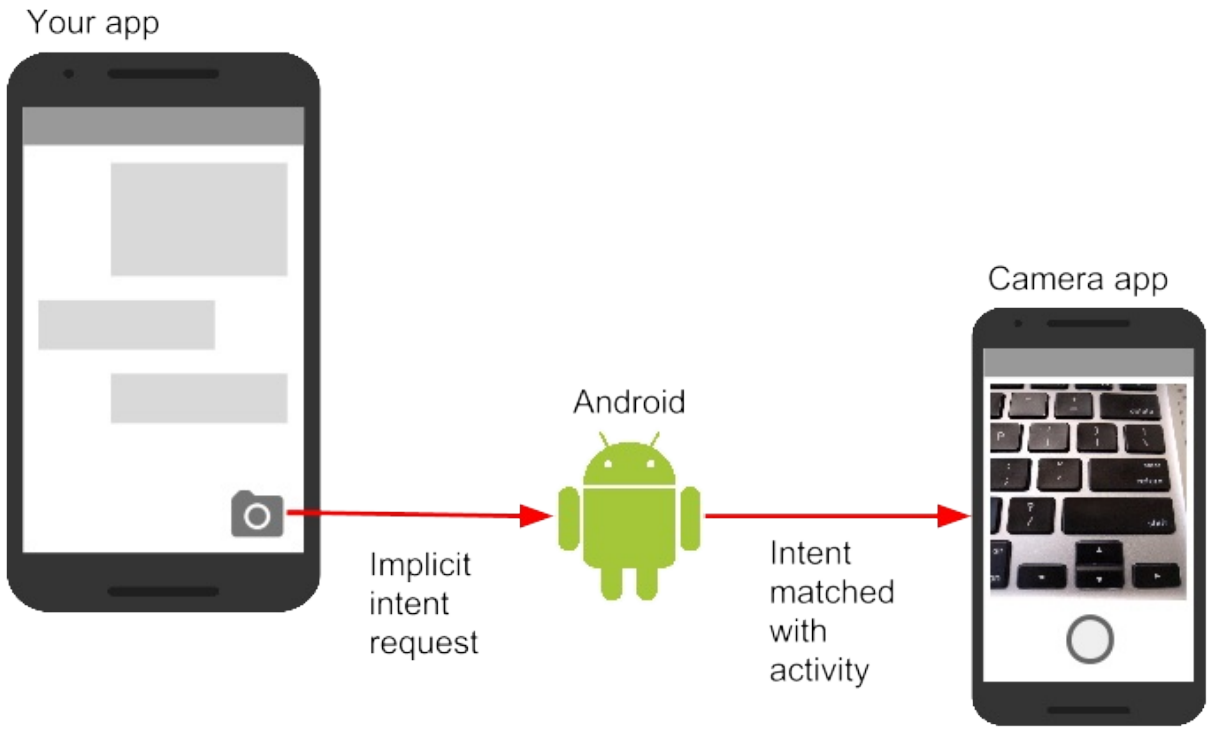
Di bab sebelumnya, Anda telah mempelajari tentang maksud, dan cara meluncurkan aktivitas khusus di aplikasi dengan *maksud eksplisit*. Di bab ini, Anda akan mempelajari cara mengirim dan menerima maksud *implisit*, tempat mendeklarasikan aksi umum yang dilakukan dalam maksud, dan sistem akan mencocokkan permintaan dengan aktivitas khusus. Selain itu, Anda akan mengetahui selengkapnya tentang *tugas* Android, dan cara mengonfigurasi aplikasi untuk mengaitkan aktivitas baru dengan tugas berbeda.

Tentang maksud implisit

Di bab awal, Anda telah mempelajari tentang maksud eksplisit, tempat Anda bisa memulai satu aktivitas dari aktivitas lain dengan menetapkan nama kelas aktivitas itu. Inilah cara paling dasar untuk menggunakan maksud, untuk memulai aktivitas atau komponen aplikasi lainnya dan meneruskan data ke maksud (dan kadang-kadang meneruskan data kembali).

Penggunaan maksud yang lebih fleksibel adalah *maksud implisit*. Dengan maksud implisit, Anda tidak menetapkan aktivitas yang tepat (atau komponen lainnya) untuk dijalankan—sebagai gantinya, Anda menyertakan informasi secukupnya dalam maksud tentang tugas yang ingin dilakukan. Sistem Android akan mencocokkan informasi dalam maksud permintaan dengan aktivitas yang tersedia di perangkat yang bisa melakukan tugas itu. Jika hanya ada satu

aktivitas yang cocok, aktivitas itu akan diluncurkan. Jika ada beberapa aktivitas yang cocok, pengguna disediakan pemilih aplikasi yang memungkinkan mereka memilih aplikasi yang disukai untuk melakukan tugas.



Misalnya, Anda memiliki aplikasi yang mencantumkan cuplikan video yang tersedia. Jika pengguna menyentuh item dalam daftar, Anda akan memutar cuplikan video itu. Daripada mengimplementasikan keseluruhan pemutar video di aplikasi, Anda bisa meluncurkan maksud yang menetapkan tugas seperti "putar objek video tipe." Sistem Android selanjutnya akan mencocokkan permintaan dengan aktivitas yang telah mendaftarkan dirinya sendiri untuk memutar objek video tipe.

Aktivitas mendaftarkan dirinya sendiri secara otomatis pada sistem agar dapat menangani maksud implisit dengan *filter* maksud, yang dideklarasikan dalam manifes Android. Misalnya, aktivitas utama (dan satu-satunya aktivitas utama) untuk aplikasi Anda memiliki filter maksud yang mendeklarasikan aktivitas utama untuk kategori peluncur. Filter maksud ini adalah cara sistem Android mengetahui cara memulai aktivitas tertentu di aplikasi Anda bila pengguna mengetuk ikon untuk aplikasi di layar utama perangkat.

Aksi maksud, kategori, dan data

Maksud implisit, seperti maksud eksplisit, merupakan instance kelas Intent. Selain bagian maksud yang telah Anda pelajari di bab awal (seperti data maksud dan ekstra maksud), bidang-bidang ini digunakan oleh maksud implisit:

- *Aksi* maksud, merupakan aksi umum yang harus dilakukan aktivitas penerima. Aksi maksud yang tersedia didefinisikan sebagai konstanta dalam kelas Intent dan dimulai dengan kata ACTION_. Aksi maksud umum adalah ACTION_VIEW, yang Anda gunakan bila memiliki beberapa informasi yang bisa ditampilkan aktivitas kepada pengguna, seperti foto untuk ditampilkan di aplikasi galeri, atau alamat untuk ditampilkan di aplikasi peta. Anda bisa menetapkan aksi untuk sebuah maksud di konstruktor maksud, atau dengan metode setAction().
- *Kategori* maksud, menyediakan informasi tambahan tentang kategori komponen yang harus menangani maksud. Kategori maksud bersifat opsional, dan Anda bisa menambahkan lebih dari satu kategori ke maksud. Kategori maksud juga didefinisikan sebagai konstanta di kelas Intent dan dimulai dengan kata CATEGORY_. Anda bisa menambahkan kategori ke maksud dengan metode addCategory().
- *Tipe* data, yang menunjukkan MIME tipe data yang digunakan untuk mengoperasikan aktivitas. Biasanya, ini diambil dari URI dalam bidang data maksud, namun Anda juga bisa secara eksplisit mendefinisikan tipe data dengan metode setType().

Tindakan, kategori, dan tipe data maksud keduanya digunakan oleh objek Maksud yang Anda buat di aktivitas pengiriman, serta di filter maksud yang didefinisikan di manifes Android untuk aktivitas penerimaan. Sistem Android menggunakan informasi ini untuk mencocokkan permintaan maksud implisit dengan aktivitas atau komponen lain yang bisa menangani maksud itu.

Mengirim maksud implisit

Memulai aktivitas bersama maksud implisit, dan meneruskan data di antara aktivitas itu, cara kerjanya sangat mirip dengan yang dilakukan untuk maksud eksplisit:

1. Di aktivitas pengirim, buat objek Maksud baru.
2. Tambahkan informasi tentang permintaan ke objek Intent, seperti data atau ekstra.
3. Kirim maksud bersama startActivity() (untuk memulai aktivitas saja) atau startActivityforResult() (untuk memulai aktivitas dan mengharapkan hasil kembali).

Bila membuat objek Intent implisit, Anda:

- Jangan menetapkan aktivitas khusus atau komponen lain untuk diluncurkan.
- Menambahkan aksi maksud atau kategori maksud (atau keduanya).
- Cocokkan maksud dengan sistem sebelum memanggil startActivity() atau startActivityforResult().
- Menampilkan pemilih aplikasi untuk permintaan (opsional).

Buat objek Intent implisit

Untuk menggunakan maksud implisit, buat objek Intent seperti yang dilakukan untuk maksud eksplisit, hanya tanpa nama komponen spesifik.

```
Intent sendIntent = new Intent();
```

Anda juga bisa membuat objek Intent dengan aksi tertentu:

```
Intent sendIntent = new Intent(Intent.ACTION_VIEW);
```

Setelah memiliki objek Intent, Anda bisa menambahkan informasi lain (kategori, data, ekstra) dengan beragam metode Intent. Misalnya, kode ini membuat objek Intent implisit, menyetel aksi maksud ke ACTION_SEND, mendefinisikan ekstra maksud untuk menampung teks, dan menyetel tipe data ke tipe MIME "text/plain".

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");
```

Cocokkan aktivitas sebelumnya memulainya

Bila Anda mendefinisikan maksud implisit dengan aksi dan/atau kategori tertentu, ada kemungkinan bahwa tidak akan ada aktivitas *apa pun* di perangkat yang bisa menangani permintaan Anda. Jika Anda hanya mengirim maksud dan tidak ada kecocokan, aplikasi akan mogok.

Untuk memverifikasi apakah aktivitas atau komponen lain tersedia untuk menerima maksud Anda, gunakan metode resolveActivity() bersama pengelola paket sistem seperti ini:

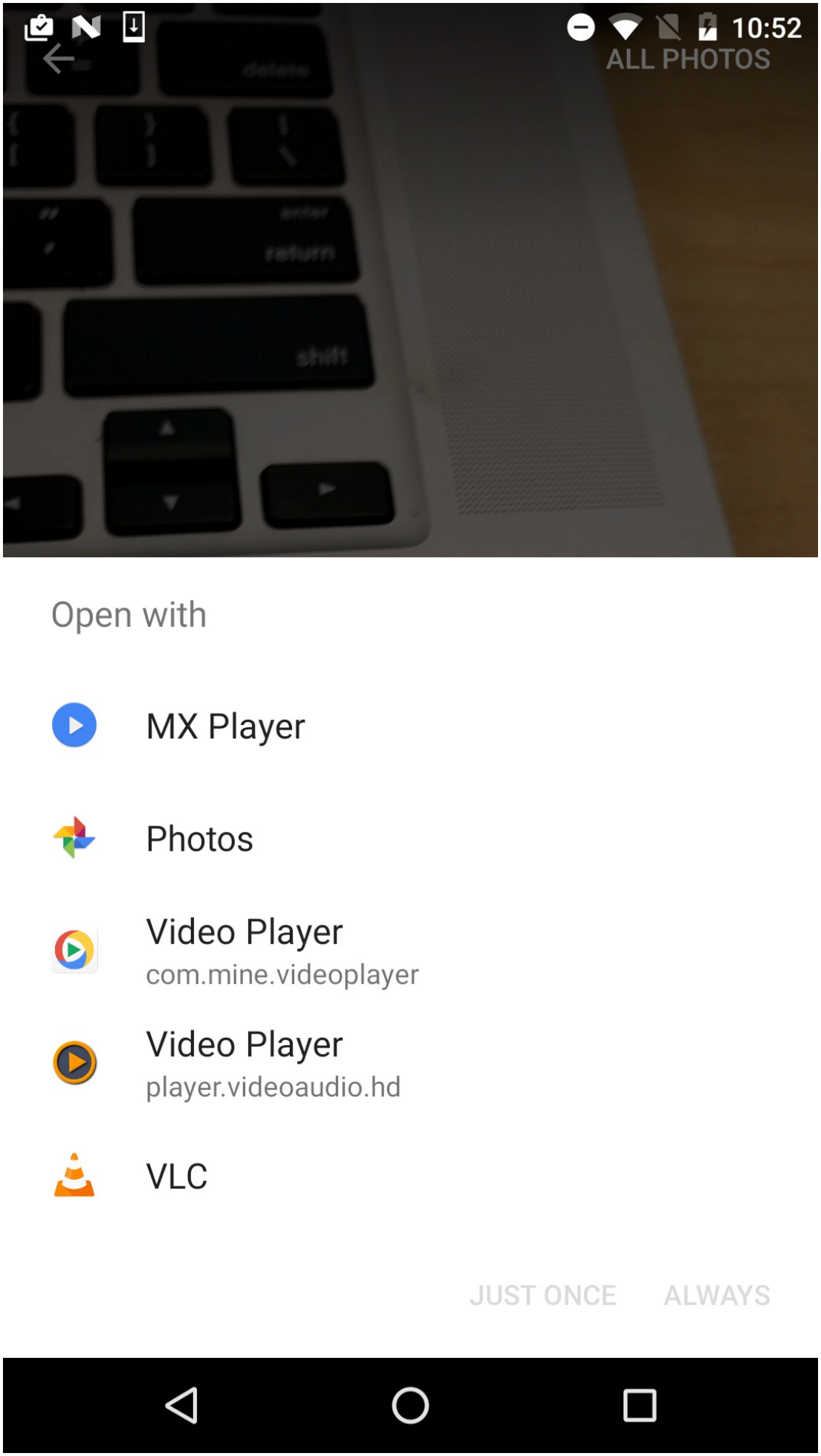
```
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(chooser);
}
```

Jika hasil resolveActivity() bukan nol, maka setidaknya ada satu aplikasi yang tersedia yang bisa menangani maksud, dan aman untuk menggunakan startActivity(). Jangan mengirim maksud jika hasilnya nol.

Jika Anda memiliki fitur yang bergantung pada aktivitas eksternal yang mungkin atau mungkin tidak tersedia di perangkat, praktik terbaik adalah menguji ketersediaan aktivitas eksternal sebelum pengguna mencoba menggunakannya. Jika tidak ada aktivitas yang bisa menangani permintaan Anda (yakni, resolveActivity() mengembalikan nol), nonaktifkan fitur atau sediakan pesan kesalahan untuk fitur tersebut kepada pengguna.

Tampilkan pemilih aplikasi

Untuk menemukan aktivitas atau komponen lain yang bisa menangani permintaan maksud Anda, sistem Android akan mencocokkan maksud implisit dengan aktivitas yang filter maksudnya menunjukkannya bisa melakukan aksi tersebut. Jika beberapa aplikasi yang telah dipasang memang cocok, pengguna akan disajikan pemilih aplikasi yang memungkinkan mereka memilih aplikasi yang ingin digunakan untuk menangani maksud tersebut.


10:52
ALL PHOTOS
Open with
MX Player
Photos
Video Player
com.mine.videoplayer
Video Player
player.videoaudio.hd
VLC
JUST ONCE
ALWAYS

Dalam banyak kasus, pengguna memiliki aplikasi pilihan untuk tugas yang diberikan, dan mereka akan memilih opsi untuk selalu menggunakan aplikasi itu bagi tugas tersebut. Akan tetapi, jika beberapa aplikasi bisa merespons maksud dan pengguna setiap kali mungkin ingin menggunakan aplikasi yang berbeda, Anda bisa memilih untuk secara eksplisit menampilkan dialog pemilih kapan saja. Misalnya, bila aplikasi melakukan aksi "bagikan ini" bersama aksi ACTION_SEND, pengguna mungkin ingin berbagi menggunakan aplikasi yang berbeda, bergantung pada situasi saat ini.

Untuk menampilkan pemilih, buat maksud wrapper untuk maksud implisit dengan metode createChooser(), kemudian cocokkan dan gunakan startActivity() dengan maksud wrapper. Metode createChooser() juga memerlukan argumen string untuk judul yang muncul pada pemilih. Anda bisa menetapkan judul bersama sumber daya string seperti yang dilakukan pada string lain.

Misalnya:

```
// The implicit Intent object
Intent sendIntent = new Intent(Intent.ACTION_SEND);
// Always use string resources for UI text.
String title = getResources().getString(R.string.chooser_title);
// Create the wrapper intent to show the chooser dialog.
Intent chooser = Intent.createChooser(sendIntent, title);
// Resolve the intent before starting the activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(chooser);
}
```

Menerima maksud implisit

Jika ingin aktivitas di aplikasi merespons maksud implisit (dari aplikasi milik Anda atau aplikasi lain), deklarasikan satu atau beberapa filter maksud di manifes Android. Setiap filter maksud menetapkan tipe maksud yang diterimanya berdasarkan aksi, data, dan kategori maksud. Sistem akan mengirimkan maksud implisit ke komponen aplikasi Anda hanya jika maksud itu bisa melewati salah satu filter maksud.

Catatan: Maksud eksplisit selalu dikirimkan ke targetnya, terlepas dari filter maksud yang dideklarasikan komponen. Sebaliknya, jika aktivitas tidak mendeklarasikan filter maksud apa pun, aktivitas hanya bisa diluncurkan dengan maksud eksplisit.
Setelah aktivitas berhasil diluncurkan bersama maksud implisit, Anda bisa menangani maksud tersebut dan datanya dengan cara sama yang dilakukan pada maksud eksplisit:

1. Mendapatkan objek Intent dengan getIntent().
2. Mendapatkan data maksud atau ekstra dari maksud itu.
3. Melakukan tugas yang diminta maksud.
4. Mengembalikan data ke aktivitas pemanggil bersama maksud lain, jika diperlukan.

Filter maksud

Definisikan filter maksud dengan satu atau beberapa elemen `<intent-filter>` di file manifes aplikasi, yang disarangkan di elemen `<activity>` yang sesuai. Di dalam `<intent-filter>` , tetapkan tipe maksud yang bisa ditangani aktivitas Anda. Sistem Android mencocokkan maksud implisit dengan aktivitas atau komponen aplikasi lain hanya jika bidang-bidang dalam objek Intent cocok dengan filter maksud untuk komponen itu.

Filter maksud dapat berisi elemen ini, sesuai dengan bidang di objek Intent yang dijelaskan di atas:

- `<action>` : Aksi maksud.
- `<data>` : Tipe data yang diterima, termasuk tipe MIME atau atribut URI data atribut lainnya (seperti skema, host, porta, jalur, dan sebagainya).
- `<category>` : Kategori maksud.

Misalnya, aktivitas utama untuk aplikasi Anda menyertakan elemen `<intent-filter>` ini, yang telah Anda lihat di bab sebelumnya:

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Filter maksud ini memiliki aksi MAIN dan kategori LAUNCHER. Elemen `<action>` menetapkan bahwa ini adalah titik masuk "utama" ke aplikasi. Elemen `<category>` menetapkan bahwa aktivitas ini harus dicantumkan dalam peluncur aplikasi sistem (untuk memungkinkan pengguna meluncurkan aktivitas ini). Hanya aktivitas utama untuk aplikasi Anda yang harus memiliki filter maksud ini.

Inilah contoh lain maksud implisit untuk berbagi sedikit teks. Filter maksud ini mencocokkan contoh maksud implisit dari bagian sebelumnya:

```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

Anda bisa menetapkan lebih dari satu aksi, data, atau kategori untuk filter maksud yang sama, atau memiliki beberapa filter maksud per aktivitas untuk menangani jenis maksud yang berbeda.

Sistem Android akan menguji maksud implisit terhadap filter maksud dengan membandingkan bagian maksud tersebut dengan ketiga elemen filter maksud (aksi, kategori, dan data). Maksud harus lulus ketiga pengujian atau sistem Android tidak akan mengirim maksud ke komponen. Akan tetapi, karena sebuah komponen mungkin memiliki beberapa filter maksud, maksud yang tidak lulus salah satu filter komponen mungkin saja lulus di filter lainnya.

Tindakan

Filter maksud bisa mendeklarasikan nol atau beberapa elemen `<action>` untuk aksi maksud. Aksi didefinisikan dalam atribut nama, dan berisi string "android.intent.action." ditambah nama aksi maksud, tanpa awalan ACTION_. Jadi, misalnya, maksud implisit dengan aksi ACTION_VIEW cocok dengan filter maksud yang aksinya adalah `android.intent.action.VIEW.`

Misalnya, filter maksud ini cocok dengan ACTION_EDIT dan ACTION_VIEW:

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

Untuk melewati filter ini, aksi yang ditetapkan dalam objek Intent yang masuk harus cocok dengan setidaknya salah satu tindakan. Anda harus menyertakan setidaknya satu aksi maksud untuk mencocokkan maksud implisit yang masuk.

Kategori

Filter maksud bisa mendeklarasikan nol atau beberapa elemen `<category>` untuk kategori maksud. Kategori didefinisikan dalam atribut nama, dan berisi string "android.intent.category." ditambah nama kategori maksud, tanpa awalan CATEGORY.

Misalnya, filter maksud ini cocok dengan CATEGORY_DEFAULT dan CATEGORY_BROWSABLE:

```
<intent-filter>
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    ...
</intent-filter>
```

Perhatikan, semua aktivitas yang Anda inginkan untuk menerima maksud implisit harus menyertakan filter maksud android.intent.category.DEFAULT. Kategori ini diterapkan ke semua objek Intent implisit oleh sistem Android.

Data

Filter maksud bisa mendeklarasikan nol atau beberapa elemen `<data>` untuk URI yang dimuat dalam data maksud. Karena data maksud terdiri dari URI dan (opsional) tipe MIME, Anda bisa membuat filter maksud untuk beragam aspek data itu, termasuk:

- Skema URI
- Host URI
- Jalur URI
- Tipe MIME

Misalnya, filter maksud ini mencocokkan maksud data dengan skema URI http dan tipe MIME "video/mpeg" atau "audio/mpeg".

```
<intent-filter>
    <data android:mimeType="video/mpeg" android:scheme="http" />
    <data android:mimeType="audio/mpeg" android:scheme="http" />
    ...
</intent-filter>
```

Berbagi data dengan ShareCompat.IntentBuilder

Tindakan berbagi adalah cara mudah bagi pengguna untuk berbagi item di aplikasi dengan jaringan sosial dan aplikasi lainnya. Meskipun Anda bisa membangun aksi berbagi di aplikasi sendiri menggunakan maksud implisit dengan aksi ACTION_SEND, Android menyediakan kelas helper ShareCompat.IntentBuilder agar mudah mengimplementasikan berbagi di aplikasi.

Catatan: Untuk aplikasi yang menargetkan rilis Android setelah API 14, Anda bisa menggunakan kelas ShareActionProvider untuk tindakan berbagi sebagai ganti ShareCompat.IntentBuilder. Kelas ShareCompat adalah bagian dari pustaka dukungan V4, dan memungkinkan Anda untuk menyediakan tindakan berbagi di aplikasi yang kompatibel-mundur. ShareCompat menyediakan API tunggal untuk berbagi pada perangkat Android lama dan baru. Anda mengetahui selengkapnya tentang pustaka dukungan Android dalam bab berikutnya.
Dengan kelas ShareCompat.IntentBuilder, Anda tidak perlu membuat atau mengirim maksud implisit untuk aksi berbagi. Gunakan metode di ShareCompat.IntentBuilder untuk menunjukkan data yang ingin dibagikan serta informasi tambahan. Mulai dengan metode from() untuk membuat builder maksud baru, tambahkan metode lain untuk menambahkan lebih banyak data, dan akhiri dengan metode startChooser() untuk membuat dan mengirim maksud. Anda bisa merangkai metode seperti ini:

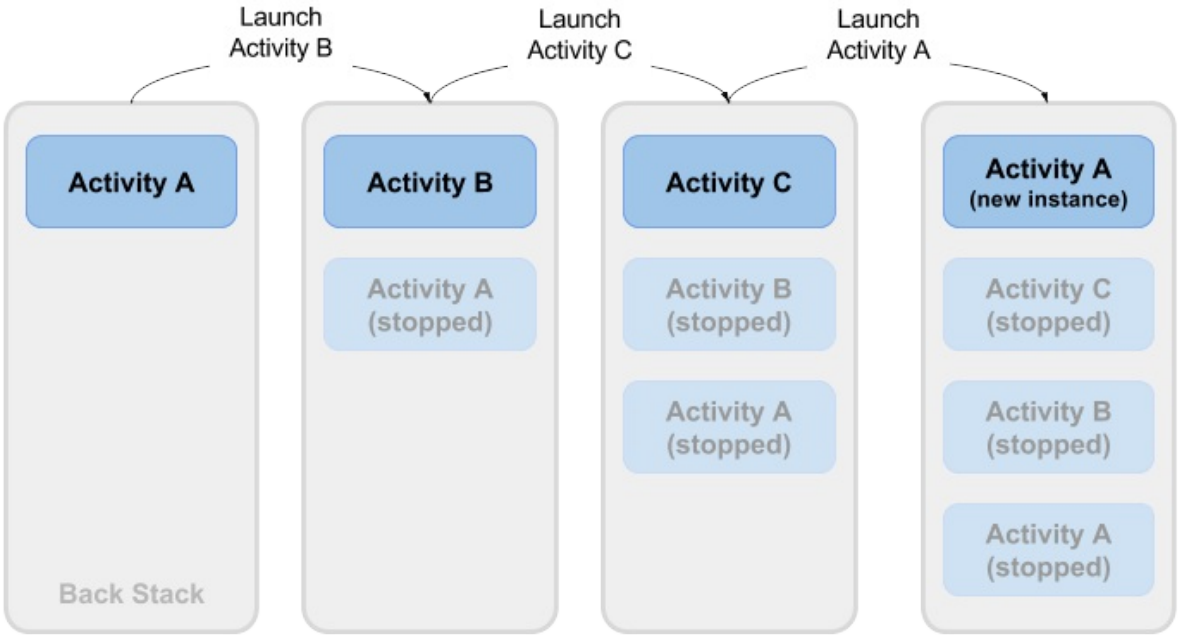
```
ShareCompat.IntentBuilder
    .from(this)         // information about the calling activity
    .setType(mimeType)  // mime type for the data
    .setChooserTitle("Share this text with: ") //title for the app chooser
    .setText(txt)       // intent data
    .startChooser();    // send the intent
```

Mengelola tugas dan aktivitas

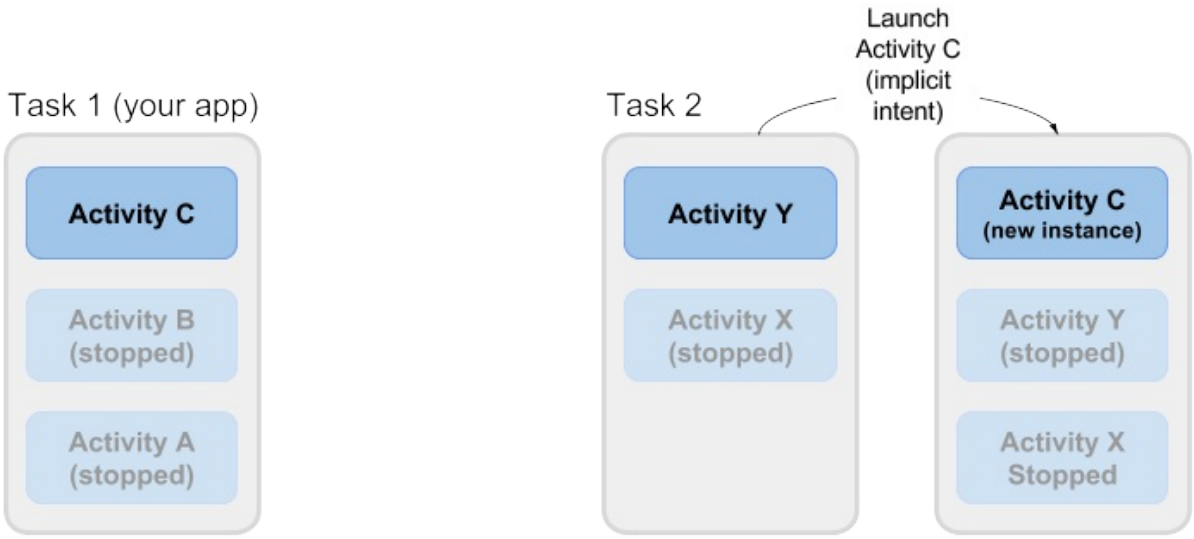
Di bab sebelumnya, Anda telah mempelajari tentang tugas dan back-stack, yakni tempat tugas untuk aplikasi yang berisi tumpukannya sendiri bagi aktivitas yang telah dikunjungi pengguna saat menggunakan aplikasi Anda. Saat pengguna menyusuri aplikasi, instance aktivitas tugas itu akan didorong dan dimunculkan dari tumpukan untuk tugas itu.

Umumnya navigasi pengguna dari aktivitas ke aktivitas dan kembali lagi melalui tumpukan secara langsung. Bergantung pada desain dan navigasi aplikasi Anda, mungkin ada komplikasi, khususnya dengan aktivitas yang dimulai dari aplikasi lain dan tugas lain.

Misalnya, katakanlah Anda memiliki aplikasi dengan tiga aktivitas: A, B, dan C. A meluncurkan B bersama sebuah maksud, dan B meluncurkan C. C, sebaliknya mengirimkan maksud untuk meluncurkan A. Dalam hal ini, sistem membuat *instance kedua* dari A di bagian atas tumpukan, dibandingkan memberikan instance yang sudah berjalan ke latar depan. Bergantung pada cara mengimplementasikan aktivitas, dua instance dari A bisa menjadi tidak sinkron dan memberikan pengalaman membingungkan pada pengguna saat menyusuri kembali melalui tumpukan.



Atau, katakanlah aktivitas C Anda bisa diluncurkan dari aplikasi kedua bersama sebuah maksud implisit. Pengguna menjalankan aplikasi kedua, yang memiliki tugas dan back-stack sendiri. Jika aplikasi itu menggunakan maksud implisit untuk meluncurkan aktivitas C Anda, instance C akan baru dibuat dan ditempatkan pada back-stack untuk tugas aplikasi kedua itu. Aplikasi Anda tetap memiliki tugas, back-stack, dan instance C sendiri.



Kebanyakan perilaku default Android untuk tugas dan aktivitas berfungsi dengan baik dan Anda tidak perlu khawatir tentang cara aktivitas dikaitkan dengan tugas atau bagaimana aktivitas bisa ada di back-stack. Jika Anda ingin mengubah perilaku normal, Android menyediakan sejumlah cara untuk mengelola tugas dan aktivitas dalam tugas tersebut, termasuk:

- Mode peluncuran aktivitas, untuk menentukan cara meluncurkan aplikasi.
- Afinitas tugas, yang menunjukkan tugas mana yang memiliki aktivitas yang diluncurkan.

Mode Peluncuran Aktivitas

Gunakan mode peluncuran aktivitas untuk menunjukkan cara memperlakukan aktivitas baru bila diluncurkan—yakni jika aktivitas harus ditambahkan ke tugas saat ini, atau diluncurkan ke dalam tugas baru. Definisikan mode peluncuran untuk aktivitas bersama atribut di elemen `<activity>` manifes Android, atau bersama flag yang disetel pada maksud yang memulai aktivitas itu.

Atribut aktivitas

Untuk mendefinisikan mode peluncuran aktivitas, tambahkan atribut android:launchMode ke elemen `<activity>` di manifes Android. Contoh ini menggunakan mode peluncuran "standar", yang merupakan default.

```
<activity
   android:name=".SecondActivity"
   android:label="@string/activity2_name"
   android:parentActivityName=".MainActivity"
   android:launchMode="standard">
   ...
</activity>
```

Ada empat mode peluncuran yang tersedia sebagai bagian dari elemen `<activity>` :

- "standard" (default): Aktivitas baru diluncurkan dan ditambahkan ke back-stack untuk tugas saat ini. Aktivitas bisa dibuat instance-nya beberapa kali, tugas tunggal bisa memiliki beberapa instance aktivitas yang sama, dan beberapa instance bisa dimiliki oleh tugas yang berbeda.
- "singleTop": Jika instance aktivitas ada di bagian atas back-stack tugas saat ini dan permintaan maksud untuk aktivitas tersebut datang, Android akan merutekan maksud tersebut ke instance aktivitas yang ada, bukan membuat instance baru. Aktivitas baru tetap dibuat instance-nya jika masih ada aktivitas di back-stack selain di bagian atas.
- "singleTask": Bila aktivitas diluncurkan, sistem akan membuat tugas baru untuk aktivitas itu. Jika tugas lain sudah ada dengan instance aktivitas itu, sistem akan merutekan maksud ke aktivitas itu sebagai gantinya.
- "singleInstance": Sama seperti tugas tunggal, hanya saja sistem tidak meluncurkan aktivitas lain ke dalam tugas yang berisi instance aktivitas. Aktivitas akan selalu tunggal dan satu-satunya anggota tugasnya.

Sebagian besar aplikasi hanya akan menggunakan mode peluncuran standar atau singleTop. Lihat dokumentasi atribut launchMode untuk informasi lebih detail tentang mode peluncuran.

Flag maksud

Flag maksud merupakan opsi yang menetapkan cara aktivitas (atau komponen aplikasi lainnya) yang menerima maksud harus menangani maksud tersebut. Flag maksud didefinisikan sebagai konstanta di kelas Intent dan dimulai dengan kata FLAG_. Anda menambahkan flag maksud ke objek Intent dengan setFlag() atau addFlag().

Tiga flag maksud spesifik digunakan untuk mengontrol mode peluncuran aktivitas, baik yang berhubungan dengan atribut launchMode atau yang menggantikannya. Flag maksud selalu diutamakan daripada mode peluncuran jika terjadi konflik.

- FLAG_ACTIVITY_NEW_TASK: memulai aktivitas di tugas baru. Ini adalah perilaku yang sama dengan mode peluncuran singleTask.
- FLAG_ACTIVITY_SINGLE_TOP: jika aktivitas yang diluncurkan berada di bagian atas back-stack, rutekan maksud ke instance aktivitas yang ada tersebut. Jika tidak, buat instance aktivitas baru. Ini adalah perilaku yang sama dengan mode peluncuran singleTop.

- FLAG_ACTIVITY_CLEAR_TOP: Jika instance aktivitas yang akan diluncurkan sudah ada di back-stack, musnahkan aktivitas lain di atasnya dan rutekan maksud ke instance yang ada. Bila digunakan bersama-sama dengan FLAG_ACTIVITY_NEW_TASK, flag ini berada di instance aktivitas yang ada dalam tugas apa pun dan membawanya ke latar depan.

Lihat kelas Intent untuk informasi selengkapnya tentang flag maksud yang tersedia.

Tangani maksud baru

Bila sistem Android merutekan maksud ke instance aktivitas yang ada, sistem akan memanggil metode callback onNewIntent() (biasanya persis sebelum metode onResume()). Metode onNewIntent() menyertakan argumen untuk maksud baru yang dirutekan ke aktivitas. Ganti metode onNewIntent() dalam kelas Anda untuk menangani informasi dari maksud baru tersebut.

Perhatikan, metode getIntent()—untuk mendapatkan akses ke maksud yag meluncurkan aktivitas—**selalu** mempertahankan maksud asal yang meluncurkan instance aktivitas. Panggil setIntent() dalam metode onNewIntent():

```
@Override
public void onNewIntent(Intent intent) {
    super.onNewIntent(intent);

    // Use the new intent, not the original one
    setIntent(intent);
}
```

Panggilan ke getIntent() setelah ini akan mengembalikan maksud baru.

Afinitas Tugas

Afinitas tugas menunjukkan tugas yang sebaiknya memiliki aktivitas pilihan bila instance aktivitas itu diluncurkan. Secara default, semua aktivitas dimiliki aplikasi yang meluncurkannya. Aktivitas dari luar aplikasi yang diluncurkan bersama maksud implisit dimiliki aplikasi yang mengirim maksud implisit.

Untuk mendefinisikan afinitas tugas, tambahkan atribut android:taskAffinity ke elemen `<activity>` di manifes Android. Afinitas tugas default adalah nama paket untuk aplikasi (yang dideklarasikan di . Nama tugas baru harus unik dan berbeda dari nama paket. Contoh ini menggunakan "com.example.android.myapp.newtask" untuk nama afinitas.

```
<activity
   android:name=".SecondActivity"
   android:label="@string/activity2_name"
   android:parentActivityName=".MainActivity"
   android:taskAffinity="com.example.android.myapp.newtask">
   ...
</activity>
```

Afinitas tugas biasanya digunakan bersama-sama dengan mode peluncuran singleTask atau flag maksud FLAG_ACTIVITY_NEW_TASK untuk menempatkan aktivitas baru dalam tugas yang diberi nama sendiri. Jika tugas baru sudah ada, maksud akan dirutekan ke tugas dan afinitas itu.

Penggunaan afinitas tugas lainnya adalah pengindukan ulang, yang memungkinkan tugas dipindahkan dari aktivitas yang semula meluncurkannya ke aktivitas yang menjadi afinitasnya. Untuk mengaktifkan pengindukan ulang atas tugas, tambahkan atribut afinitas tugas ke elemen `<activity>` dan setel android:allowTaskReparenting ke true.

```
<activity
   android:name=".SecondActivity"
   android:label="@string/activity2_name"
   android:parentActivityName=".MainActivity"
   android:taskAffinity="com.example.android.myapp.newtask"
   android:allowTaskReparenting="true" >
   ...
</activity>
```

Praktik Terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Aktivitas dan Kejadian Implisit

Ketahui Selengkapnya

- Aktivitas (Panduan API)
- Aktivitas (Referensi API)
- Maksud dan Filter Maksud
- Maksud (Referensi API)
- `<intent-filter>`
- Mengizinkan Aplikasi Lain untuk Memulai Aktivitas Anda
- ShareCompat.IntentBuilder (Referensi API)
- Uri (Referensi API)
- Maksud Google Maps
- Tugas dan Back-Stack
- `<activity>`
- Memanipulasi tugas dan back-stack Android
- Penjelasan afinitas tugas Android
- Memahami launchMode Aktivitas Android

3.1: Debugger Android Studio

Materi:

- Pengantar
- Tentang men-debug
- Menjalankan debugger
- Menggunakan Breakpoint
- Merunut kode
- Menampilkan bingkai tumpukan eksekusi
- Memeriksa dan memodifikasi variabel-variabel
- Menyetel watches
- Mengevaluasi ekspresi
- Alat (bantu) lainnya untuk men-debug
- Perekaman pelacakan ke log dan manifes Android
- Praktik terkait
- Ketahui selengkapnya

Dalam bab ini Anda akan mempelajari tentang men-debug aplikasi di Android Studio.

Tentang men-debug

Men-debug adalah proses menemukan dan memperbaiki kesalahan (bug) atau perilaku tak terduga dalam kode Anda. Semua kode memiliki bug, mulai dari perilaku yang salah dalam aplikasi, perilaku yang mengonsumsi memori atau sumber daya jaringan secara berlebihan, hingga aplikasi yang membeku atau mogok.

Bug bisa terjadi karena berbagai alasan:

- Kesalahan dalam desain atau implementasi Anda.
- Batasan (bug) kerangka kerja Android.
- Tidak adanya persyaratan atau asumsi tentang cara kerja aplikasi yang seharusnya.
- Batasan (atau bug) perangkat

Gunakan kemampuan men-debug, menguji, dan membuat profil di Android Studio untuk membantu Anda mengulangi kembali, menemukan, dan mengatasi semua masalah ini. Kemampuan itu antara lain:

- Monitor Android (logcat)
- Debugger Android Studio
- Kerangka kerja pengujian seperti JUnit atau Espresso
- Dalvik Debug Monitor Server (DDMS), untuk melacak penggunaan sumber daya

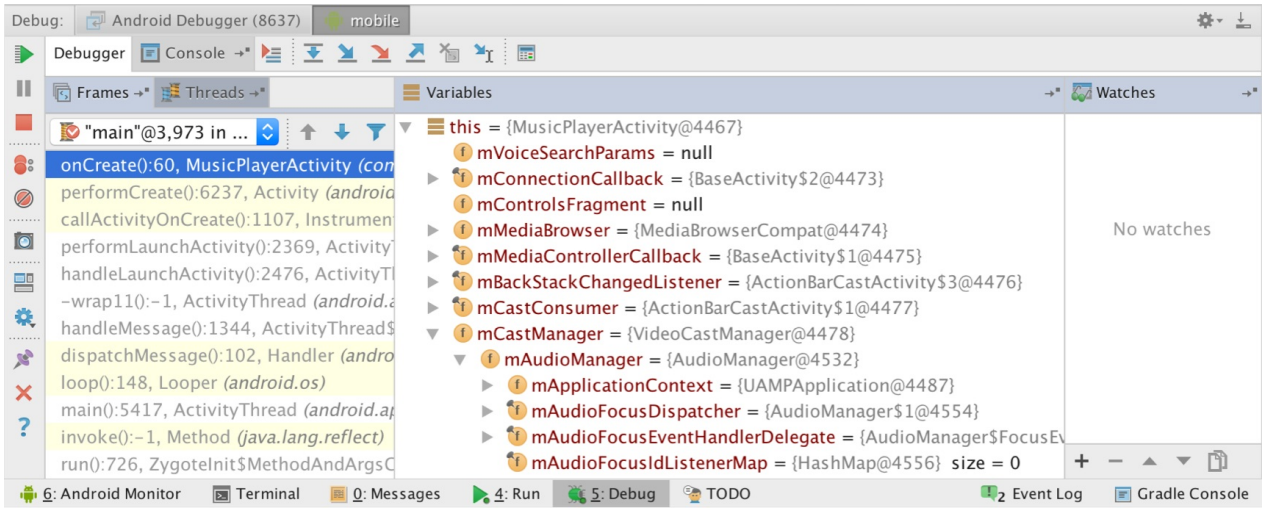
Dalam bab ini Anda akan mempelajari cara men-debug aplikasi dengan debugger Android Studio, menyetel dan menampilkan breakpoint, merunut kode, dan memeriksa variabel-variabel.

Menjalankan debugger

Menjalankan aplikasi dalam mode debug sama seperti menjalankan aplikasi seperti biasanya. Anda bisa menjalankan aplikasi dalam mode debug, atau melampirkan debugger ke aplikasi yang sudah berjalan.

Menjalankan aplikasi Anda dalam mode debug

Untuk mulai men-debug, klik Debug  di bilah alat. Android Studio akan membangun APK, menandainya dengan kunci debug, memasangnya pada perangkat yang Anda pilih, kemudian menjalankannya dan membuka jendela Debug.



Men-debug aplikasi yang berjalan

Jika aplikasi Anda sudah berjalan pada perangkat atau emulator, mulailah men-debug aplikasi itu dengan langkah-langkah ini:

1. Pilih **Run > Attach debugger to Android process** atau klik ikon **Attach**  di bilah alat.
2. Dalam dialog **Choose Process**, pilih proses yang ingin Anda lampirkan debugger.

   Secara default, debugger menampilkan perangkat dan proses aplikasi untuk proyek saat ini, serta perangkat keras atau perangkat virtual yang terhubung pada komputer Anda. Pilih Show all processes untuk menampilkan semua proses di semua perangkat.

3. Klik **OK**. Jendela Debug akan muncul seperti sebelumnya.

Melanjutkan atau Menghentikan Debug

Untuk melanjutkan eksekusi aplikasi setelah men-debug, pilih **Run > Resume Program** atau klik ikon Resume  .

Untuk berhenti men-debug aplikasi Anda, pilih **Run > Stop** atau klik ikon Stop  di bilah alat.

Menggunakan Breakpoint

Breakpoint adalah tempat dalam kode yang Anda inginkan untuk menghentikan sementara eksekusi normal aplikasi untuk melakukan tindakan lainnya seperti memeriksa variabel-variabel atau mengevaluasi ekspresi, atau mengeksekusi kode setiap baris untuk menentukan penyebab kesalahan waktu proses.

Menambahkan breakpoint

Untuk menambahkan breakpoint ke sebuah baris dalam kode Anda, gunakan langkah-langkah ini:

1. Carilah baris kode yang diinginkan untuk menjadi tempat Anda menghentikan sementara eksekusi.
2. Klik di gutter kiri jendela editor pada baris tersebut, di sebelah nomor baris. Titik merah muncul pada baris itu, yang menunjukkan sebuah breakpoint.

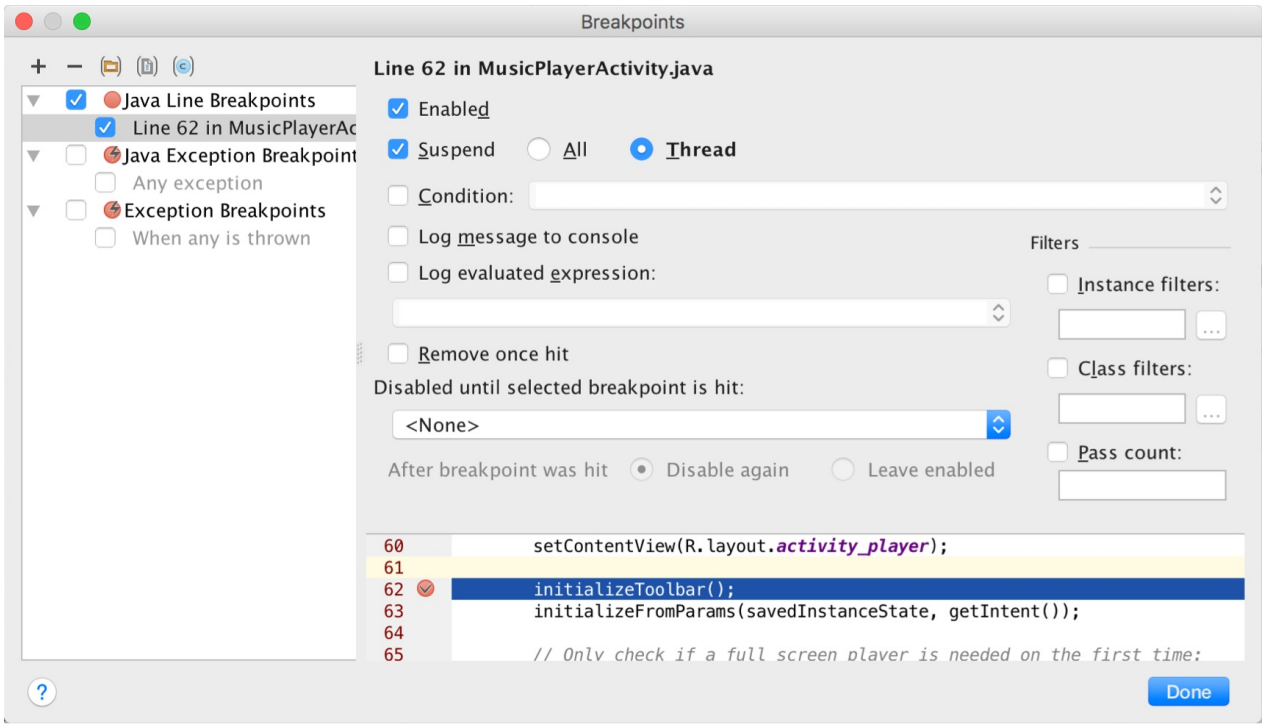
   Anda juga bisa menggunakan **Run > Toggle Line Breakpoint** atau Control-F8 (Command-F8 di OS X) untuk menyetel breakpoint pada baris.

Jika aplikasi sudah berjalan, Anda tidak perlu memperbaruinya untuk menambahkan breakpoint.

Bila eksekusi kode sudah mencapai breakpoint, Android Studio menghentikan sementara eksekusi aplikasi Anda. Selanjutnya Anda bisa menggunakan alat (bantu) dalam debugger Android untuk menampilkan keadaan aplikasi dan men-debug aplikasi saat berjalan.

Menampilkan dan mengonfigurasi breakpoint

Untuk menampilkan semua breakpoint yang telah disetel dan mengonfigurasi setelan breakpoint, klik ikon View Breakpoints  di tepi kiri jendela debugger. Jendela Breakpoints akan muncul.



Dalam jendela ini semua breakpoint yang telah Anda setel muncul di panel kiri, dan Anda bisa mengaktifkan atau menonaktifkan setiap breakpoint dengan kotak centang. Jika breakpoint dinonaktifkan, Android Studio tidak akan menghentikan sementara aplikasi Anda bila eksekusi mencapai breakpoint tersebut.

Pilih breakpoint dari daftar untuk mengonfigurasi setelannya. Anda bisa mengonfigurasi breakpoint agar dinonaktifkan di awal dan meminta sistem mengaktifkannya setelah breakpoint yang berbeda ditemukan. Anda juga bisa mengonfigurasi apakah breakpoint harus dinonaktifkan setelah tercapai.

Untuk menyetel breakpoint bagi suatu pengecualian, pilih Exception Breakpoints dalam daftar breakpoint.

Menonaktifkan (membisukan) semua breakpoint

Penonaktifan breakpoint memungkinkan Anda mem"bisukan" sementara breakpoint tanpa membuangnya dari kode. Jika breakpoint dibuang semuanya, Anda juga akan kehilangan semua ketentuan atau fitur lainnya yang telah dibuat untuk breakpoint tersebut, sehingga menonaktifkannya bisa menjadi pilihan yang lebih baik.

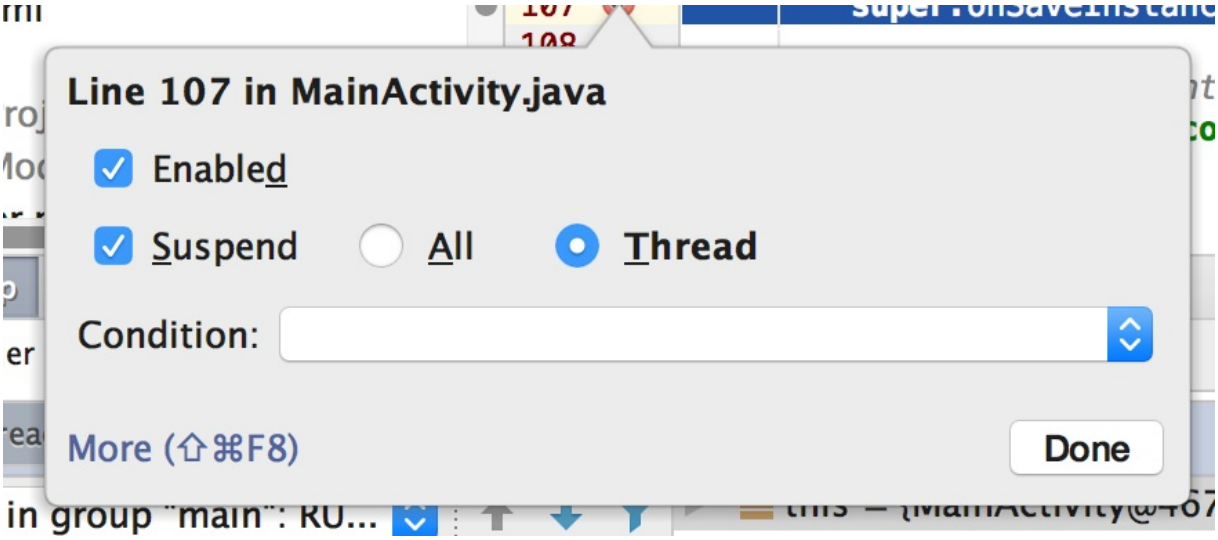
Untuk membisukan semua breakpoint, klik ikon **Mute Breakpoints**  . Klik lagi ikon tersebut untuk mengaktifkan (membunyikan) semua breakpoint.

Menggunakan breakpoint bersyarat

Breakpoint bersyarat adalah breakpoint yang hanya menghentikan eksekusi aplikasi Anda jika ketentuan pengujian terpenuhi. Untuk mendefinisikan pengujian breakpoint bersyarat, gunakan langkah-langkah ini:

1. Klik-kanan pada ikon breakpoint, dan masukkan pengujian dalam bidang Condition.

   Anda juga bisa menggunakan jendela Breakpoints untuk memasukkan ketentuan breakpoint.



   Pengujian yang dimasukkan dalam bidang ini bisa berupa ekspresi Java asalkan mengembalikan nilai boolean. Anda bisa menggunakan nama variabel dari aplikasi sebagai bagian dari ekspresi.

2. Jalankan aplikasi dalam mode debug. Eksekusi aplikasi Anda akan berhenti di breakpoint bersyarat, jika syarat bernilai true.

Merunut kode

Setelah eksekusi aplikasi berhenti karena telah mencapai breakpoint, Anda bisa mengeksekusi kode dari titik itu sebanyak satu baris dengan fungsi Step Over, Step Into, dan Step Out.

Untuk menggunakan salah satu fungsi langkah:

1. Mulailah men-debug aplikasi Anda. Hentikan sementara eksekusi aplikasi dengan breakpoint.

   Eksekusi aplikasi akan berhenti, dan debugger menampilkan keadaan aplikasi saat ini. Baris saat ini akan disorot dalam kode Anda.

2. Klik ikon **Step Over**  , pilih **Run > Step Over,** atau tekan F8.

   Step Over mengeksekusi baris kode berikutnya di kelas dan metode saat ini, yang mengeksekusi semua panggilan metode pada baris itu dan sisanya dalam file yang sama.

3. Klik ikon **Step Into**  , pilih **Run > Step Into,** atau tekan F7.

   Step Into akan melompat ke eksekusi panggilan metode pada baris saat ini (dibandingkan dengan hanya mengeksekusi metode tersebut dan sisanya pada baris yang sama). Tampilan **Frame** (yang akan Anda pelajari di bagian berikutnya) akan diperbarui untuk menampilkan bingkai tumpukan baru (metode baru). Jika panggilan metode dimuat dalam kelas lain, file untuk kelas itu akan dibuka dan baris saat ini dalam file itu akan disorot. Anda bisa terus merunut baris-baris dalam panggilan metode baru ini, atau merunut lebih dalam ke metode lainnya.

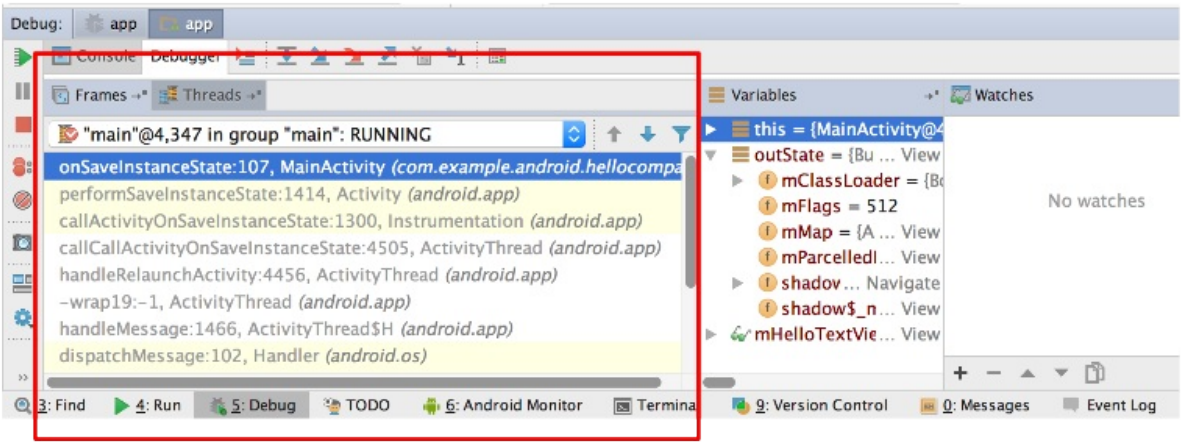
4. Klik ikon **Step Out**  , pilih **Run > Step Out,** atau tekan Shift-F8.

   Step Out akan menyelesaikan eksekusi metode saat ini dan mengembalikan ke titik pemanggilan metode semula.

5. Untuk melanjutkan eksekusi normal aplikasi, pilih **Run > Resume Program** atau klik ikon Resume  .

Menampilkan bingkai tumpukan eksekusi

Tampilan **Frames** pada jendela debugger memungkinkan Anda memeriksa tumpukan eksekusi dan bingkai tertentu yang menyebabkan breakpoint saat ini tercapai.



Tumpukan eksekusi menampilkan semua kelas dan metode (bingkai) yang sedang dieksekusi hingga titik ini dalam aplikasi, dalam urutan terbalik (bingkai terbaru lebih dahulu). Saat eksekusi bingkai tertentu selesai, bingkai tersebut akan dimunculkan dari tumpukan dan eksekusi kembali ke bingkai berikutnya.

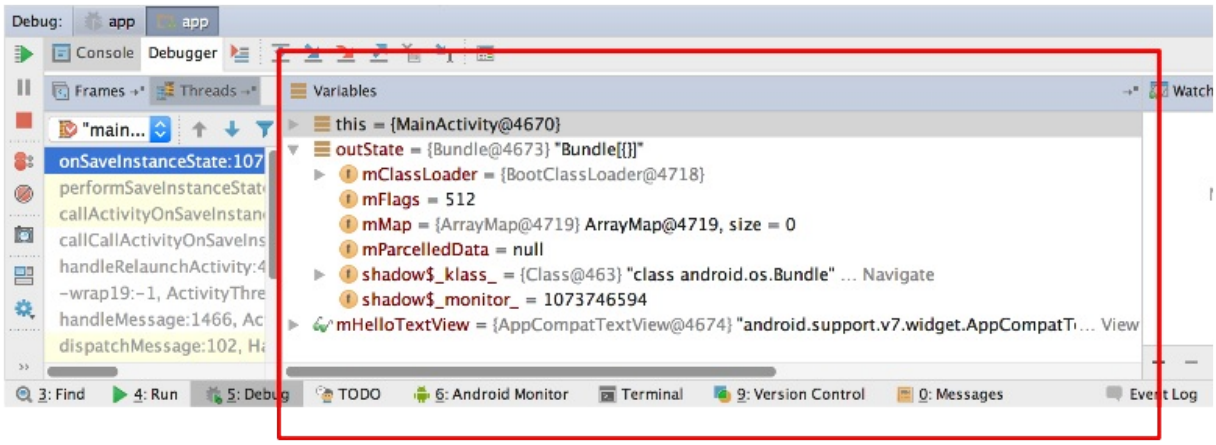
Mengeklik baris untuk bingkai dalam tampilan Frames akan membuka sumber terkait dalam editor dan menyoroti baris tempat bingkai dieksekusi pada awalnya. Tampilan Variables dan Watches juga diperbarui untuk mencerminkan keadaan lingkungan eksekusi ketika bingkai itu terakhir dimasuki.

Memeriksa dan memodifikasi variabel-variabel

Tampilan **Variables** pada jendela debugger memungkinkan Anda memeriksa variabel yang tersedia di bingkai tumpukan saat ini ketika sistem menghentikan aplikasi pada breakpoint. Variabel-variabel yang menyimpan objek atau kumpulan seperti larik bisa diluaskan untuk menampilkan komponennya.

Panel **Variable** juga memungkinkan Anda mengevaluasi ekspresi dengan cepat menggunakan variabel dan/atau metode statis yang tersedia dalam bingkai yang dipilih.

Jika tampilan Variables tidak terlihat, klik ikon **Restore Variables View**  .



Untuk memodifikasi variabel-variabel di aplikasi Anda saat dijalankan:

1. Klik-kanan sutau variabel di tampilan Variable, dan pilih **Set Value**. Anda juga bisa menggunakan F2.
2. Masukkan nilai baru untuk variabel, dan tekan Return.

Nilai yang Anda masukkan harus sesuai tipenya dengan variabel tersebut, atau Android Studio akan mengembalikan kesalahan "type mismatch".

Menyetel watches

Tampilan **Watches** menyediakan fungsionalitas yang sama dengan tampilan Variables hanya saja ekspresi yang ditambahkan ke panel Watches akan bertahan di antara sesi debug. Tambahkan watches untuk variabel-variabel dan bidang yang sering Anda akses atau yang menyediakan keadaan yang berguna untuk sesi debug saat ini.

Untuk menggunakan watches:

1. Mulailah men-debug aplikasi Anda.
2. Di panel Watches, klik tombol plus (+).

   Dalam kotak teks yang muncul, ketikkan nama variabel atau ekspresi yang ingin Anda amati, kemudian tekan Enter.

Buang item dari daftar Watches dengan memilih item tersebut kemudian mengeklik tombol minus (-).

Ubah urutan elemen dalam daftar Watches dengan memilih sebuah item kemudian mengeklik ikon naik atau turun.

Mengevaluasi ekspresi

Gunakan Evaluate Expression untuk menjelajahi keadaan variabel-variabel dan objek dalam aplikasi Anda, termasuk metode panggilan pada objek itu. Untuk mengevaluasi ekspresi:

1. Klik ikon Evaluate Expression  , atau pilih **Run > Evaluate Expression**. Anda juga bisa mengeklik-kanan pada suatu variabel dan memilih Evaluate Expression.

   Jendela Evaluate Expression akan muncul.

2. Masukkan suatu ekspresi ke dalam jendela Expression dan klik **Evaluate**.

   Jendela Evaluate Expression akan memperbarui hasil eksekusi. Perhatikan, hasil yang Anda dapat dari mengevaluasi ekspresi adalah berdasarkan pada keadaan aplikasi saat ini. Bergantung pada nilai variabel-variabel dalam aplikasi saat mengevaluasi ekspresi, Anda mungkin mendapatkan hasil yang berbeda. Mengubah nilai variabel-variabel dalam ekspresi juga akan mengubah keadaan aplikasi yang berjalan saat ini.

Alat (bantu) lainnya untuk men-debug

Android Studio dan Android SDK menyertakan sejumlah alat (bantu) lainnya untuk membantu Anda menemukan dan mengoreksi masalah dalam kode. Alat (bantu) ini antara lain:

- Log sistem (logcat). Seperti yang telah dipelajari dalam pelajaran sebelumnya, Anda bisa menggunakan kelas Log untuk mengirim pesan ke log sistem Android, dan menampilkan pesan itu di Android Studio.
  - Untuk menulis pesan log di kode Anda, gunakan kelas Log. Pesan log membantu memahami alur eksekusi dengan mengumpulkan keluaran debug sistem saat Anda berinteraksi dengan aplikasi. Pesan log bisa memberi tahu Anda tentang bagian aplikasi yang gagal. Untuk informasi selengkapnya tentang pembuatan log, lihat Membaca dan Menulis Log.
- Melacak dan Membuat Log. Analisis jejak memungkinkan Anda melihat banyaknya waktu yang dihabiskan dalam metode tertentu, dan metode yang membutuhkan waktu terlama.
  - Untuk membuat file pelacakan, sertakan kelas Debug dan panggil salah satu metode startMethodTracing(). Dalam panggilan tersebut, tetapkan nama dasar untuk file pelacakan yang dihasilkan sistem. Untuk menghentikan pelacakan, panggil stopMethodTracing(). Semua metode ini memulai dan menghentikan pelacakan metode di seluruh mesin virtual. Misalnya, Anda bisa memanggil startMethodTracing() di metode onCreate() aktivitas, dan memanggil stopMethodTracing() di metode onDestroy() aktivitas itu.

- Android Debug Bridge (ADB). ADB adalah alat (bantu) baris perintah yang memungkinkan Anda berkomunikasi dengan instance emulator atau perangkat berbasis Android yang terhubung.
- Dalvik Debug Monitor Server (DDMS). Alat (bantu) DDSM menyediakan layanan penerusan porta, tangkapan layar, informasi thread dan heap, logcat, proses, dan informasi keadaan radio, panggilan masuk dan spoofing SMS, spoofing data lokasi, dan lainnya.
- Monitor CPU dan memori. Android Studio menyertakan sejumlah monitor untuk membantu memvisualisasikan perilaku dan kinerja aplikasi Anda.
- Tangkapan layar dan rekaman video.

Perekaman pelacakan ke log dan manifes Android

Ada beberapa jenis debug yang tersedia untuk Anda di luar setelan breakpoint dan perunutan kode. Anda juga bisa menggunakan pembuatan log dan pelacakan untuk menemukan masalah pada kode. Bila memiliki file log jejak (dihasilkan dengan menambahkan kode pelacakan ke aplikasi Anda atau melalui DDMS), Anda bisa memuat file log di Traceview, yang akan menampilkan data log dalam dua panel:

- Panel timeline -- menjelaskan waktu memulai dan menghentikan setiap thread dan metode
- Panel profil -- menyediakan rangkuman tentang hal yang terjadi di dalam metode

Demikian juga, Anda bisa menyetel android:debuggable di tag `<application>` Manifes Android ke `"true"` , yang menyetel apakah aplikasi bisa di-debug atau tidak, bahkan saat dijalankan pada perangkat dalam mode pengguna. Secara default, nilai ini disetel ke `"false"` .

Anda bisa membuat dan mengonfigurasi tipe pembangunan dalam file build.gradle tingkat modul di dalam blok {} android. Bila membuat modul baru, Android Studio secara otomatis membuatkan tipe pembangunan rilis dan debug untuk Anda. Meskipun tipe pembangunan debug tidak muncul dalam file konfigurasi pembangunan, Android Studio akan mengonfigurasinya dengan debuggable true. Hal ini memungkinkan Anda men-debug aplikasi pada perangkat Android yang aman dan mengonfigurasi penandatanganan APK dengan keystore debug generik. Anda bisa menambahkan tipe pembangunan debug ke konfigurasi jika ingin menambahkan atau mengubah setelan tertentu.

Semua perubahan yang dibuat untuk men-debug harus dibuang dari kode sebelum rilis karena bisa memengaruhi eksekusi dan kinerja kode produksi.

Saat mempersiapkan aplikasi untuk rilis, Anda harus membuang semua kode ekstra di file sumber yang ditulis untuk keperluan pengujian.

Selain mempersiapkan kode itu sendiri, ada beberapa tugas lain yang perlu diselesaikan agar aplikasi Anda siap dipublikasikan. Tugas tersebut antara lain:

- Membuang pernyataan log
- Membuang panggilan untuk menampilkan Toast
- Menonaktifkan debug di manifes Android dengan:
  - Membuang atribut android:debuggable dari tag `<application>`
  - Atau menyetel atribut android:debuggable ke false

Buang semua panggilan pelacakan debug dari file kode sumber Anda seperti startMethodTracing() dan stopMethodTracing().

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Menggunakan Debugger

Ketahui selengkapnya

- Debug Aplikasi Anda
- Tulis dan Tampilkan Log
- Analisis Pelacakan Tumpukan
- Android Monitor
- Menggunakan DDMS
- Android Debug Bridge
- Ringkasan Android Monitor
- Buat dan Edit Konfigurasi Run/Debug
- Men-debug dan Menguji di Android Studio (video)

3.2: Menguji Aplikasi Anda

Materi:

- Pengantar
- Tentang pengujian
- Mempersiapkan pengujian
- Membuat dan menjalankan pengujian unit
- Praktik Terkait
- Ketahui Selengkapnya

Dalam bab ini Anda akan mendapatkan ringkasan pengujian Android, serta tentang membuat dan menjalankan pengujian unit lokal di Android Studio bersama JUnit.

Tentang pengujian

Meskipun memiliki aplikasi yang dikompilasi, dijalankan dan terlihat seperti yang diinginkan pada perangkat lain, Anda harus memastikan bahwa aplikasi akan berperilaku seperti yang diharapkan dalam setiap situasi, terutama saat aplikasi tumbuh dan berubah. Bahkan jika Anda mencoba menguji aplikasi secara manual setiap kali membuat perubahan — prospek yang membosankan — Anda mungkin melewatkan sesuatu atau tidak mengantisipasi hal yang mungkin dilakukan pengguna akhir dengan aplikasi tersebut sehingga menyebabkan kegagalan.

Menulis dan menjalankan pengujian adalah bagian penting dari proses development perangkat lunak. "Test-Driven Development" (TDD) atau Pengembangan yang Digerakkan oleh Pengujian adalah filosofi development perangkat lunak populer yang menempatkan pengujian pada inti dari semua development perangkat lunak untuk aplikasi atau layanan. Ini tidak menghilangkan kebutuhan akan pengujian lebih jauh, namun hanya memberikan dasar yang kuat untuk digunakan.

Pengujian kode bisa membantu Anda menemukan masalah lebih awal di development—bila masalah sangat mahal untuk diatasi—dan meningkatkan ketangguhan kode saat aplikasi menjadi lebih besar dan lebih kompleks. Dengan beberapa pengujian dalam kode, Anda bisa menguji beberapa bagian kecil aplikasi secara terpisah, dan dengan cara yang dapat diulangi serta dilakukan secara otomatis. Karena….kode yang Anda tulis untuk menguji aplikasi tidak berakhir di versi produksi aplikasi; kode hanya berada di mesin development, bersama kode aplikasi Anda di Android Studio.

Tipe pengujian

Android mendukung sejumlah pengujian yang berbeda dan kerangka kerja pengujian. Dua bentuk dasar pengujian yang didukung Android Studio adalah pengujian unit lokal dan pengujian instrumentasi.

Pengujian unit lokal adalah pengujian yang dikompilasi dan dijalankan sepenuhnya pada mesin lokal Anda dengan Java Virtual Machine (JVM). Gunakan pengujian unit lokal untuk menguji bagian-bagian aplikasi (seperti logika internal) yang tidak memerlukan akses ke kerangka kerja Android atau perangkat Android atau pun emulator, atau bagian-bagian yang bisa Anda buatkan objek palsu ("tiruan" atau stub) yang seakan-akan berperilaku seperti padanan kerangka kerja.

Pengujian instrumentasi adalah pengujian yang berjalan pada perangkat Android atau emulator. Pengujian tersebut memiliki akses ke kerangka kerja Android dan ke informasi Instrumentasi seperti Konteks aplikasi. Anda bisa menggunakan pengujian instrumentasi untuk pengujian unit, pengujian antarmuka pengguna (UI), atau pengujian integrasi, yang memastikan komponen aplikasi berinteraksi dengan aplikasi lain dengan benar. Paling umum, Anda menggunakan pengujian instrumentasi untuk pengujian UI, yang memungkinkan Anda menguji apakah aplikasi berperilaku dengan benar bila pengguna berinteraksi dengan aktivitas aplikasi atau memasukkan masukan tertentu.

Untuk sebagian besar bentuk pengujian antarmuka pengguna, Anda menggunakan kerangka kerja Espresso, yang memungkinkan Anda menulis pengujian UI otomatis. Anda akan mempelajari tentang pengujian instrumentasi dan Espresso dalam bab berikutnya.

Pengujian Unit

Pengujian unit harus menjadi pengujian mendasar dalam strategi pengujian aplikasi Anda. Dengan membuat dan menjalankan pengujian unit terhadap kode, Anda bisa memverifikasi apakah logika area atau unit kode fungsional sudah benar. Menjalankan pengujian unit setelah setiap pembangunan akan membantu Anda menemukan dan mengatasi masalah yang ditimbulkan oleh perubahan kode pada aplikasi.

Pengujian unit umumnya menguji fungsionalitas unit kemungkinan kode terkecil (yang bisa berupa metode, kelas, atau komponen) dengan cara yang bisa diulang. Buat pengujian unit bila Anda perlu memverifikasi logika kode tertentu di aplikasi. Misalnya, jika Anda melakukan pengujian unit atas kelas, pengujian mungkin akan memeriksa apakah kelas dalam keadaan yang tepat. Untuk metode, Anda mungkin menguji perilakunya untuk nilai parameter yang berbeda, terutama nol. Biasanya, unit kode diuji secara terpisah; pengujian Anda hanya memantau perubahan pada unit itu saja. Kerangka kerja tiruan seperti Mockito bisa digunakan untuk mengisolasi unit Anda dari dependensinya. Anda juga bisa menulis pengujian unit untuk Android di JUnit 4, yakni kerangka kerja pengujian unit umum untuk kode Java.

Android Testing Support Library

Android Testing Support Library menyediakan infrastruktur dan API untuk pengujian aplikasi Android, termasuk dukungan untuk JUnit 4. Dengan pustaka dukungan pengujian, Anda bisa membangun dan menjalankan kode pengujian untuk aplikasi.

Anda mungkin sudah memiliki Android Testing Support Library yang dipasang bersama Android Studio. Untuk memeriksa Android Support Repository, ikuti langkah-langkah ini:

1. Di Android Studio pilih **Tools > Android > SDK Manager**.
2. Klik tab **SDK Tools**, dan cari Support Repository.
3. Jika perlu, perbarui atau pasang pustaka tersebut.

Kelas-kelas Android Testing Support Library berada di paket android.support.test. Ada juga API pengujian lama di android.test. Anda harus menggunakan pustaka dukungan terlebih dulu, bila diberi pilihan antara pustaka dukungan dan API lama, karena pustaka dukungan membantu membangun dan mendistribusikan pengujian secara lebih bersih dan lebih bisa diandalkan daripada pengkodean langsung terhadap API itu sendiri.

Mempersiapkan pengujian

Untuk mempersiapkan proyek bagi pengujian di Android Studio, Anda perlu:

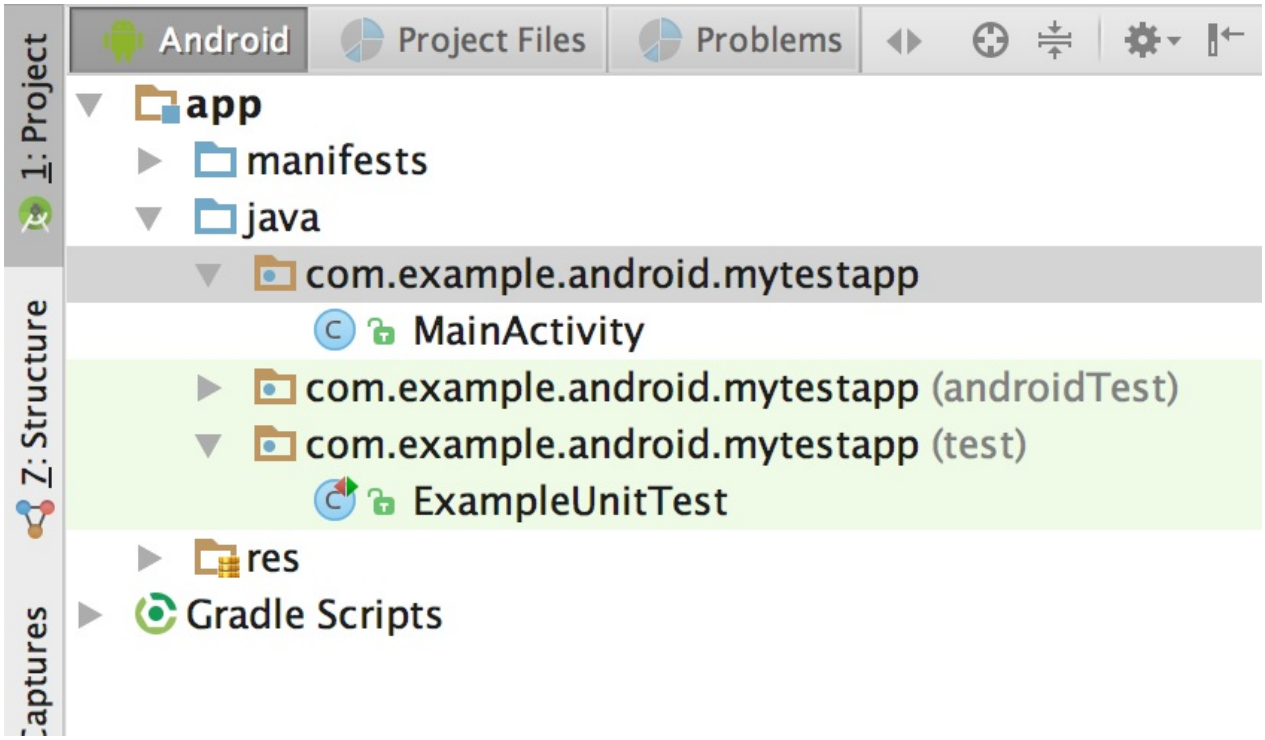
- Menyusun pengujian di *set sumber*.
- Mengonfigurasi dependensi gradle proyek untuk menyertakan API yang terkait pengujian.

Set sumber Android Studio

Set sumber adalah kumpulan kode terkait dalam proyek Anda, yaitu untuk target pembangunan atau "ragam" aplikasi yang berbeda. Bila Android Studio membuat proyek, maka akan membuat tiga set sumber untuk Anda:

- Set sumber utama, untuk kode dan sumber daya aplikasi.
- Set sumber **test**, untuk pengujian unit lokal aplikasi.
- Set sumber **androidTest**, untuk pengujian instrumentasi Android.

Set sumber muncul di tampilan Android pada Android Studio pada nama paket untuk aplikasi Anda. Set sumber utama hanya menyertakan nama paket. Set sumber test dan androidTest memiliki nama paket yang masing-masing diikuti dengan (test) atau (androidTest).



Set sumber ini sesuai dengan folder dalam direktori src untuk proyek Anda. Misalnya, file untuk set sumber pengujian test ada di src/test/java.

Konfigurasilah Gradle untuk dependensi pengujian

Untuk menggunakan API pengujian unit, Anda perlu mengonfigurasi dependensi proyek. File pembangunan gradle default untuk proyek Anda menyertakan sebagian dependensi ini secara default, namun Anda mungkin perlu menambahkan dependensi lainnya untuk fitur pengujian tambahan seperti kerangka kerja pencocokan atau tiruan.

Dalam file build.gradle level atas aplikasi Anda, tetapkan pustaka-pustaka ini sebagai dependensi. Perhatikan, nomor versi untuk pustaka ini mungkin telah berubah. Jika Android Studio melaporkan pustaka yang lebih baru, perbarui nomor untuk merefleksikan versi saat ini.

```
dependencies {
   // Required -- JUnit 4 framework
    testCompile 'junit:junit:4.12'
    // Optional -- hamcrest matchers
    testCompile 'org.hamcrest:hamcrest-library:1.3'
    // Optional -- Mockito framework
    testCompile 'org.mockito:mockito-core:1.10.19'
}
```

Setelah menambahkan dependensi ke file build.gradle, Anda mungkin harus menyinkronkan proyek untuk melanjutkan. Klik **Sync Now** di Android Studio bila dikonfirmasi.

Konfigurasilah runner pengujian

Runner pengujian adalah pustaka atau serangkaian alat (bantu) yang memungkinkan pengujian terjadi dan hasilnya dicetak ke log. Proyek Android Anda memiliki akses ke runner pengujian JUnit dasar sebagai bagian dari JUnit4 API. Pustaka dukungan pengujian Android menyertakan runner pengujian untuk pengujian instrumentasi dan Espresso, `AndroidJUnitRunner`, yang juga mendukung Junit 3 dan 4.

Bab ini hanya memperagakan runner default untuk pengujian unit. Untuk menyetel AndroidJUnitRunner sebagai runner pengujian default di proyek Gradle Anda, tambahkan dependensi berikut ke file build.gradle. Mungkin sudah ada dependensi di bagian defaultConfig. Jika demikian, tambahkan baris testInstrumentationRunner ke bagian tersebut.

```
android {
    defaultConfig {
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
}
```

Membuat dan menjalankan pengujian unit

Buat pengujian unit Anda sebagai file Java generik dengan menggunakan JUnit 4 API, dan simpan pengujian itu di set sumber **test**. Setiap template proyek Android Studio menyertakan set sumber dan file pengujian Java contoh yang disebut ExampleUnitTest.

Buat kelas pengujian baru

Untuk membuat file kelas pengujian baru, tambahkan file Java ke set sumber pengujian proyek Anda. File kelas pengujian untuk pengujian unit biasanya diberi nama untuk kelas dalam aplikasi yang diuji, dengan ditambahkan "Test". Misalnya, jika Anda memiliki kelas yang disebut Calculator di aplikasi, kelas untuk pengujian unit akan menjadi CalculatorTest.

Untuk menambahkan file kelas pengujian, gunakan langkah-langkah ini:

1. Luaskan folder java dan folder untuk set sumber pengujian aplikasi. File kelas pengujian unit yang ada akan ditampilkan.
2. Klik-kanan pada folder set sumber pengujian dan pilih **New > Java Class.**
3. Beri nama file dan klik **OK**.

Tulis pengujian Anda

Gunakan sintaks dan anotasi JUnit 4 untuk menulis pengujian Anda. Misalnya, kelas pengujian yang ditampilkan di bawah ini menyertakan anotasi berikut:

- Anotasi @RunWith menunjukkan runner pengujian yang harus digunakan untuk pengujian di kelas ini.
- Anotasi @SmallTest menunjukkan bahwa pengujian ini kecil (dan cepat).
- Anotasi @Before menandakan metode sedang dipersiapkan untuk pengujian.
- Anotasi @Test menandakan metode sebagai pengujian sebenarnya.

Untuk informasi selengkapnya tentang Anotasi JUnit, lihat Dokumentasi Referensi JUnit.

```
@RunWith(JUnit4.class)
@SmallTest
public class CalculatorTest {
   private Calculator mCalculator;
   // Set up the environment for testing
   @Before
   public void setUp() {
      mCalculator = new Calculator();
   }

   // test for simple addition
   @Test
   public void addTwoNumbers() {
       double resultAdd = mCalculator.add(1d, 1d);
       assertThat(resultAdd, is(equalTo(2d)));
   }
}
```

Metode addTwoNumbers() adalah satu-satunya pengujian sebenarnya. Bagian utama pengujian unit adalah pernyataan, yang didefinisikan di sini melalui metode assertThat(). Pernyataan adalah ekspresi yang harus mengevaluasi dan menghasilkan nilai true agar lulus pengujian. JUnit 4 menyediakan sejumlah metode pernyataan, namun assertThat()

adalah yang paling fleksibel, karena memungkinkan metode perbandingan serba-guna yang disebut *matchers*. Kerangka kerja Hamcrest umumnya digunakan untuk matchers ("Hamcrest" merupakan anagram untuk matchers). Hamcrest menyertakan sejumlah besar metode perbandingan serta memungkinkan Anda untuk menulis sendiri.

Untuk informasi selengkapnya tentang pernyataan, lihat dokumentasi referensi JUnit untuk kelas Assert. Untuk informasi selengkapnya tentang kerangka kerja hamcrest, lihat Tutorial Hamcrest.

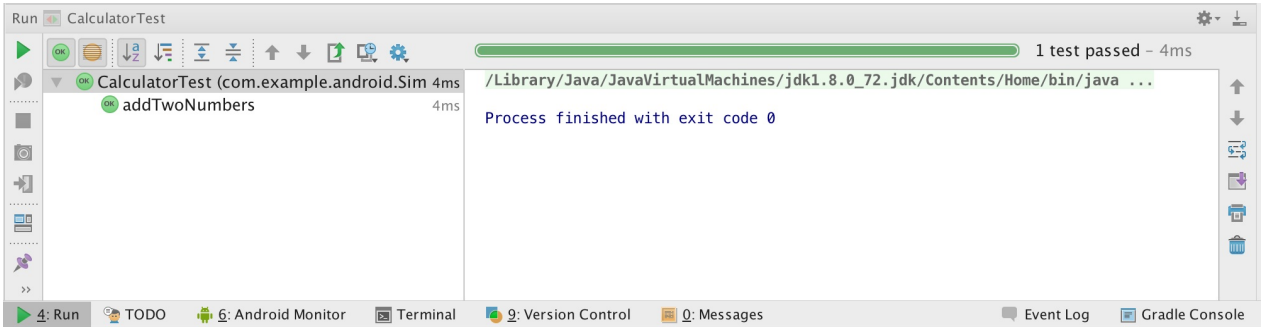
Perhatikan, metode addTwoNumbers() dalam contoh ini hanya menyertakan satu pernyataan. Aturan umum pengujian unit adalah untuk menyediakan metode pengujian terpisah bagi setiap pernyataan individual. Pengelompokan lebih dari satu pernyataan menjadi metode tunggal bisa membuat pengujian Anda lebih sulit di-debug jika hanya satu pernyataan yang gagal, dan mengaburkan pengujian yang berhasil.

Jalankan pengujian Anda

Untuk menjalankan pengujian unit lokal, gunakan langkah-langkah ini:

- Untuk menjalankan pengujian tunggal, klik-kanan metode pengujian tersebut dan pilih **Run**.
- Untuk menguji semua metode di kelas pengujian, klik-kanan file pengujian di tampilan proyek dan pilih **Run**.
- Untuk menjalankan semua pengujian di sebuah direktori, klik-kanan pada direktori dan pilih **Run tests**.

Proyek akan membangun, jika perlu, dan tampilan pengujian akan muncul di bagian bawah layar. Jika semua pengujian yang dijalankan berhasil, bilah kemajuan di bagian atas tampilan akan berubah menjadi hijau. Pesan status di footer juga melaporkan "Tests Passed".



Praktik Terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Menguji Aplikasi dengan Pengujian Unit

Ketahui Selengkapnya

- Praktik Terbaik untuk Pengujian
- Memulai Pengujian
- Membangun Pengujian Unit Lokal
- Laman Beranda JUnit 4
- Referensi JUnit 4 API
- Laman Beranda Mockito
- Dukungan Pengujian Android - Pola Pengujian (video)
- Codelab Pengujian Android
- Android Tools Protip: Anotasi Ukuran Pengujian
- [Manfaat Menggunakan assertThat dibandingkan Metode Pern
- yataan lainnya dalam Pengujian Unit](https://objectpartners.com/2013/09/18/the-benefits-of-using-assertthat-over-other-assert-methods-in-unit-tests/)

3.3: Pustaka Dukungan Android

Materi:

- Pengantar
- Tentang Pustaka Dukungan Android
- Pustaka dan fitur dukungan
- Mempersiapkan dan Menggunakan Pustaka Dukungan Android
- Praktik Terkait
- Ketahui Selengkapnya

Di bab ini Anda akan mendalami Pustaka Dukungan Android, bagian dari alat (bantu) Android SDK. Anda bisa menggunakan Pustaka Dukungan Android untuk versi yang kompatibel-mundur dari fitur Android dan untuk elemen UI tambahan serta fitur yang tidak disertakan dalam kerangka kerja Android standar.

Tentang Pustaka Dukungan Android

Alat (bantu) Android SDK menyertakan sejumlah pustaka yang secara kolektif disebut *Pustaka Dukungan Android*. Paket pustaka ini menyediakan sejumlah fitur yang tidak disertakan dalam kerangka kerja Android standar, dan menyediakan kompatibilitas mundur untuk perangkat lama. Sertakan pustaka ini di aplikasi Anda untuk menyertakan fungsionalitas pustaka itu.

Catatan: Pustaka Dukungan Android adalah paket yang berbeda dari pustaka Dukungan **Pengujian** Android yang telah Anda pelajari di bab sebelumnya. Pustaka dukungan pengujian menyediakan alat (bantu) dan API hanya untuk pengujian, sedangkan pustaka dukungan yang lebih umum menyediakan fitur dari semua jenis (namun tidak ada pengujian).

Fitur

Fitur Pustaka Dukungan Android antara lain:

- Versi kompatibel-mundur dari komponen kerangka kerja. Pustaka kompatibilitas ini memungkinkan Anda menggunakan fitur dan komponen yang tersedia pada versi platform Android yang lebih baru bahkan bila aplikasi berjalan pada versi platform lama. Misalnya, perangkat lama tidak memiliki akses ke fitur baru seperti fragmen, bilah aksi, atau elemen Desain Material. Pustaka dukungan menyediakan akses ke fitur itu pada perangkat lama.
- Layout tambahan dan elemen antarmuka pengguna. Pustaka dukungan menyertakan tampilan dan layout yang bisa berguna bagi aplikasi Anda, namun tidak disertakan dalam kerangka kerja Android standar. Misalnya, tampilan RecyclerView yang akan Anda gunakan dalam bab berikutnya adalah bagian dari pustaka dukungan.
- Dukungan untuk faktor bentuk perangkat yang berbeda, seperti TV atau perangkat yang dapat dikenakan: Misalnya, pustaka Leanback menyertakan komponen khusus untuk development aplikasi pada perangkat TV.
- Dukungan desain: Pustaka dukungan desain menyertakan komponen untuk mendukung elemen Desain Material di aplikasi Anda, termasuk tombol aksi mengambang (FAB). Anda akan mengetahui selengkapnya tentang Desain Material dalam bab berikutnya.
- Beragam fitur lainnya seperti dukungan palet, anotasi, dimensi layout berdasarkan persentase, dan preferensi.

Kompatibilitas Mundur

Pustaka dukungan memungkinkan aplikasi berjalan pada versi platform Android lama untuk mendukung fitur yang tersedia pada versi platform yang lebih baru. Misalnya, aplikasi yang berjalan pada versi Android di bawah 5.0 (API level 21) yang bergantung pada kelas kerangka kerja tidak bisa menampilkan elemen Desain Material, karena versi kerangka kerja Android tersebut tidak mendukung Desain Material. Akan tetapi, jika aplikasi menyertakan pustaka v7 appcompat, aplikasi tersebut akan memiliki akses ke banyak fitur yang tersedia di API level 21, termasuk dukungan untuk Desain Material. Hasilnya, aplikasi Anda bisa memberikan pengalaman yang lebih konsisten di beragam versi platform yang lebih luas.

API pustaka dukungan juga menyediakan layer kompatibilitas di antara versi API kerangka kerja yang berbeda. Layer kompatibilitas ini secara transparan mencegat panggilan API dan mengubah argumen yang diteruskan, menangani operasi itu sendiri, atau mengalihkan operasi. Dalam hal pustaka dukungan, dengan menggunakan metode layer kompatibilitas, Anda bisa memastikan interoperabilitas antara rilis Android lama dan baru. Setiap rilis Android baru akan menambahkan kelas dan metode baru, dan mungkin tidak lagi menggunakan beberapa kelas dan metode lama. Pustaka dukungan menyertakan kelas kompatibilitas yang bisa digunakan untuk kompatibilitas mundur. Anda bisa mengidentifikasi kelas ini berdasarkan namanya karena namanya menyertakan kata "Compat" (seperti ActivityCompat).

Bila aplikasi memanggil salah satu metode kelas dukungan, perilaku metode tersebut akan bergantung pada versi Android yang mendasarinya. Jika perangkat menyertakan fungsionalitas kerangka kerja yang diperlukan, pustaka dukungan akan menggunakan kerangka kerja. Jika perangkat menjalankan Android versi lama, pustaka dukungan akan berupaya mengimplementasikan perilaku kompatibel yang serupa dengan API yang tersedia.

Untuk sebagian besar kasus, Anda tidak perlu menulis kode rumit yang memeriksa versi Android dan menjalankan operasi yang berbeda berdasarkan versi itu. Anda bisa mengandalkan pustaka dukungan untuk melakukan pemeriksaan itu dan memilih perilaku yang sesuai.

Bila ragu, pilih kelas kompatibilitas pustaka dukungan daripada kelas kerangka kerja.

Versi

Setiap paket di pustaka dukungan memiliki nomor versi dalam tiga bagian (X.Y.Z) yang sesuai dengan level Android API, dan dengan revisi tertentu pustaka itu. Misalnya, nomor versi pustaka dukungan 22.3.4 adalah versi pustaka dukungan 3.4 untuk API 22.

Sebagai aturan umum, gunakan pustaka dukungan versi terbaru untuk API yang dikompilasi dan ditargetkan aplikasi Anda, atau versi yang lebih baru. Misalnya, jika aplikasi Anda menargetkan API 25, gunakan pustaka dukungan versi 25.X.X.

Kapan saja Anda bisa menggunakan pustaka dukungan yang lebih baru daripada pustaka dukungan untuk API yang ditargetkan. Misalnya, jika aplikasi menargetkan API 22, Anda bisa menggunakan pustaka dukungan versi 25 atau yang lebih tinggi. Hal sebaliknya tidak berlaku—Anda tidak bisa menggunakan pustaka dukungan lama bersama API yang lebih baru. Sebagai aturan umum, Anda harus mencoba menggunakan API dan pustaka dukungan terbaru di aplikasi.

Level API

Selain nomor versi yang sebenarnya, nama pustaka dukungan itu sendiri menunjukkan API level yang kompatibel-mundur dengan pustaka. Anda tidak bisa menggunakan pustaka dukungan di aplikasi untuk API yang lebih tinggi daripada API minimum yang didukung oleh aplikasi Anda. Misalnya, jika API minimum yang didukung aplikasi adalah 10, Anda tidak bisa menggunakan pustaka dukungan v13 atau pustaka dukungan preferensi V14 di aplikasi. Jika aplikasi Anda menggunakan beberapa pustaka dukungan, API minimum harus lebih tinggi daripada nomor terbesar, yakni jika Anda menyertakan pustaka dukungan untuk v7, v13, dan V14, API minimum Anda setidaknya harus 14.

Semua pustaka dukungan, termasuk pustaka v4 dan v7, memerlukan SDK minimum API 9.

Pustaka dukungan dan dukungan

Bagian ini menjelaskan fitur penting yang disediakan oleh pustaka di Pustaka Dukungan Android. Anda akan mengetahui selengkapnya tentang banyak fitur yang dijelaskan di bagian ini dalam bab berikutnya.

Pustaka dukungan v4

Pustaka dukungan v4 menyertakan rangkaian API terbesar dibandingkan dengan pustaka lainnya, termasuk dukungan untuk komponen aplikasi, fitur antarmuka pengguna, aksesibilitas, penanganan data, konektivitas jaringan, dan utilitas pemrograman.

Pustaka dukungan v4 menyertakan komponen khusus ini:

- Pustaka compat v4: Wrapper kompatibilitas (kelas-kelas yang menyertakan kata "Compat") untuk sejumlah API kerangka kerja inti.
- Pustaka core-utils v4: Menyediakan sejumlah kelas utilitas
- Pustaka core-ui v4: Mengimplementasikan berbagai komponen terkait UI.
- Pustaka media-compat v4: Bagian backport kerangka kerja media dari API 21.
- Pustaka fragmen v4: Menambahkan dukungan untuk fragmen Android.

Pustaka dukungan v7

Pustaka dukungan v7 menyertakan pustaka kompatibilitas dan fitur tambahan.

Pustaka dukungan v7 menyertakan semua pustaka dukungan v4, sehingga Anda tidak perlu menambahkannya secara terpisah. Dependensi pada pustaka dukungan v7 disertakan di setiap proyek Android Studio baru, dan aktivitas baru di proyek Anda yang meluaskan dari AppCompatActivity.

Pustaka dukungan v7 menyertakan komponen khusus ini:

- Pustaka appcompat v7: Menambahkan dukungan untuk pola desain antarmuka pengguna Bilah Aksi dan dukungan untuk implementasi antarmuka pengguna desain material.
- Pustaka cardview v7: Menyediakan kelas CardView, tampilan yang memungkinkan Anda menampilkan informasi di dalam kartu.
- Pustaka gridlayout v7: Menyertakan kelas GridLayout, yang memungkinkan Anda menyusun elemen antarmuka pengguna menggunakan petak sel persegi panjang
- Pustaka mediarouter v7: Menyediakan kelas MediaRouter dan kelas media terkait yang mendukung Google Cast.
- Pustaka palette v7: Mengimplementasikan kelas Palette, yang memungkinkan Anda mengekstrak warna yang menonjol dari gambar.
- Pustaka recyclerview v7: Menyediakan kelas RecyclerView, tampilan untuk menampilkan rangkaian data besar dengan menyediakan jendela item data yang terbatas secara efisien.
- Pustaka preferensi v7: Menyediakan API untuk mendukung objek preferensi di setelan aplikasi.

Pustaka lainnya

- Pustaka renderscript v8: Menambahkan dukungan untuk RenderScript, kerangka kerja untuk menjalankan tugas yang sarat komputasi dengan kinerja tinggi.
- Pustaka dukungan v13: Menyediakan dukungan untuk fragmen dengan kelas FragmentCompat dan kelas dukungan fragmen tambahan.
- Pustaka dukungan preferensi v14, dan pustaka dukungan preferensi v17 untuk TV: menyediakan API untuk menambahkan dukungan bagi antarmuka preferensi di perangkat seluler dan TV.
- Pustaka leanback v17: Menyediakan API untuk mendukung pembangunan antarmuka pengguna pada perangkat TV.
- Pustaka dukungan anotasi: Berisi API untuk mendukung penambahan metadata anotasi ke aplikasi Anda.
- Pustaka dukungan desain: Menambahkan dukungan untuk beragam komponen Desain Material dan pola seperti panel samping navigasi, tombol aksi mengambang (FAB), snackbar, dan tab.
- Pustaka dukungan Tab Khusus: Menambahkan dukungan untuk menambahkan dan mengelola tab khusus di aplikasi Anda.
- Pustaka dukungan persen: Memungkinkan Anda menambahkan dan mengelola dimensi berbasis persentase di aplikasi.
- Pustaka dukungan saran aplikasi untuk TV: Menyediakan API untuk mendukung penambahan saran materi di aplikasi yang berjalan pada perangkat TV.

Mempersiapkan dan menggunakan Pustaka Dukungan Android

Paket Pustaka Dukungan Android adalah bagian dari Android SDK, dan tersedia untuk diunduh di Android SDK Manager. Untuk mempersiapkan proyek Anda agar bisa menggunakan salah satu pustaka dukungan, gunakan langkah-langkah ini:

1. Unduh pustaka dukungan dengan Android SDK Manager, atau verifikasi apakah pustaka dukungan sudah tersedia.
2. Temukan pernyataan dependensi pustaka untuk pustaka dukungan yang mungkin Anda minati.
3. Tambahkan pernyataan dependensi itu ke file build.gradle.

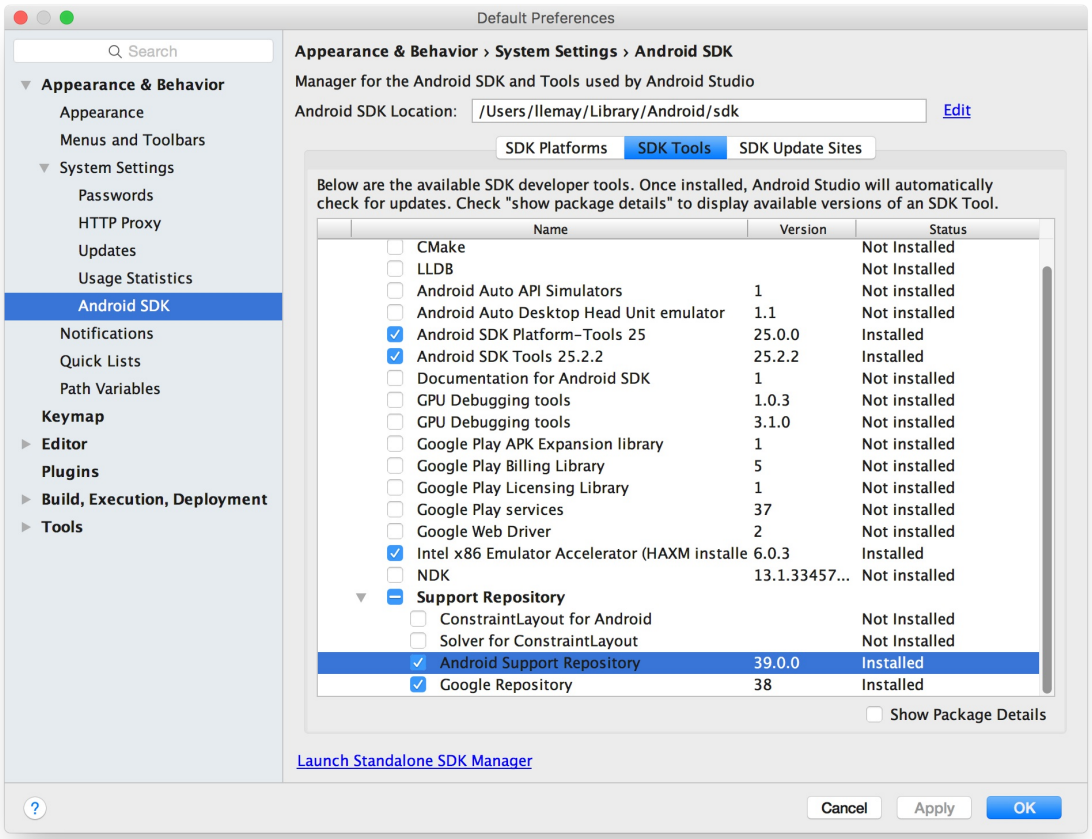
Mengunduh pustaka dukungan

Di Android Studio, Anda akan menggunakan Repositori Dukungan Android—repositori di SDK manager untuk semua pustaka dukungan—untuk mendapatkan akses ke pustaka dari dalam proyek Anda.

Anda mungkin sudah mengunduh dan memasang pustaka dukungan Android bersama Android Studio. Untuk memverifikasi apakah Anda telah memiliki pustaka dukungan yang tersedia, ikuti langkah-langkah ini:

1. Di Android Studio, pilih **Tools > Android > SDK Manager**, atau klik ikon SDK Manager  .

   Panel preferensi SDK Manager akan muncul.



2. Klik tab **SDK Tools** dan luaskan Support Repository.
3. Cari **Android Support Repository** dalam daftar.
   - Jika **Installed** muncul di kolom Status, berarti Anda sudah siap. Klik **Cancel**.
   - Jika **Not installed** atau **Update Available** muncul, klik kotak centang di sebelah Android Support Repository. Ikon unduhan akan muncul di sebelah kotak centang. Klik **OK.**
4. Klik **OK** lagi, kemudian **Finish** bila repositori dukungan telah dipasang.

Menemukan pernyataan dependensi pustaka

Untuk menyediakan akses ke pustaka dukungan dari proyek Anda, tambahkan pustaka tersebut ke file pembangunan gradle sebagai dependensi. Pernyataan dependensi memiliki format khusus yang menyertakan nama dan nomor versi pustaka.

1. Kunjungi laman Fitur Pustaka Dukungan di developer.android.com.
2. Temukan pustaka yang Anda minati di laman itu, misalnya, Pustaka Dukungan Desainuntuk dukungan Desain Material.
3. Salin pernyataan dependensi yang ditampilkan di akhir bagian. Misalnya, dependensi untuk pustaka dukungan desain terlihat seperti ini:

   ```
   com.android.support:design:23.3.0
   ```

   Nomor versi di akhir baris mungkin berbeda dari yang ditampilkan di atas. Anda akan memperbarui nomor versi bila menambahkan dependensi ke file build.gradle di langkah berikutnya.

Menambahkan dependensi ke file build.gradle

Skrip gradle untuk proyek Anda mengelola cara aplikasi dibangun, termasuk menetapkan dependensi yang dimiliki aplikasi di pustaka lainnya. Untuk menambahkan pustaka dukungan ke proyek Anda, modifikasi file pembangunan gradle untuk menyertakan dependensi ke pustaka yang ditemukan di bagian sebelumnya.

1. Di Android Studio, pastikan panel **Project** terbuka dan tab Android diklik.
2. Luaskan **Gradle Scripts**, jika perlu, dan buka file **build.gradle (Modul: app)**.

   Perhatikan, build.gradle untuk keseluruhan proyek (build.gradle (Project: app_name) adalah file yang berbeda dari build.gradle untuk modul aplikasi.

3. Cari bagian `dependencies` build.gradle, di dekat akhir file.

   Bagian dependensi untuk proyek baru mungkin sudah menyertakan dependensi sejumlah pustaka lainnya.

4. Tambahkan dependensi untuk pustaka dukungan yang menyertakan pernyataan yang disalin dalam tugas sebelumnya. Misalnya, dependensi lengkap pada pustaka dukungan desain terlihat seperti ini:

   ```
   compile 'com.android.support:design:23.3.0'
   ```

5. Perbarui nomor versi, jika perlu.

   Jika nomor versi yang Anda tetapkan lebih rendah dari nomor versi pustaka yang tersedia saat ini, Android Studio akan memperingatkan bahwa versi yang diperbarui telah tersedia. ("a newer version of `com.android.support:design` is available"). Edit nomor versi ke versi yang diperbarui, atau tekan Shift+Enter dan pilih "Change to XX.X.X", dalam hal ini XX.X.X adalah nomor versi yang diperbarui.

6. Klik **Sync Now** untuk menyinkronkan file gradle yang diperbarui bersama proyek, jika dikonfirmasi.

Menggunakan API pustaka dukungan

Semua kelas pustaka dukungan dimuat dalam paket android.support, misalnya, android.support.v7.app.AppCompatActivity adalah nama yang benar-benar memenuhi syarat untuk kelas AppCompatActivity, dari semua aktivitas yang diperluas.

Kelas-kelas Pustaka Dukungan yang menyediakan dukungan untuk API kerangka kerja yang ada biasanya memiliki nama yang sama dengan kelas kerangka kerja namun berada di paket kelas android.support. Pastikan bila mengimpor kelas, Anda menggunakan nama paket yang benar untuk kelas yang diminati. Misalnya, saat menerapkan kelas ActionBar, gunakan salah satu dari:

- `android.support.v7.app.ActionBar` saat menggunakan Pustaka Dukungan.
- `android.app.ActionBar` saat mengembangkan hanya untuk API level 11 atau yang lebih tinggi.

Pustaka dukungan juga menyertakan sejumlah kelas View yang digunakan dalam file layout XML. Dalam kasus tampilan, Anda harus selalu menggunakan nama yang benar-benar memenuhi syarat tampilan itu dalam elemen XML untuk tampilan itu:

```
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
</android.support.design.widget.CoordinatorLayout>
```

Catatan: Anda akan mempelajari tentang CoordinatorLayout dalam bab berikutnya.

Memeriksa versi sistem

Meskipun pustaka dukungan bisa membantu Anda mengimplementasikan aplikasi tunggal yang bekerja di seluruh versi platform Android, mungkin ada saatnya Anda perlu memeriksa versi Android yang dijalankan aplikasi, dan menyediakan kode yang benar untuk versi itu.

Android menyediakan kode unik untuk setiap versi platform di kelas konstanta Build. Gunakan kode ini dalam aplikasi Anda untuk menguji versi dan untuk memastikan bahwa kode yang bergantung pada API level yang lebih tinggi hanya dijalankan bila API itu tersedia pada sistem.

```
private void setUpActionBar() {
    // Make sure we're running on Honeycomb or higher to use ActionBar APIs
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
        ActionBar actionBar = getActionBar();
        actionBar.setDisplayHomeAsUpEnabled(true);
    } else { // do something else }
}
```

Praktik Terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Menggunakan Pustaka Dukungan Android

Ketahui Selengkapnya

- Pustaka Dukungan Android (pengantar)
- Persiapan Pustaka Dukungan
- Fitur Pustaka Dukungan
- Mendukung Versi Platform Berbeda
- Memilih compileSdkVersion, minSdkVersion, dan targetSdkVersion
- Semua Tentang Compat
- Referensi API (semua paket yang dimulai dengan android.support)

4.1: Kontrol Masukan Pengguna

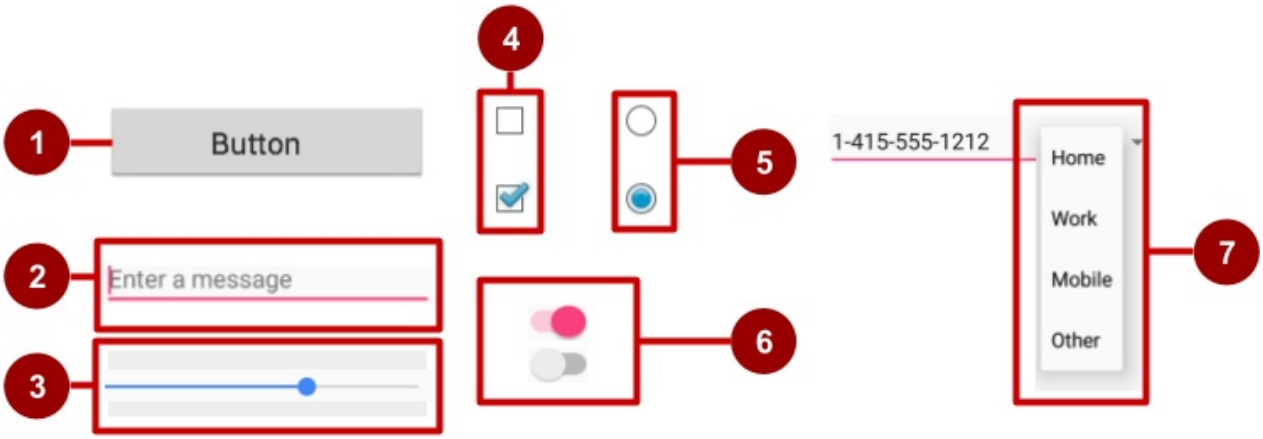
Desain interaksi untuk masukan pengguna

Alasan Anda mendesain aplikasi adalah menyediakan beberapa fungsi bagi pengguna, dan untuk menggunakannya, harus ada cara agar pengguna dapat *berinteraksi* dengannya. Untuk aplikasi Android, interaksi biasanya menyertakan mengetuk, menekan, atau berbicara dan mendengarkan. Kerangka kerja menyediakan elemen antarmuka pengguna (UI) yang sesuai, seperti tombol, menu, keyboard, bidang entri teks, dan mikrofon.

Dalam bab ini, Anda akan mempelajari cara mendesain interaksi pengguna suatu aplikasi—tombol yang diperlukan untuk memicu tindakan, dan bidang entri teks untuk masukan pengguna. Dalam mendesain, Anda perlu mengantisipasi apa yang mungkin perlu dilakukan pengguna, dan memastikan bahwa UI memiliki elemen yang mudah diakses, dimengerti, dan digunakan. Saat aplikasi Anda perlu mengambil data dari pengguna, buatlah mudah dan jelas. Berikan pengguna bantuan apa pun yang memungkinkan untuk menyediakan data masukan, misalnya mengantisipasi sumber data, meminimalkan jumlah isyarat pengguna (seperti ketukan dan gesekan), dan formulir yang sudah diisi jika memungkinkan.

Pastikan aplikasi Anda intuitif; yaitu, aplikasi harus bekerja seperti yang diharapkan oleh pengguna. Saat menyewa mobil, Anda mengharapkan roda, persneling, lampu utama, dan indikator ada di tempat tertentu. Saat masuk suatu ruangan, Anda mengharapkan sakelar lampu ada di tempat tertentu. Saat memulai aplikasi, pengguna mengharapkan tombol dapat diklik, spinner untuk menampilkan menu tarik-turun, dan bidang pengeditan teks untuk menampilkan keyboard di layar saat mengetuk di dalamnya. Jangan rusak harapan yang sudah mapan, atau Anda akan menyulitkan pengguna dalam menggunakan aplikasi tersebut.

Catatan: Pengguna Android telah familier dengan elemen UI yang berfungsi dengan cara tertentu, jadi Anda perlu konsisten dengan pengalaman aplikasi Android lain, dan pilihan Anda serta layout-nya harus dapat diprediksi. Dengan demikian, Anda akan membuat aplikasi yang memuaskan pelanggan.
Bab ini memperkenalkan *kontrol masukan* Android, yaitu komponen interaktif dalam antarmuka pengguna aplikasi Anda. Anda bisa menggunakan berbagai kontrol masukan dalam UI Anda, seperti bidang teks, tombol, kotak centang, tombol radio, tombol beralih, spinner, dan lain-lain.



Pada gambar di atas:

1. Tombol
2. Bidang teks
3. Bilah telusur
4. Kotak centang
5. Tombol radio
6. Beralih
7. Spinner

Untuk keterangan singkat setiap kontrol masukan, lihat Kontrol Masukan dalam dokumentasi developer.

Kontrol masukan dan fokus tampilan

Android menerapkan abstraksi program yang umum untuk semua kontrol masukan yang disebut *tampilan*. Kelas View menyatakan blok pembangunan dasar untuk semua komponen UI, termasuk kontrol masukan. Di bab sebelumnya, kita telah mengetahui bahwa `View` adalah kelas dasar untuk kelas-kelas yang menyediakan dukungan untuk komponen UI interaktif, seperti tombol, bidang teks, dan pengelola layout.

Jika ada banyak komponen masukan UI dalam aplikasi Anda, manakah yang pertama mendapat masukan dari pengguna? Misalnya, jika Anda memiliki sejumlah objek TextView dan objek EditText dalam aplikasi, komponen UI mana (yaitu, Tampilan yang mana) yang pertama menerima teks yang diketikkan pengguna?

Tampilan yang "memiliki fokus" adalah komponen yang menerima masukan pengguna.

Fokus menunjukkan tampilan mana yang saat ini dipilih untuk menerima masukan. Fokus bisa dimulai oleh pengguna dengan menyentuh sebuah Tampilan, seperti objek TextView atau EditText. Anda bisa mendefinisikan urutan fokus yang memandu pengguna dari kontrol UI ke kontrol UI menggunakan tombol Return, tombol Tab, atau tanda panah. Fokus juga bisa dikontrol lewat program; programmer bisa `requestFocus()` pada setiap Tampilan yang bisa difokus.

Atribut lain dari kontrol masukan adalah *clickable*. Jika atribut ini (boolean) `true`, maka Tampilan bisa bereaksi pada kejadian klik. Karena dengan fokus, clickable bisa dikontrol lewat program.

Perbedaan antara *bisa diklik* dan *bisa difokus* adalah bahwa dapat diklik berarti tampilan bisa diklik atau diketuk, sementara bisa difokus berarti tampilan dimungkinkan untuk mendapat fokus dari perangkat masukan seperti keyboard. Perangkat masukan seperti keyboard tidak bisa menentukan ke tampilan mana kejadian masukan mereka dikirim, jadi mereka mengirimnya ke tampilan yang memiliki fokus.

Metode masukan perangkat Android menjadi sangat beragam: tombol arah, trackball, layar sentuh, keyboard, dan masih banyak lagi. Navigasi utama beberapa perangkat, seperti tablet dan ponsel cerdas, adalah dengan sentuhan. Perangkat lain, seperti Google TV, tidak memiliki layar sentuh atau semacamnya dan bergantung pada perangkat masukan seperti tombol arah (d-pad). Saat pengguna mengarahkan melalui antarmuka pengguna dengan perangkat masukan seperti tombol arah atau trackball, Anda perlu:

- Secara visual perjelas tampilan mana yang memiliki fokus, sehingga pengguna tahu di mana harus melakukan masukan.
- Setel fokus secara eksplisit dalam kode Anda untuk menyediakan jalur kepada pengguna dalam menyusuri elemen masukan dengan menggunakan tombol arah atau trackball.

Dalam sebagian besar kasus, Anda tidak perlu mengontrol fokus sendiri, kecuali jika Anda ingin menyediakan serangkaian bidang masukan teks dan Anda ingin pengguna dapat berpindah dari satu bidang ke bidang berikutnya dengan mengetuk tombol Return atau Tab. Android menyediakan "mode sentuh" untuk perangkat yang bisa disentuh, seperti ponsel cerdas dan tablet. Saat pengguna mulai berinteraksi dengan antarmuka pengguna dengan menyentuhnya, hanya Tampilan dengan `isFocusableInTouchMode()` yang disetel ke `true` yang bisa difokus, seperti bidang masukan teks. Tampilan lain yang dapat disentuh, seperti tombol, tidak dapat mengambil fokus saat disentuh. Jika pengguna menekan tombol arah atau menggulir trackball, perangkat keluar dari "mode sentuh" dan mencari tampilan untuk mengambil fokus.

Gerakan fokus berdasarkan algoritme yang mencari tetangga terdekat pada arah yang ditentukan:

- Saat pengguna menyentuh layar, tampilan paling atas di bawah sentuhan berada dalam fokus, menyediakan akses-sentuh untuk tampilan anak dari tampilan paling atas.
- Jika Anda menyetel tampilan EditText ke satu baris tunggal, pengguna bisa mengetuk tombol Return pada keyboard untuk menutup keyboard dan memindah fokus ke tampilan kontrol masukan berikutnya berdasarkan apa yang ditemukan sistem Android:
  - Sistem biasanya mencari kontrol masukan terdekat pada arah yang sama dengan yang diarahkan pengguna (atas, bawah, kiri, atau kanan).
  - Jika ada beberapa kontrol masukan yang berdekatan dan pada arah yang sama, sistem akan memindai dari kiri ke kanan, atas ke bawah.
- Fokus juga bisa dipindah ke tampilan yang berbeda jika pengguna berinteraksi dengan kontrol arah, seperti tombol arah (d-pad) atau trackball.

Anda bisa mempengaruhi cara Android menangani fokus dengan mengatur kontrol masukan seperti elemen EditText pada layout tertentu dari kiri ke kanan dan atas ke bawah, sehingga fokus berpindah dari satu elemen ke elemen lainnya dalam urutan yang Anda inginkan.

Jika algoritme tidak menghasilkan apa yang Anda inginkan, Anda bisa menggantinya dengan menambahkan atribut XML `nextFocusDown`, `nextFocusLeft`, `nextFocusRight`, dan `nextFocusUp` pada file layout.

1. Tambahkan salah satu atribut ini pada suatu tampilan untuk menentukan ke mana harus pergi setelah meninggalkan tampilan tersebut—dengan kata lain, tampilan mana yang harus menjadi tampilan *berikutnya*.
2. Definisikan nilai atribut menjadi `id` pada tampilan berikutnya. Misalnya:

```
<LinearLayout
android:orientation="vertical"
... >
<Button android:id="@+id/top"
      android:nextFocusUp="@+id/bottom"
      ... />
<Button android:id="@+id/bottom"
      android:nextFocusDown="@+id/top"
      ... />
</LinearLayout>
```

Umumnya dalam `LinearLayout` vertikal, mengarahkan ke atas dari `Button` pertama, tidak akan bergerak ke mana pun, begitu juga mengarahkan ke bawah dari `Button` yang kedua. Namun dalam contoh di atas, `Button` atas telah mendefinisikan `button` bawah sebagai `nextFocusUp` (dan sebaliknya), sehingga fokus navigasi akan silih berganti dari atas ke bawah dan bawah ke atas.

Jika Anda ingin mendeklarasikan Tampilan sebagai bisa difokus dalam UI (yang biasanya tidak), tambahkan atribut XML `android:focusable` ke Tampilan di layout, dan setel nilainya ke `true`. Anda juga bisa mendeklarasikan Tampilan sebagai bisa difokus saat dalam "mode sentuh" dengan `android:focusableInTouchMode` yang disetel ke `true`.

Anda juga bisa secara eksplisit menyetel fokus atau mencari tampilan yang memiliki fokus dengan menggunakan metode berikut:

- Panggil onFocusChanged untuk menentukan asal fokus.
- Untuk mengetahui tampilan mana yang saat ini memiliki fokus, panggil Activity.getCurrentFocus(), atau gunakan ViewGroup.getFocusedChild() untuk kembali ke anak tampilan yang difokuskan (jika ada).
- Untuk menemukan tampilan dalam hierarki yang saat ini memiliki fokus, gunakan findFocus().
- Gunakan requestFocus untuk memberi fokus ke tampilan yang ditetapkan.
- Untuk mengubah apakah suatu tampilan bisa mengambil fokus, panggil setFocusable.
- Untuk menyetel listener yang akan mendapat notifikasi saat tampilan mendapat atau kehilangan fokus, gunakan setOnFocusChangeListener.

Memahami fokus terkait kontrol masukan penting untuk memahami cara kerja keyboard di layar dengan tampilan pengeditan teks. Misalnya, mengetuk tombol Return pada keyboard bisa memasukkan baris baru atau melanjutkan fokus ke tampilan berikutnya, bergantung pada atribut yang Anda gunakan dengan tampilan EditText. Anda akan mengetahui selengkapnya tentang fokus dengan tampilan pengeditan teks di bab ini nanti.

Menggunakan tombol

Orang suka menekan tombol.Tampilkan tombol merah besar pada seseorang dengan pesan yang berbunyi "Jangan tekan", maka kemungkinan orang tersebut akan menekannya hanya untuk merasakan kesenangan saat menekan tombol merah besar (larangan menekan tombol juga menjadi penyebabnya).

Anda bisa membuat Tombol menggunakan:

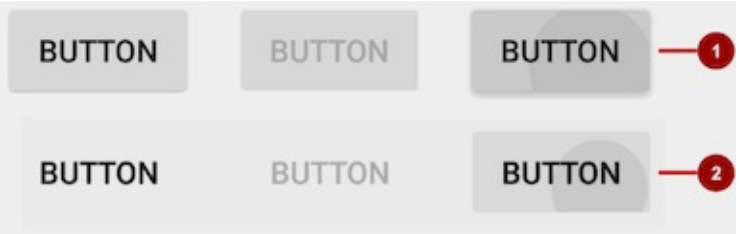
- Hanya teks, seperti yang ditampilkan pada sisi kiri gambar di bawah ini.
- Hanya ikon, seperti yang ditampilkan di tengah gambar di bawah ini.
- Teks dan ikon sekaligus, seperti yang ditampilkan pada sisi kanan gambar di bawah ini.

Bila disentuh atau diklik, tombol akan melakukan suatu aksi. Teks dan/atau ikon menyediakan petunjuk mengenai aksi itu.

Ini disebut "tombol-tekan" dalam dokumentasi Android. 

Tombol adalah suatu persegi atau persegi bersudut tumpul, dengan alas lebih lebar daripada tingginya, dengan teks deskriptif di tengahnya. Tombol Android mengikuti panduan dalam Spesifikasi Desain Material Android—Anda akan mengetahui selengkapnya tentang hal ini pada pelajaran selanjutnya.

Android menawarkan sejumlah tipe tombol, termasuk tombol timbul dan tombol datar seperti yang ditampilkan dalam gambar di bawah ini. Tombol-tombol ini memiliki tiga keadaan: normal, dinonaktifkan, dan ditekan.



Dalam gambar di atas:

1. Tombol timbul dalam tiga keadaan: normal, dinonaktifkan, dan ditekan.
2. Tombol datar dalam tiga keadaan: normal, dinonaktifkan, dan ditekan.

Mendesain tombol timbul

Tombol timbul adalah persegi atau persegi bersudut tumpul yang tampak terangkat dari layar—bayangan di sekitarnya menunjukkan bahwa tombol ini bisa disentuh atau diklik. Tombol timbul bisa menampilkan teks atau ikon, atau menampilkan teks dan ikon sekaligus.

Untuk menggunakan tombol timbul yang sesuai dengan Spesifikasi Desain Material, ikuti langkah-langkah ini:

1. Dalam file **build.gradle (Modul: app)** Anda, tambahkan pustaka appcompat terbaru ke bagian `dependencies` :

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

   Di atas, *x.x.x.* adalah nomor versi. Jika nomor versi yang Anda tetapkan lebih rendah dari nomor versi pustaka yang tersedia saat ini, Android Studio akan memperingatkan Anda ("a newer version is available"). Perbarui nomor versi ke versi yang disebutkan Android Studio untuk digunakan.

2. Buat aktivitas Anda memperluas `android.support.v7.app.AppCompatActivity` :

   ```
   public class MainActivity extends AppCompatActivity {
   ...
   }
   ```

3. Gunakan elemen `Button` dalam file layout. Tidak perlu atribut tambahan, karena tombol timbul adalah gaya default.

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
... />
```

Gunakan tombol timbul untuk lebih menonjolkan tindakan di layout dengan berbagai macam materi. Tombol timbul menambahkan dimensi pada layout datar—tombol tersebut menekankan fungsi pada ruang yang penuh atau longgar. Tombol timbul menampilkan bayangan latar belakang saat disentuh (ditekan) atau diklik, seperti yang ditampilkan di bawah ini. 

Dalam gambar di atas:

1. Keadaan normal: Dalam keadaan normal, tombol tampak seperti tombol timbul.
2. Keadaan dinonaktifkan: Bila dinonaktifkan, tombol akan berwarna abu-abu dan tidak aktif dalam konteks aplikasi. Di sebagian besar kasus, Anda akan menyembunyikan tombol nonaktif, namun kadang Anda ingin menampilkannya sebagai dinonaktifkan.
3. Keadaan ditekan: Keadaan ditekan, dengan bayangan latar belakang lebih besar, menunjukkan bahwa tombol sedang disentuh atau diklik. Bila Anda memasang callback ke tombol (seperti atribut OnClick), callback akan dipanggil saat tombol dalam keadaan ini.

Membuat tombol timbul dengan teks

Beberapa tombol timbul paling baik didesain sebagai teks, tanpa ikon, seperti tombol "Save", karena ikon saja tidak menyampaikan makna yang jelas. Untuk membuat tombol timbul dengan teks, gunakan kelas Button, yang memperluas kelas TextView.

Untuk membuat tombol timbul dengan teks saja, gunakan kelas `Button` di layout XML Anda seperti berikut:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    ... />
```

Praktik terbaik terkait tombol teks adalah mendefinisikan satu kata yang sangat pendek sebagai sumber daya string (`button_text` dalam contoh di atas), sehingga string bisa diterjemahkan. Misalnya, "Save" bisa diterjemahkan ke dalam bahasa Prancis sebagai "Enregister" tanpa mengubah kode.

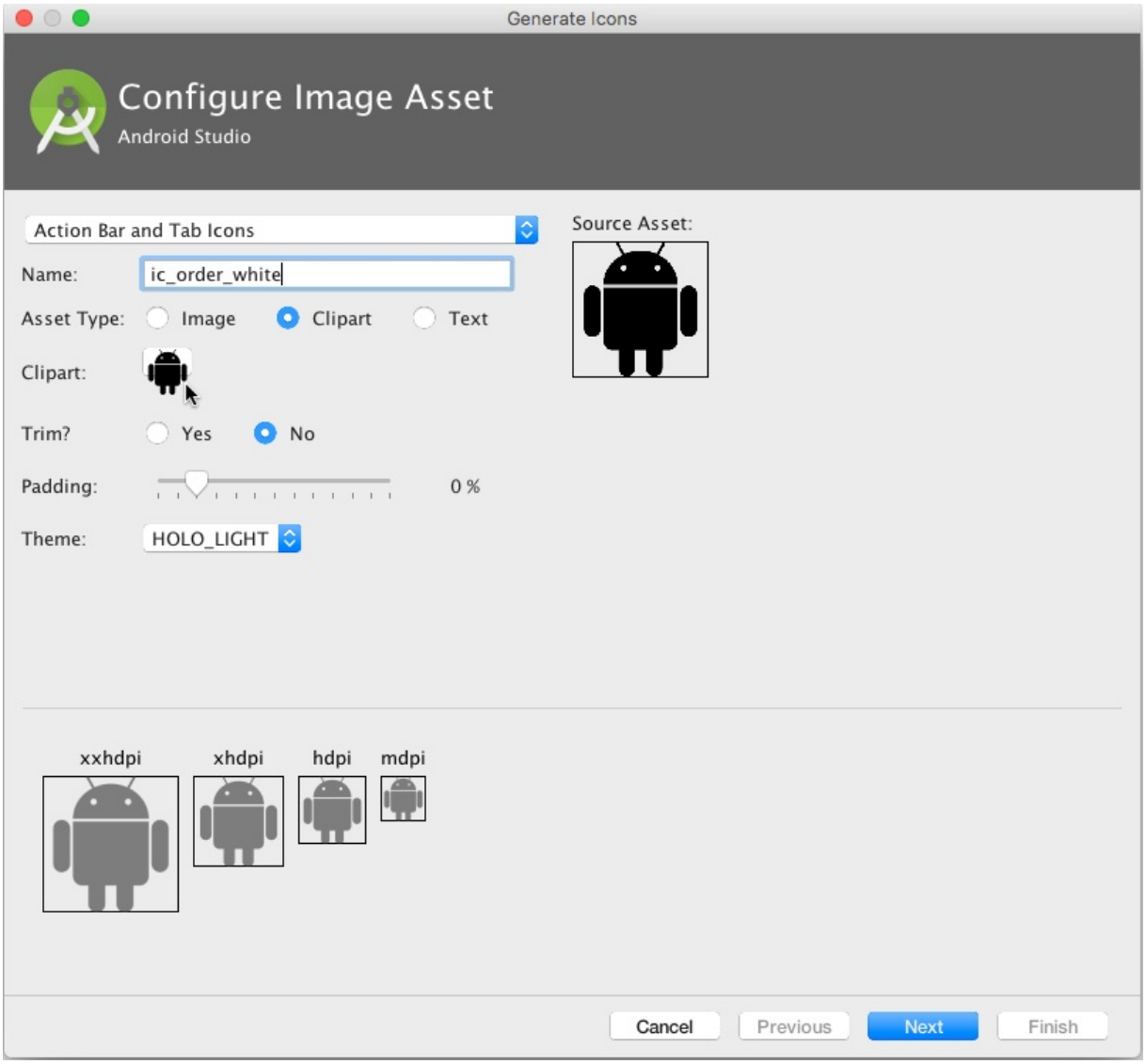
Membuat tombol timbul dengan ikon dan teks

Walaupun tombol biasanya menampilkan teks yang memberi tahu pengguna fungsi tombol tersebut, tombol timbul juga bisa menampilkan ikon beserta teks.

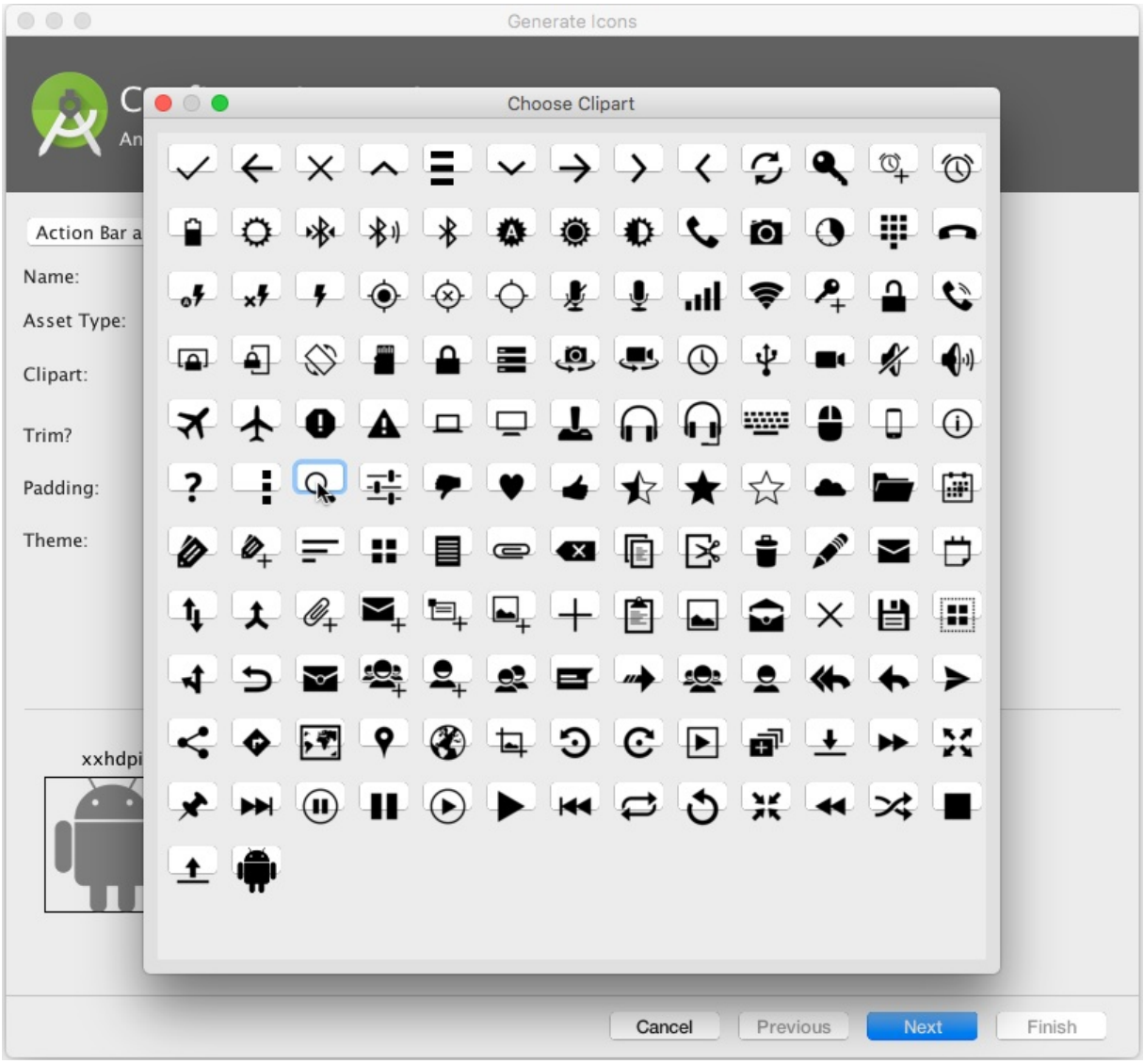
Memilih ikon

Untuk memilih gambar ikon standar yang diubah ukurannya untuk berbagai tampilan, ikuti langkah-langkah ini:

1. Luaskan **app > res** dalam tampilan Project, dan klik-kanan (atau klik Command) **drawable**.
2. Pilih **New > Image Asset**. Dialog Configure Image Asset akan muncul.



3. Pilih **Action Bar and Tab Items** dalam menu tarik-turun dialog Configure Image Asset (lihat Image Asset Studio untuk keterangan lengkap dialog ini.)
4. Klik gambar **Clipart:** (logo Android) untuk memilih gambar clipart sebagai ikon. Laman ikon akan muncul seperti yang ditampilkan di bawah ini. Klik ikon yang ingin Anda gunakan.



5. Anda mungkin perlu membuat penyesuaian berikut:
   - Pilih **HOLO_DARK** dari menu tarik-turun Theme untuk menyetel ikon menjadi putih dengan latar belakang berwarna gelap (atau hitam).
   - Bergantung pada bentuk ikon, Anda mungkin ingin menambahkan pengisi pada ikon sehingga ikon tidak memenuhi teks. Seret slider Padding ke kanan untuk menambahkan lapisan lebih banyak.
6. Klik **Next**, kemudian klik **Finish** dalam dialog Confirm Icon Path. Nama ikon seharusnya sekarang muncul di folder **app > res > drawable**.

Gambar vektor ikon standar diubah ukurannya secara otomatis untuk ukuran tampilan perangkat yang berbeda. Untuk memilih gambar vektor, ikuti langkah-langkah ini:

1. Luaskan **app > res** pada tampilan Project, dan klik-kanan (atau klik Command) **drawable**.
2. Pilih **New > Vector Asset** untuk ikon yang mengubah ukurannya sendiri secara otomatis untuk setiap tampilan.
3. Dialog Vector Asset Studio muncul untuk aset vektor. Klik tombol radio **Material Icon**, kemudian klik tombol **Choose** untuk memilih ikon dari spesifikasi Desain Material (lihat Tambahkan Grafik Vektor Multi-Kepadatan untuk keterangan lengkap mengenai dialog ini).
4. Klik **Next** setelah memilih ikon, dan klik **Finish** untuk menyelesaikan. Nama ikon seharusnya sekarang muncul di folder **app > res > drawable**.

Menambahkan tombol dengan teks dan ikon pada layout

Untuk membuat tombol dengan teks dan ikon seperti yang ditampilkan dalam gambar di bawah ini, gunakan `Button` dalam layout XML Anda. Tambahkan atribut `android:drawableLeft` untuk menggambar ikon di sebelah kiri teks tombol, seperti yang ditampilkan dalam gambar di bawah ini:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    android:drawableLeft="@drawable/button_icon"
    ... />
```

Membuat tombol timbul dengan ikon saja

Jika ikon dipahami secara universal, Anda mungkin ingin menggunakannya sebagai ganti teks.

Untuk membuat tombol timbul hanya dengan ikon atau gambar (tanpa teks), gunakan kelas ImageButton, yang memperluas kelas ImageView. Anda bisa menambahkan sebuah `ImageButton` ke layout XML Anda seperti berikut:

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/button_icon"
    ... />
```

Mengubah gaya dan penampilan tombol timbul

Cara termudah untuk menampilkan tombol timbul yang lebih menonjol adalah menggunakan warna latar belakang yang berbeda untuk tombol tersebut. Anda bisa menetapkan atribut `android:background` dengan sumber daya warna atau sumber daya dapat digambar:

```
android:background="@color/colorPrimary"
```

Penampilan tombol Anda—warna latar belakang dan font—mungkin bervariasi antara satu perangkat dengan perangkat lain, karena perangkat dari produsen yang berbeda sering kali memiliki gaya default berbeda untuk kontrol masukan. Anda bisa mengontrol dengan tepat gaya tombol dan kontrol masukan lainnya menggunakan *tema* yang Anda terapkan untuk seluruh aplikasi.

Misalnya, untuk memastikan bahwa semua perangkat yang bisa menjalankan tema Holo akan menggunakan tema Holo untuk aplikasi Anda, deklarasikan yang berikut ini dalam elemen `<application>` file AndroidManifest.xml:

```
android:theme="@android:style/Theme.Holo"
```

Setelah menambahkan deklarasi di atas, aplikasi akan ditampilkan menggunakan tema.

Aplikasi yang didesain untuk Android 4.0 dan yang lebih tinggi juga bisa menggunakan keluarga tema publik DeviceDefault. Tema DeviceDefault adalah nama lain untuk tampilan dan cara kerja perangkat. Keluarga tema dan keluarga gaya widget DeviceDefault menawarkan berbagai cara bagi developer untuk menargetkan tema asli perangkat dengan semua intact penyesuaian.

Untuk aplikasi Android yang menjalankan versi 4.0 dan yang lebih baru, Anda memiliki opsi berikut:

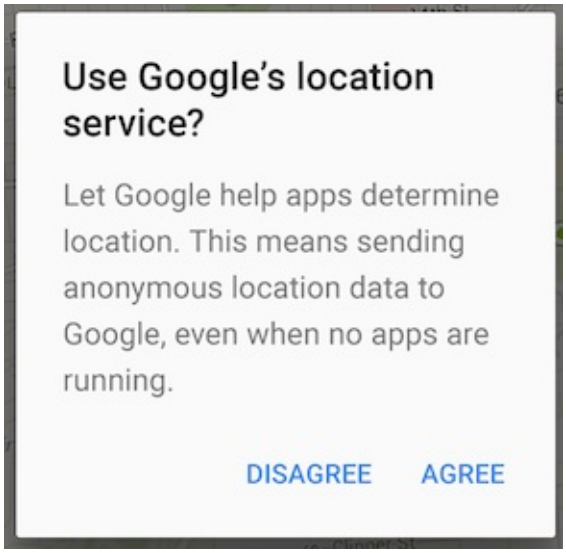
- Gunakan sebuah tema, misalnya salah satu tema Holo, sehingga aplikasi Anda memiliki penampilan yang sama di semua perangkat Android yang menjalankan versi 4.0 atau yang lebih baru. Dalam kasus ini, penampilan aplikasi tidak berubah saat dijalankan pada perangkat dengan kulit default atau kulit khusus yang berbeda.
- Gunakan salah satu tema DeviceDefault sehingga aplikasi Anda tampil dengan kulit default perangkat.
- Jangan gunakan tema, tetapi Anda mungkin akan mendapatkan hasil yang tidak dapat diprediksi pada beberapa perangkat.

Jika Anda tidak familier dengan sistem tema dan gaya Android, Anda harus membaca Gaya dan Tema. Entri blog "Holo Everywhere" menyediakan informasi tentang cara menggunakan tema Holo dengan tetap mendukung perangkat lama.

Sebagai panduan penataan gaya dan penyesuaian tombol menggunakan XML, lihat Tombol (di bagian "Antarmuka Pengguna" panduan developer Android). Untuk panduan komprehensif dalam mendesain tombol, lihat "Komponen - Tombol" dalam Spesifikasi Desain Material.

Mendesain tombol datar

Tombol datar, disebut juga dengan tombol tanpa bingkai, adalah tombol teks saja yang muncul datar di layar tanpa bayangan. Manfaat utama tombol datar adalah kemudahan — tombol ini meminimalkan gangguan dari materi. Tombol datar berguna saat Anda memiliki sebuah dialog, seperti yang ditampilkan dalam gambar di bawah ini, yang memerlukan interaksi atau masukan pengguna. Dalam hal ini, Anda mungkin perlu memiliki gaya dan font yang sama dengan teks sekitar tombol. Tombol ini menjaga tampilan dan cara kerja tetap sama di semua elemen dalam dialog.



Tombol datar, yang ditampilkan di bawah ini, menyerupai tombol dasar kecuali dalam hal tidak adanya bingkai atau latar belakang, tetapi tetap mengubah penampilan dalam keadaan yang berbeda. Sebuah tombol datar menampilkan bayangan tinta di sekitarnya bila ditekan (disentuh atau diklik).

Dalam gambar di atas:

1. Keadaan normal: Dalam keadaan normal, tombol tampak seperti teks pada umumnya.
2. Keadaan dinonaktifkan: Saat teks berwarna abu-abu, tombol tidak aktif dalam konteks aplikasi.
3. Keadaan ditekan: Keadaan ditekan, dengan bayangan latar belakang, menunjukkan bahwa tombol sedang disentuh atau diklik. Jika Anda memasang callback pada tombol tersebut (misalnya atribut `android:onClick`). callback akan dipanggil jika tombol dalam keadaan ini.
   Catatan: Jika Anda menggunakan tombol datar dalam sebuah layout, pastikan untuk menggunakan lapisan untuk membedakannya dari teks yang mengelilinginya, agar pengguna mudah melihatnya.

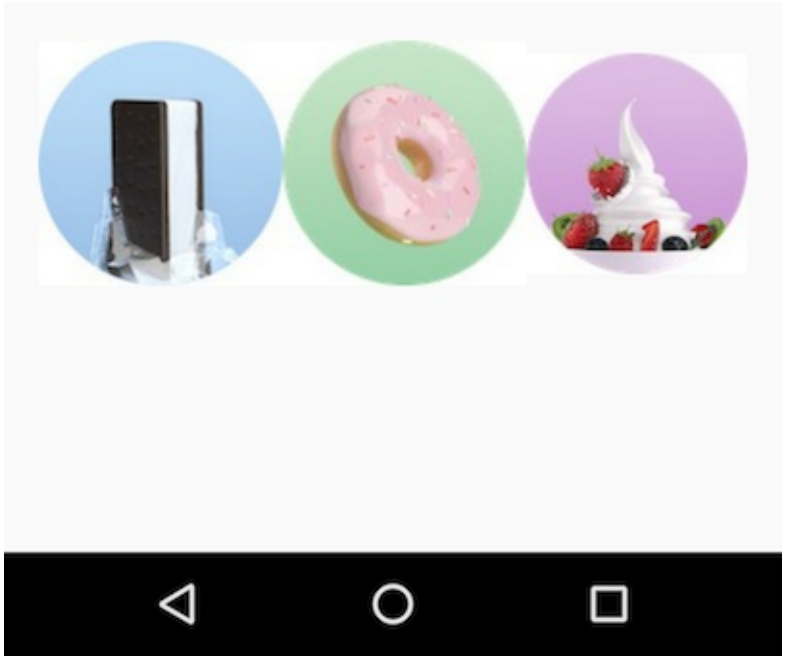
Untuk membuat tombol datar, gunakan kelas Button. Tambahkan sebuah `Button` ke layout XML Anda, dan terapkan `"?android:attr/borderlessButtonStyle"` sebagai atribut `style` :

```
<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" />
```

Mendesain gambar sebagai tombol

Anda bisa mengubah Tampilan apa pun, seperti ImageView, menjadi sebuah tombol dengan menambahkan atribut `android:onClick` dalam layout XML. Gambar untuk ImageView harus sudah disimpan dalam folder **drawables** proyek Anda.

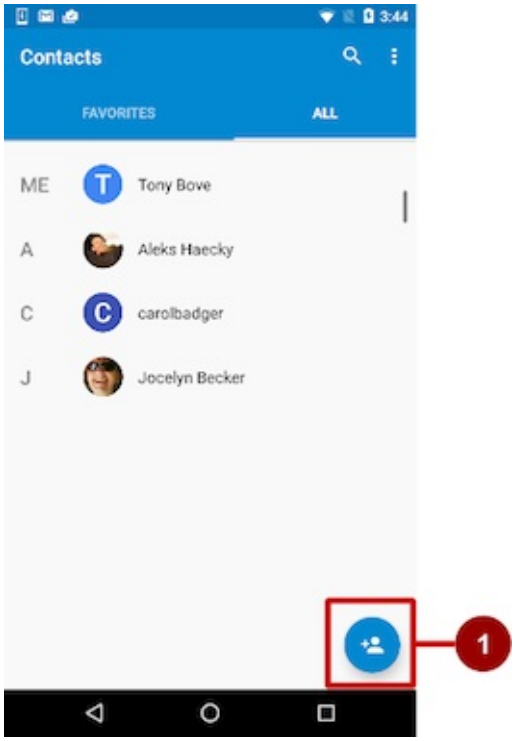
Catatan: Untuk memasukkan gambar ke dalam proyek Android Studio Anda, buat atau simpan gambar dalam format JPEG, lalu salin file gambar ke dalam folder **app > src > main > res > drawables** proyek Anda. Untuk informasi selengkapnya tentang sumber daya dapat digambar, lihat Sumber Daya Dapat Digambar dalam bagian Sumber Daya Aplikasi pada Panduan Developer Android.
Jika Anda menggunakan beberapa gambar sebagai tombol, atur gambar dalam sebuah viewgroup sehingga dikelompokkan bersama. Misalnya, gambar berikut dalam folder drawable (icecream_circle.jpg, donut_circle.jpg, dan froyo_circle.jpg) didefinisikan untuk ImageView yang dikelompokkan dalam LinearLayout dan disetel ke orientasi horizontal, sehingga gambar-gambar tersebut muncul bersebelahan:



```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal"
        android:layout_marginTop="260dp">
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/icecream_circle"
            android:onClick="orderIcecream"/>
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/donut_circle"
            android:onClick="orderDonut"/>
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/froyo_circle"
            android:onClick="orderFroyo"/>
    </LinearLayout>
```

Mendesain tombol aksi mengambang

Tombol aksi mengambang, yang ditampilkan di bawah ini sebagai #1 dalam gambar di bawah, adalah tombol bundar yang muncul mengambang di atas layout.



Anda harus menggunakan tombol aksi mengambang hanya untuk menyatakan aksi utama suatu layar. Misalnya, aksi utama untuk layar utama aplikasi Kontak adalah menambahkan kontak, seperti yang ditampilkan dalam gambar di atas. Tombol aksi mengambang adalah pilihan tepat jika aplikasi Anda mengharuskan suatu aksi yang persisten dan siap tersedia di layar. Hanya satu tombol aksi mengambang yang disarankan untuk setiap layar.

Tombol aksi mengambang menggunakan tipe ikon yang sama yang Anda gunakan untuk tombol dengan ikon, atau untuk tindakan pada bilah aplikasi di bagian atas layar. Anda bisa menambahkan sebuah ikon sebagaimana dijelaskan sebelumnya dalam "Memilih ikon untuk tombol".

Untuk menggunakan tombol aksi mengambang di proyek Android Studio, Anda harus menambahkan pernyataan berikut ke file **build.gradle (Modul: app)** di bagian `dependencies` :

```
compile 'com.android.support:design:23.4.0'
```

Catatan: Nomor versi di akhir pernyataan bisa berubah; gunakan versi terbaru yang disarankan Android Studio.
Untuk membuat tombol aksi mengambang, gunakan kelas FloatingActionButton, yang memperluas kelas ImageButton. Anda bisa menambahkan tombol aksi mengambang ke layout XML Anda seperti berikut:

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

Tombol aksi mengambang secara default berukuran 56 x 56 dp. Ini adalah ukuran default terbaik untuk digunakan, kecuali jika Anda membutuhkan versi yang lebih kecil untuk membuat kontinuitas visual dengan elemen layar lain.

Anda bisa menyetel ukuran *mini* (30 x 40) dengan atribut `app:fabSize` :

```
app:fabSize="mini"
```

Untuk menyetelnya kembali ke ukuran default (56 x 56 dp):

```
app:fabSize="normal"
```

Untuk petunjuk desain selengkapnya yang melibatkan tombol aksi mengambang, lihat Komponen–Tombol: Tombol Aksi Mengambang dalam Spesifikasi Desain Material.

Merespons kejadian klik-tombol

Gunakan *event listener* yang disebut OnClickListener, yaitu antarmuka kelas View, untuk merespons kejadian klik yang timbul saat pengguna mengetuk atau mengeklik objek dapat diklik, seperti Button, ImageButton, atau FloatingActionButton. Untuk informasi selengkapnya mengenai event listener, atau tipe kejadian UI lainnya, baca bagian Kejadian Masukan pada Dokumentasi Developer Android.

Menambahkan onClick pada elemen layout

Untuk mempersiapkan OnClickListener bagi objek yang dapat diklik dalam kode Aktivitas Anda dan menempatkan sebuah metode callback, gunakan atribut `android:onClick` dengan elemen objek yang dapat diklik dalam layout XML. Misalnya:

```
<Button
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

Dalam hal ini, bila pengguna mengeklik tombol, sistem Android akan memanggil metode `sendMessage()` Aktivitas:

```
public void sendMessage(View view) {
    // Do something in response to button click
}
```

Metode yang Anda deklarasikan sebagai atribut `android:onClick` harus `public` , mengembalikan `void` , dan mendefinisikan sebuah `View` sebagai satu-satunya parameter (yaitu tampilan yang diklik). Gunakan metode untuk menjalankan tugas atau memanggil metode lainnya sebagai respons terhadap klik tombol.

Menggunakan pola desain listener tombol

Anda juga bisa menangani kejadian klik lewat program dengan menggunakan pola desain tombol-listener (lihat gambar di bawah ini). Untuk informasi selengkapnya mengenai pola desain "listener", lihat Membuat Listener Khusus.

Gunakan event listener View.OnClickListener, yaitu antarmuka pada kelas View yang terdiri dari metode callback tunggal, onClick(). Metode ini dipanggil oleh kerangka kerja Android jika tampilan dipicu oleh interaksi pengguna.

Event listener harus sudah didaftarkan pada tampilan agar dipanggil jika ada kejadian. Ikuti langkah-langkah ini untuk mendaftarkan listener dan menggunakannya (lihat gambar di bawah langkah-langkah):

1. Gunakan metode findViewById() dari kelas View untuk menemukan tombol dalam file layout XML:

   ```
   Button button = (Button) findViewById(R.id.button_send);
   ```

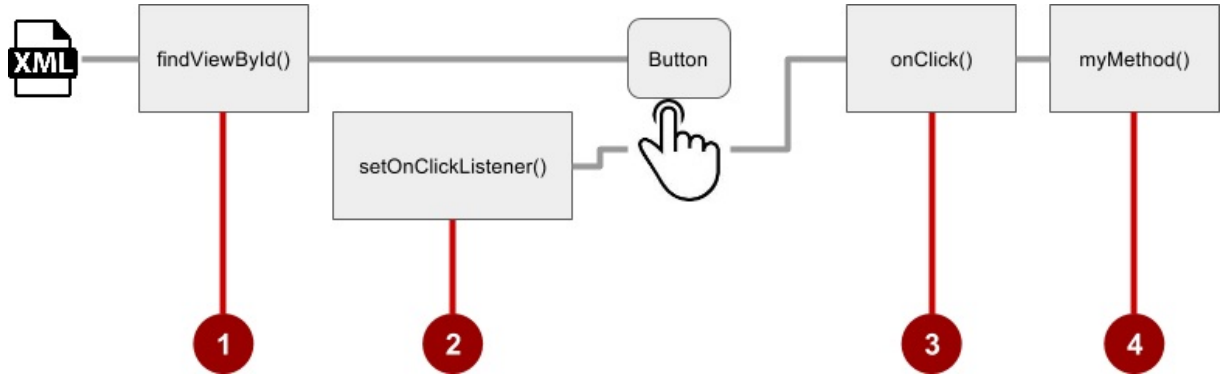
2. Dapatkan objek `View.OnClickListener` yang baru dan daftarkan ke tombol dengan memanggil metode setOnClickListener(). Argumen untuk `setOnClickListener()` memerlukan objek yang mengimplementasikan antarmuka View.OnClickListener, yang memiliki satu metode: `onClick()` .

   ```
   button.setOnClickListener(new View.OnClickListener() {
   ...
   ```

3. Definisikan metode `onClick()` sebagai `public` , kembalikan `void` , dan definisikan `View` sebagai parameternya satu-satunya:

   ```
   public void onClick(View v) {
       // Do something in response to button click
   }
   ```

4. Buat metode untuk melakukan sesuatu sebagai respons terhadap klik tombol, misalnya melakukan suatu aksi.



Untuk menyetel listener klik lewat program, sebagai ganti dengan atribut onClick, sesuaikan kelas View.OnClickListener dan ganti penangan `onClick()` untuk melakukan beberapa aksi, seperti yang ditampilkan di bawah ini:

```
fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Add a new word to the wordList.
            }
        });
```

Menggunakan antarmuka event listener untuk kejadian lain

Kejadian lainnya bisa terjadi dengan elemen UI, dan Anda bisa menggunakan metode callback yang sudah didefinisikan dalam antarmuka event listener untuk menanganinya. Metode tersebut dipanggil oleh kerangka kerja Android jika tampilan —yang sudah didaftarkan listener—dipicu oleh interaksi pengguna. Oleh karena itu, Anda harus menyetel listener yang sesuai untuk menggunakan metode tersebut. Berikut ini adalah beberapa listener yang tersedia di kerangka kerja Android dan metode callback yang terkait:

- `onClick()` dari View.OnClickListener: Menangani kejadian klik di mana pengguna menyentuh dan kemudian melepas area tampilan perangkat yang dikuasai oleh suatu tampilan. Callback onClick() tidak memiliki nilai kembalian.
- `onLongClick()` dari View.OnLongClickListener: Menangani suatu kejadian di mana pengguna menahan sentuhan pada sebuah tampilan untuk waktu yang lama. Ini akan mengembalikan boolean untuk menunjukkan apakah Anda telah menggunakan kejadian dan tidak boleh dilakukan lebih jauh. Yaitu, kembalikan `true` untuk menunjukkan bahwa Anda telah menangani kejadian dan harus berhenti di sini; kembalikan `false` jika Anda belum menanganinya dan/atau kejadian harus berlanjut ke on-click listener lainnya.
- `onTouch()` dari View.OnTouchListener: Menangani segala bentuk kontak sentuh dengan layar termasuk satu atau beberapa gerakan sentuh dan gerakan isyarat, termasuk isyarat menekan, melepas, atau gerakan lain pada layar (dalam batas elemen UI). Suatu MotionEvent diteruskan sebagai argumen, yang menyertakan informasi arah, dan mengembalikan boolean untuk menunjukkan apakah listener Anda menggunakan kejadian ini.
- `onFocusChange()` dari View.OnFocusChangeListener: Menangani saat fokus menjauh dari tampilan saat ini sebagai hasil interaksi dengan trackball atau tombol navigasi.
- `onKey()` dari View.OnKeyListener: Menangani bila sebuah tombol perangkat keras ditekan saat tampilan memiliki fokus.

Menggunakan kontrol masukan untuk membuat pilihan

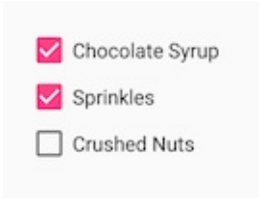
Android menawarkan kontrol masukan siap-buat bagi pengguna untuk memilih satu atau beberapa pilihan:

- Kotak centang: Pilih satu atau beberapa nilai dari serangkaian nilai dengan mengeklik setiap kotak centang nilai.
- Tombol radio: Pilih satu nilai saja dari serangkaian nilai dengan mengeklik tombol "radio" bundar nilai. Jika hanya menyediakan dua atau tiga pilihan, Anda mungkin ingin menggunakan tombol radio untuk pilihan jika memiliki ruang di layout untuk tombol-tombol itu.
- Tombol beralih: Pilih salah satu dari dua atau beberapa keadaan: Tombol beralih biasanya menawarkan dua keadaan yang tampak, seperti "aktif" dan "nonaktif".
- Spinner: Pilih satu nilai dari serangkaian nilai di menu tarik-turun. Hanya satu nilai yang bisa dipilih. Spinner berguna untuk tiga atau beberapa pilihan, dan membutuhkan sedikit ruang di layout Anda.

Kotak centang

Gunakan kotak centang jika Anda memiliki daftar pilihan dan pengguna boleh memilih *berapa pun* pilihan, termasuk tanpa pilihan. Setiap kotak centang bersifat independen dari kotak centang lain pada daftar tersebut, sehingga mencentang satu kotak tidak akan menghapus centang pada kotak lainnya. (Jika Anda ingin membatasi pilihan pengguna menjadi satu item saja dari satu rangkaian, gunakan tombol radio.) Pengguna bisa juga menghapus centang pada kotak centang.

Pengguna mengharapkan kotak centang muncul dalam daftar vertikal, seperti daftar kerja, atau bersebelahan secara horizontal jika labelnya pendek.



Setiap kotak centang adalah instance tersendiri dari kelas CheckBox. Anda membuat setiap kotak centang menggunakan elemen `CheckBox` dalam layout XML Anda. Untuk membuat beberapa kotak centang dengan orientasi vertikal, gunakan LinearLayout vertikal:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">

   <CheckBox android:id="@+id/checkbox1_chocolate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/chocolate_syrup" />
    <CheckBox android:id="@+id/checkbox2_sprinkles"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/sprinkles" />
    <CheckBox android:id="@+id/checkbox3_nuts"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/crushed_nuts" />

</LinearLayout>
```

Umumnya, program mengambil keadaan kotak centang saat pengguna menyentuh atau mengeklik tombol **Submit** atau **Done** pada aktivitas yang sama, yang menggunakan atribut `android:onClick` untuk memanggil metode seperti `onSubmit()`:

```
<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/submit"
   android:onClick="onSubmit"/>
```

Metode callback— `onSubmit()` dalam contoh `Button` di atas—harus `public`, mengembalikan `void`, dan mendefinisikan sebuah `View` sebagai parameter (tampilan yang diklik). Dalam metode ini, Anda bisa menentukan apakah kotak centang dipilih dengan menggunakan metode isChecked() (diwarisi dari CompoundButton). Metode `isChecked()` akan mengembalikan (boolean) `true` jika ada tanda centang dalam kotak. Misalnya, pernyataan berikut menetapkan nilai boolean dari `true` atau `false` ke `checked` bergantung pada apakah kotak centang itu dicentang:

```
boolean checked = ((CheckBox) view).isChecked();
```

Cuplikan kode berikut menampilkan bagaimana metode `onSubmit()` akan memeriksa untuk mengetahui kotak centang mana yang dipilih, dengan menggunakan `id` sumber daya untuk elemen kotak centang:

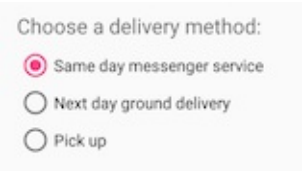
```
public void onSubmit(View view){
   StringBuffer toppings = new
                  StringBuffer().append(getString(R.string.toppings_label));
   if  (((CheckBox) findViewById(R.id.checkbox1_chocolate)).isChecked()) {
      toppings.append(getString(R.string.chocolate_syrup_text));
   }
   if  (((CheckBox) findViewById(R.id.checkbox2_sprinkles)).isChecked()) {
      toppings.append(getString(R.string.sprinkles_text));
   }
   if  (((CheckBox) findViewById(R.id.checkbox3_nuts)).isChecked()) {
      toppings.append(getString(R.string.crushed_nuts_text));
   }
...
}
```

Tip: Untuk merespons kotak centang dengan cepat—seperti menampilkan pesan (semacam peringatan), atau menampilkan serangkaian opsi lebih jauh—Anda bisa menggunakan atribut `android:onClick` dalam layout XML untuk setiap kotak centang untuk mendeklarasikan metode callback bagi kotak centang tersebut, yang harus didefinisikan dalam aktivitas yang menjadi host layout ini.

Untuk informasi selengkapnya tentang kotak centang, lihat Kotak centang di bagian Antarmuka Pengguna dalam Dokumentasi Developer Android.

Tombol radio

Gunakan tombol radio jika Anda menggunakan dua atau beberapa opsi yang saling lepas—pengguna harus memilih salah satu di antaranya. (Jika Anda ingin mengaktifkan lebih dari satu pilihan dari rangkaian tersebut, gunakan kotak centang.)



Pengguna mengharapkan tombol radio muncul dalam bentuk daftar vertikal, atau bersebelahan secara horizontal jika labelnya pendek.

Setiap tombol radio adalah instance kelas RadioButton. Tombol radio umumnya digunakan bersama dalam RadioGroup. Bila sejumlah tombol radio berada dalam suatu grup radio, mencentang satu tombol radio akan menghapus semua centang lainnya. Buat setiap tombol radio menggunakan elemen RadioButton pada layout XML Anda dalam grup tampilan RadioButton:

```
<RadioGroup
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_below="@id/orderintrotext">
        <RadioButton
            android:id="@+id/sameday"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/same_day_messenger_service"
            android:onClick="onRadioButtonClicked"/>
        <RadioButton
            android:id="@+id/nextday"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/next_day_ground_delivery"
            android:onClick="onRadioButtonClicked"/>
        <RadioButton
            android:id="@+id/pickup"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/pick_up"
            android:onClick="onRadioButtonClicked"/>
    </RadioGroup>
```

Gunakan atribut `android:onClick` setiap tombol radio untuk mendeklarasikan metode penangan kejadian klik bagi tombol radio, yang harus didefinisikan dalam aktivitas yang menjadi host layout ini. Dalam layout di atas, mengeklik tombol radio akan memanggil metode `onRadioButtonClicked()` yang sama dalam aktivitas tersebut, namun Anda bisa membuat metode tersendiri dalam aktivitas dan mendeklarasikannya dalam setiap atribut `android:onClick` tombol radio.

Metode penangan kejadian klik harus `public`, mengembalikan `void`, dan mendefinisikan sebuah `View` sebagai satu-satunya parameter (tampilan yang diklik). Yang berikut ini menampilkan satu metode, `onRadioButtonClicked()`, untuk semua tombol radio, dengan menggunakan pernyataan `switch case` untuk memeriksa `id` sumber daya bagi elemen tombol radio untuk menentukan tombol radio yang dicentang:

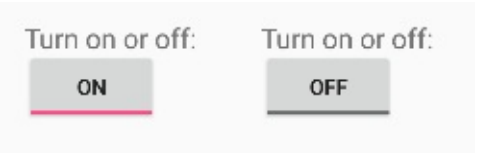
```
public void onRadioButtonClicked(View view) {
   // Check to see if a button has been clicked.
   boolean checked = ((RadioButton) view).isChecked();

   // Check which radio button was clicked.
   switch(view.getId()) {
      case R.id.sameday:
         if (checked)
            // Same day service
            break;
      case R.id.nextday:
         if (checked)
            // Next day delivery
            break;
      case R.id.pickup:
         if (checked)
            // Pick up
            break;
   }
}
```

Tip: Untuk memberi kesempatan pengguna meninjau pilihan tombol radio mereka sebelum aplikasi merespons, Anda bisa mengimplementasikan tombol **Submit** atau **Done** seperti yang ditampilkan sebelumnya bersama kotak centang, dan membuang atribut `android:onClick` dari tombol radio. Kemudian, tambahkan metode `onRadioButtonClicked()` ke atribut `android:onClick` untuk tombol **Submit** atau **Done**.

Untuk informasi selengkapnya tentang tombol radio, baca "Tombol Radio" pada bagian Antarmuka Pengguna dalam Dokumentasi Developer Android.

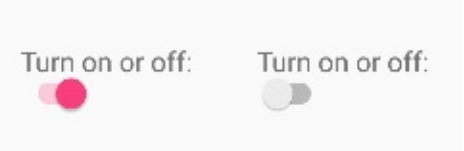
Switch dan tombol beralih

Kontrol masukan beralih memungkinkan pengguna mengubah setelan di antara dua keadaan. Android menyediakan kelas ToggleButton, yang menampilkan tombol timbul dengan "OFF" dan "ON".



Contoh beralih menyertakan switch Aktif/Nonaktif untuk Wi-Fi, Bluetooth, dan opsi lain dalam aplikasi Settings.

Android juga menyediakan kelas Switch, yaitu slider pendek yang terlihat seperti sakelar bulat yang menawarkan dua keadaan (hidup dan mati). Keduanya adalah ekstensi kelas CompoundButton.



Menggunakan tombol beralih

Membuat tombol beralih dengan menggunakan elemen `ToggleButton` dalam layout XML Anda:

```
<ToggleButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/my_toggle"
        android:text="
        android:onClick="onToggleClick"/>
```

Tip: Atribut android:text tidak menyediakan label teks untuk tombol beralih—tombol beralih selalu menampilkan baik "ON" atau "OFF". Untuk menyediakan label teks di sebelah (atau di atas) tombol beralih, gunakan `TextView` tersendiri.

Untuk merespons ketukan beralih, deklarasikan metode callback `android:onClick` bagi `ToggleButton` . Metode harus didefinisikan dalam aktivitas yang melakukan hosting layout, dan itu harus berupa `public` , mengembalikan `void` , dan mendefinisikan sebuah `View` sebagai satu-satunya parameter (yaitu tampilan yang diklik). Gunakan CompoundButton.OnCheckedChangeListener() untuk mendeteksi perubahan keadaan beralih. Buat objek `CompoundButton.OnCheckedChangeListener` dan tetapkan ke tombol dengan memanggil `setOnCheckedChangeListener()` . Misalnya, metode `onToggleClick()` akan memeriksa apakah peralihan aktif atau nonaktif, dan menampilkan pesan toast:

```
public void onToggleClick(View view) {
   ToggleButton toggle = (ToggleButton) findViewById(R.id.my_toggle);
   toggle.setOnCheckedChangeListener(new
                              CompoundButton.OnCheckedChangeListener() {
      public void onCheckedChanged(CompoundButton buttonView,
                              boolean isChecked) {
         StringBuffer onOff = new StringBuffer().append("On or off? ");
         if (isChecked) { // The toggle is enabled
            onOff.append("ON ");
         } else { // The toggle is disabled
            onOff.append("OFF ");
         }
         Toast.makeText(getApplicationContext(), onOff.toString(),
                              Toast.LENGTH_SHORT).show();
      }
   });
}
```

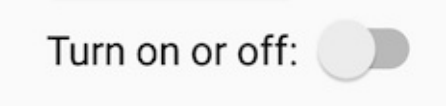
Tip: Anda juga bisa mengubah keadaan ToggleButton lewat program dengan menggunakan metode setChecked(boolean). Akan tetapi, perhatikan, metode yang ditetapkan oleh atribut `android:onClick()` tidak akan dieksekusi dalam kasus ini.

Menggunakan switch

Switch adalah instance tersendiri dari kelas Switch, yang memperluas kelas CompoundButton seperti ToggleButton. Buat switch beralih dengan menggunakan elemen `Switch` dalam layout XML Anda:

```
<Switch
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/my_switch"
        android:text="@string/turn_on_or_off"
        android:onClick="onSwitchClick"/>
```

Atribut android:text mendefinisikan sebuah string yang muncul di sebelah kiri switch, seperti yang ditampilkan di bawah ini:



Untuk merespons ketukan switch, deklarasikan metode callback `android:onClick` bagi `Switch` —kode pada dasarnya sama dengan kode untuk `ToggleButton` . Metode harus didefinisikan dalam aktivitas yang melakukan hosting layout, dan itu harus berupa `public` , mengembalikan `void` , dan mendefinisikan sebuah `View` sebagai satu-satunya parameter (yaitu tampilan yang diklik). Gunakan CompoundButton.OnCheckedChangeListener() untuk mendeteksi perubahan keadaan switch. Buat objek `CompoundButton.OnCheckedChangeListener` dan tetapkan ke tombol dengan memanggil `setOnCheckedChangeListener()` . Misalnya, metode `onSwitchClick()` akan memeriksa apakah switch aktif atau nonaktif, dan menampilkan pesan toast:

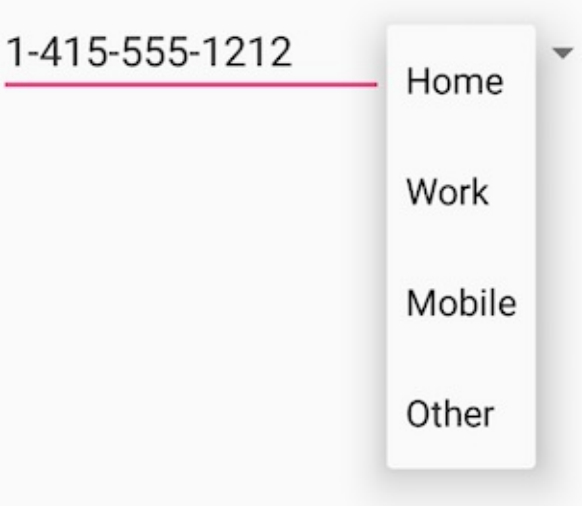
```
public void onSwitchClick(View view) {
   Switch aSwitch = (Switch) findViewById(R.id.my_switch);
   aSwitch.setOnCheckedChangeListener(new
                           CompoundButton.OnCheckedChangeListener() {
      public void onCheckedChanged(CompoundButton buttonView,
                           boolean isChecked) {
         StringBuffer onOff = new StringBuffer().append("On or off? ");
         if (isChecked) { // The switch is enabled
            onOff.append("ON ");
         } else { // The switch is disabled
            onOff.append("OFF ");
         }
         Toast.makeText(getApplicationContext(), onOff.toString(),
                           Toast.LENGTH_SHORT).show();
      }
   });
}
```

Tip: Anda juga bisa mengubah keadaan Switch lewat program dengan menggunakan metode setChecked(boolean). Akan tetapi, perhatikan, metode yang ditetapkan oleh atribut `android:onClick()` tidak akan dieksekusi dalam kasus ini.

Untuk informasi selengkapnya tentang beralih, lihat "Tombol Beralih" di bagian Antarmuka Pengguna dalam Dokumentasi Developer Android.

Spinner

Spinner menyediakan cara cepat untuk memilih salah satu dari serangkaian nilai. Menyentuh spinner akan menampilkan daftar tarik-turun dengan semua nilai yang tersedia, yang bisa dipilih oleh pengguna.



Jika Anda memiliki daftar pilihan yang panjang, spinner bisa memanjang melebihi layout, sehingga mengharuskan pengguna untuk menggulirnya. Spinner menggulir secara otomatis, tanpa memerlukan tambahan kode. Akan tetapi, menggulir daftar panjang (seperti daftar negara) tidak disarankan karena akan menyulitkan untuk memilih sebuah item.

Untuk membuat spinner, gunakan kelas Spinner, yang membuat tampilan yang menampilkan nilai spinner individual sebagai tampilan anak, dan memungkinkan pengguna memilih salah satu. Ikuti langkah-langkah ini:

1. Buat elemen `Spinner` dalam layout XML Anda, dan tetapkan nilainya menggunakan larik serta ArrayAdapter.
2. Buat spinner dan adapternya menggunakan kelas SpinnerAdapter.
3. Untuk mendefinisikan callback pilihan bagi spinner, perbarui Aktivitas yang menggunakan spinner untuk mengimplementasikan antarmuka AdapterView.OnItemSelectedListener.

Buat elemen UI spinner

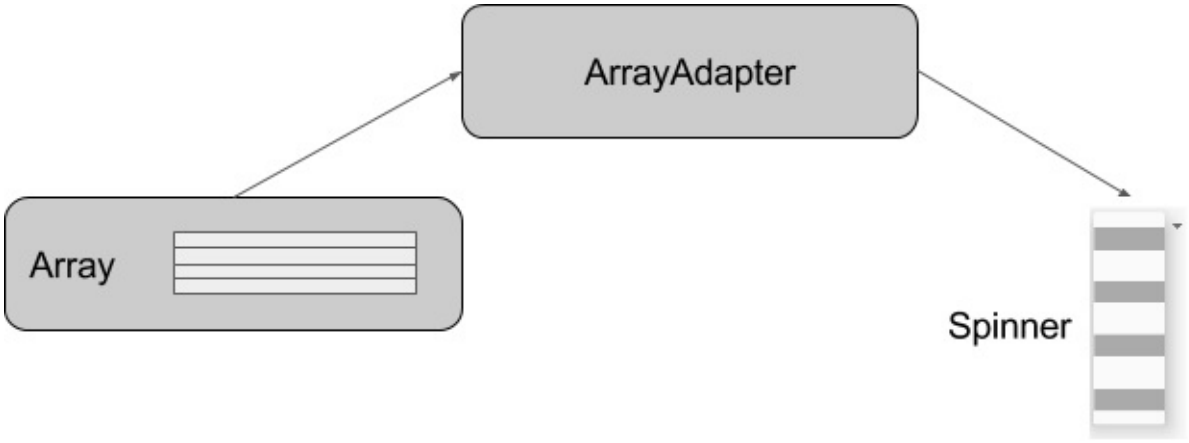
Untuk membuat spinner dalam layout XML Anda, tambahkan elemen Spinner, yang menyediakan daftar tarik-turun:

```
<Spinner
   android:id="@+id/label_spinner"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content">
</Spinner>
```

Tetapkan nilai spinner

Tambahkan sebuah adapter yang mengisi daftar spinner dengan nilai-nilai. Adapter bagaikan jembatan, atau perantara, antara dua antarmuka yang tidak kompatibel. Misalnya, pembaca kartu memori berfungsi sebagai adapter antara kartu memori dan laptop. Anda memasang kartu memori ke dalam pembaca kartu, dan memasang pembaca kartu ke dalam laptop, sehingga laptop bisa membaca kartu memori.

Pola spinner-adapter mengambil serangkaian data yang Anda tetapkan dan membuat tampilan untuk setiap item dalam rangkaian data tersebut, seperti yang ditampilkan dalam gambar di bawah ini.



Kelas SpinnerAdapter, yang mengimplementasikan kelas Adapter, memungkinkan Anda mendefinisikan dua tampilan yang berbeda: yang menampilkan nilai data dalam spinner itu sendiri, dan yang menampilkan data dalam daftar tarik-turun saat spinner disentuh atau diklik.

Nilai yang Anda sediakan untuk spinner bisa berasal dari sumber apa pun, tetapi harus disediakan melalui SpinnerAdapter, seperti ArrayAdapter jika nilainya tersedia dalam larik. Contoh berikut menampilkan larik sederhana yang disebut `labels_array` nilai pra-determinan dalam file strings.xml:

```
<string-array name="labels_array">
        <item>Home</item>
        <item>Work</item>
        <item>Mobile</item>
        <item>Other</item>
</string-array>
```

Tip: Anda bisa menggunakan CursorAdapter jika nilai berasal dari sebuah sumber seperti file tersimpan atau database. Anda akan mengetahui tentang data tersimpan dalam bab lain.

Buat spinner dan adapternya

Buat spinner, dan setel listener-nya ke aktivitas yang mengimplementasikan metode callback. Tempat terbaik untuk melakukannya adalah saat tampilan dibuat dalam metode `onCreate()` . Ikuti langkah-langkah ini (lihat metode `onCreate()` lengkap di akhir langkah):

1. Tambahkan kode di bawah ke metode `onCreate()` , yang melakukan hal-hal berikut:
2. Mendapatkan objek spinner yang Anda tambahkan pada layout menggunakan `findViewById()` untuk menemukannya dengan `id` -nya (`label_spinner`).
3. Menyetel onItemSelectedListener ke aktivitas mana pun yang mengimplementasikan callback (`this`) dengan menggunakan metode setOnItemSelectedListener().

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

   // Create the spinner.
   Spinner spinner = (Spinner) findViewById(R.id.label_spinner);
   if (spinner != null) {
      spinner.setOnItemSelectedListener(this);
   }
}
```

4. Juga di metode `onCreate()`, tambahkan pernyataan yang membuat ArrayAdapter dengan larik string:

   ```
   // Create ArrayAdapter using the string array and default spinner layout.
   ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
               R.array.labels_array, android.R.layout.simple_spinner_item);
   ```

   Seperti yang ditampilkan di atas, Anda menggunakan metode createFromResource(), yang memerlukan argumen:

5. Aktivitas yang mengimplementasikan callback untuk memproses hasil spinner (`this`)
6. Larik (`labels_array`)
7. Layout untuk setiap item spinner (`layout.simple_spinner_item`).

   Tip: Anda harus menggunakan layout default `simple_spinner_item` , kecuali jika Anda ingin mendefinisikan layout sendiri untuk item di spinner.

8. Tetapkan layout yang harus digunakan adapter untuk menampilkan pilihan daftar spinner dengan memanggil metode setDropDownViewResource() dari kelas ArrayAdapter. Misalnya, Anda bisa menggunakan `simple_spinner_dropdown_item` sebagai layout Anda:

   ```
   // Specify the layout to use when the list of choices appears.
   adapter.setDropDownViewResource
                     (android.R.layout.simple_spinner_dropdown_item);
   ```

   Tip: Anda harus menggunakan layout default `simple_spinner_dropdown_item` , kecuali jika Anda ingin mendefinisikan layout sendiri untuk penampilan spinner.

9. Gunakan setAdapter() untuk menerapkan adapter ke spinner:

   ```
   // Apply the adapter to the spinner.
   spinner.setAdapter(adapter);
   ```

Kode lengkap untuk metode `onCreate()` ditampilkan di bawah ini:

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

   // Create the spinner.
   Spinner spinner = (Spinner) findViewById(R.id.label_spinner);
   if (spinner != null) {
            spinner.setOnItemSelectedListener(this);
   }

   // Create ArrayAdapter using the string array and default spinner layout.
   ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
                R.array.labels_array, android.R.layout.simple_spinner_item);

   // Specify the layout to use when the list of choices appears.
   adapter.setDropDownViewResource
                (android.R.layout.simple_spinner_dropdown_item);

   // Apply the adapter to the spinner.
   if (spinner != null) {
            spinner.setAdapter(adapter);
   }
}
```

Implementasikan antarmuka OnItemSelectedListener dalam Aktivitas

Untuk mendefinisikan callback pilihan spinner, perbarui Aktivitas yang menggunakan spinner untuk mengimplementasikan antarmuka AdapterView.OnItemSelectedListener.

```
public class MainActivity extends AppCompatActivity implements
            AdapterView.OnItemSelectedListener {
```

Android Studio secara otomatis mengimpor widget AdapterView. Implementasikan antarmuka `AdapterView.OnItemSelectedListener` untuk mendapatkan metode callback `onItemSelected()` dan `onNothingSelected()` yang akan digunakan bersama objek spinner.

Saat pengguna memilih satu item dari daftar tarik-turun spinner, inilah yang akan terjadi dan cara mengambil item tersebut:

1. Objek `Spinner` menerima kejadian on-item-selected.
2. Kejadian ini memicu panggilan metode callback onItemSelected() dari antarmuka AdapterView.OnItemSelectedListener.
3. Ambil item yang dipilih dalam menu spinner dengan menggunakan metode getItemAtPosition() kelas AdapterView:

   ```
   public void onItemSelected(AdapterView<?> adapterView, View view, int
                pos, long id) {
      String spinner_item = adapterView.getItemAtPosition(pos).toString();
   }
   ```

   Argumen untuk `onItemSelected()` adalah seperti berikut:

`parent AdapterView`	AdapterView di mana pemilihan terjadi
`view View`	Tampilan dalam AdapterView yang diklik
`int pos`	Posisi tampilan dalam adapter
`long id`	ID baris item yang dipilih

4. Implementasikan/ganti metode callback onNothingSelected() dari antarmuka AdapterView.OnItemSelectedListener

untuk melakukan sesuatu jika tidak ada yang dipilih.

Untuk informasi selengkapnya tentang spinner, baca Spinner di bagian Antarmuka Pengguna dalam Dokumentasi Developer Android.

Masukan teks

Gunakan kelas EditText untuk mendapatkan masukan pengguna yang terdiri dari karakter tekstual, termasuk angka dan simbol. EditText memperluas kelas TextView, agar TextView dapat diedit.

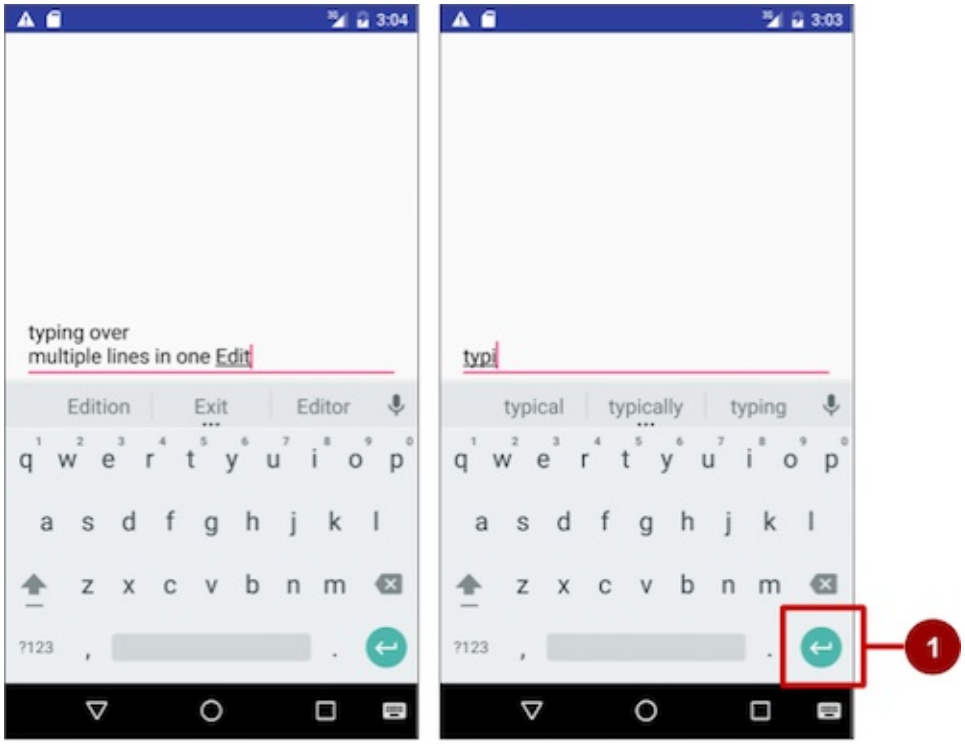
Menyesuaikan objek EditText untuk masukan pengguna

Dalam Layout Manager Android Studio, buat tampilan EditText dengan menambahkan sebuah `EditText` ke layout Anda dengan XML berikut:

```
<EditText
    android:id="@+id/edit_simple"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
</EditText>
```

Mengaktifkan beberapa baris masukan

Secara default, tampilan EditText memungkinkan beberapa baris masukan seperti yang ditampilkan dalam gambar di bawah ini, dan menyarankan koreksi ejaan. Mengetuk tombol Return (disebut juga dengan Enter) pada keyboard di layar mengakhiri suatu baris dan memulai baris baru dalam tampilan EditText yang sama.



Catatan: Dalam gambar di atas, #1 adalah tombol Return (disebut juga dengan Enter).

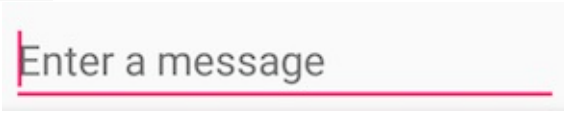
Memungkinkan Return untuk lanjut ke tampilan berikutnya

Jika Anda menambahkan atribut `android:inputType` ke tampilan EditText dengan nilai seperti `"textCapCharacters"` (untuk mengubah masukan menjadi huruf besar semua) atau `"textAutoComplete"` (untuk mengaktifkan saran ejaan saat pengguna mengetik), mengetuk tombol Return akan menutup keyboard di layar dan melanjutkan fokus ke tampilan

berikutnya. Perilaku ini berguna jika Anda ingin pengguna mengisi suatu formulir yang terdiri dari bidang EditText, sehingga pengguna bisa melanjutkan dengan cepat ke tampilan EditText berikutnya.

Atribut untuk menyesuaikan tampilan EditText

Gunakan atribut untuk menyesuaikan tampilan EditText untuk masukan. Misalnya:

- `android:maxLines="1"` : Setel entri teks untuk menampilkan satu baris saja.
- `android:lines="2"` : Setel entri teks untuk menampilkan 2 baris, bahkan jika panjang teksnya kurang.
- `android:maxLength="5"` : Setel jumlah karakter masukan maksimum ke 5.
- `android:inputType="number"` : Batasi entri teks untuk angka.
- `android:digits="01"` : Batasi digit yang dimasukkan hanya "0" dan "1".
- `android:textColorHighlight="#7cff88"` : Setel warna latar belakang teks yang dipilih (disorot).
- `android:hint="@string/my_hint"` : Setel teks untuk muncul dalam bidang yang menyediakan petunjuk untuk pengguna, seperti "Masukkan pesan". 

Untuk daftar atribut EditText, termasuk atribut TextView yang diwarisi, lihat "Rangkuman" keterangan kelas EditText.

Mendapatkan masukan pengguna

Memungkinkan pengguna untuk memasukkan teks hanya berguna jika Anda bisa *menggunakan* teks itu dalam beberapa cara dalam aplikasi. Untuk menggunakan masukan, Anda terlebih dahulu harus mendapatkannya dari tampilan EditText dalam layout XML. Langkah-langkah yang bisa Anda ikuti untuk mempersiapkan tampilan EditText dan mendapatkan masukan pengguna adalah:

1. Buat elemen tampilan EditText di layout XML untuk suatu aktivitas. Pastikan untuk mengidentifikasi elemen ini dengan sebuah `android:id` sehingga Anda bisa merujuknya dengan `id` :

   ```
   android:id="@+id/editText_main"
   ```

2. Dalam kode Java untuk aktivitas yang sama, buat sebuah metode dengan parameter `View` yang mengambil objek `EditText` (dalam contoh di bawah, `editText`) untuk tampilan `EditText` , dengan menggunakan metode findViewById() kelas View untuk menemukan tampilan melalui ID-nya (`editText_main`):

   ```
   EditText editText = (EditText) findViewById(R.id.editText_main);
   ```

3. Gunakan metode getText() dari kelas EditText (diwarisi dari kelas TextView) untuk mendapatkan teks sebagai urutan karakter (CharSequence). Anda bisa mengonversi urutan karakter menjadi string dengan menggunakan metode toString() dari kelas CharSequence, yang mengembalikan string yang menyatakan data dalam urutan karakter.

   ```
   String showString = editText.getText().toString();
   ```

Tip: Anda bisa menggunakan metode valueOf() dari kelas Integer untuk mengonversi string menjadi integer jika masukan adalah sebuah integer.

Mengubah perilaku keyboard dan masukan

Sistem Android menampilkan keyboard di layar—disebut dengan metode masukan *lunak*—jika sebuah bidang teks dalam UI menerima fokus. Untuk menyediakan pengalaman pengguna terbaik, Anda bisa menetapkan karakteristik tipe masukan yang diharapkan aplikasi, misalnya apakah berupa nomor telepon atau alamat email. Anda juga bisa menetapkan perilaku metode masukan, misalnya apakah menampilkan saran ejaan atau menyediakan huruf besar di awal kalimat. Anda bisa mengubah metode masukan lunak menjadi keypad numerik untuk memasukkan angka saja, atau bahkan keypad telepon untuk nomor telepon.

Android juga menyediakan kerangka kerja yang bisa diperluas bagi programmer tingkat lanjut untuk mengembangkan dan memasang Input Method Editor (IME) mereka sendiri untuk masukan ucapan, entri keyboard tipe khusus, dan aplikasi lain.

Deklarasikan metode masukan dengan menambahkan atribut android:inputType ke tampilan EditText. Misalnya, atribut berikut menyetel keyboard di layar menjadi keypad telepon:

```
android:inputType="phone"
```

Gunakan atribut android:inputType dengan nilai berikut:

- `textCapSentences` : Setel keyboard untuk menjadikan huruf besar di awal kalimat.
- `textAutoCorrect` : Aktifkan saran ejaan saat pengguna mengetik.
- `textPassword` : Ubah setiap karakter yang dimasukkan pengguna menjadi titik untuk menyembunyikan sandi yang dimasukkan.
- `textEmailAddress` : Untuk entri email, tampilkan keyboard email dengan simbol "@" yang ditempatkan agar mudah digunakan di sebelah tombol spasi.
- `phone` : Untuk entri nomor telepon, tampilkan keypad telepon numerik.

Tip: Anda bisa menggunakan karakter pipa `(|)` (Java bitwise OR) untuk mengombinasikan nilai atribut `android:inputType` :

```
android:inputType="textAutoCorrect|textCapSentences"
```

Untuk detail tentang atribut `android:inputType` baca Menetapkan Tipe Metode Masukan dalam dokumentasi developer. Untuk daftar lengkap nilai konstanta android:inputType, lihat bagian "android:inputType" di dokumentasi TextView.

Mengubah tombol "aksi" pada keyboard

Pada perangkat Android, tombol "aksi" adalah tombol **Return**. Tombol ini umumnya digunakan untuk memasukkan baris teks lain untuk elemen EditText yang memungkinkan beberapa baris. Jika Anda menyetel atribut `android:inputType` untuk tampilan EditText dengan nilai seperti `"textCapCharacters"` (untuk mengubah masukan menjadi huruf besar semua) atau `"textAutoComplete"` (untuk mengaktifkan saran ejaan saat pengguna mengetik), tombol **Return** menutup keyboard di layar dan melanjutkan fokus ke tampilan berikutnya.

Jika Anda ingin pengguna memasukkan sesuatu selain teks, seperti nomor telepon, Anda mungkin ingin mengubah tombol "aksi" menjadi ikon untuk tombol **Send**, dan mengubah tindakan menjadi menekan nomor telepon. Ikuti langkah-langkah ini:

1. Gunakan atribut `android:inputType` untuk menyetel tipe masukan keyboard:

   ```
   <EditText
      android:id="@+id/phone_number"
      android:inputType="phone"
      ... >
   </EditText>
   ```

   Metode `android:inputType` dalam contoh di atas, menyetel tipe keyboard ke `phone` , yang memaksa satu baris masukan (untuk nomor telepon).

2. Gunakan `setOnEditorActionListener()` untuk menyetel listener tampilan EditText dalam merespons penggunaan tombol "aksi":

   ```
   EditText editText = (EditText) findViewById(R.id.phone_number);
   editText.setOnEditorActionListener(new
                              TextView.OnEditorActionListener() {
      // Add onEditorAction() method
   }
   ```

3. Gunakan konstanta `IME_ACTION_SEND` dalam kelas EditorInfo bagi `actionId` untuk menampilkan tombol **Send** sebagai tombol "aksi", dan membuat metode untuk merespons tombol **Send** yang ditekan (dalam hal ini, `dialNumber` untuk memanggil nomor telepon):

```
@Override
public boolean onEditorAction(TextView textView,
                              int actionId, KeyEvent keyEvent) {
   boolean handled = false;
   if (actionId == EditorInfo.IME_ACTION_SEND) {
      dialNumber();
      handled = true;
   }
   return handled;
});
```

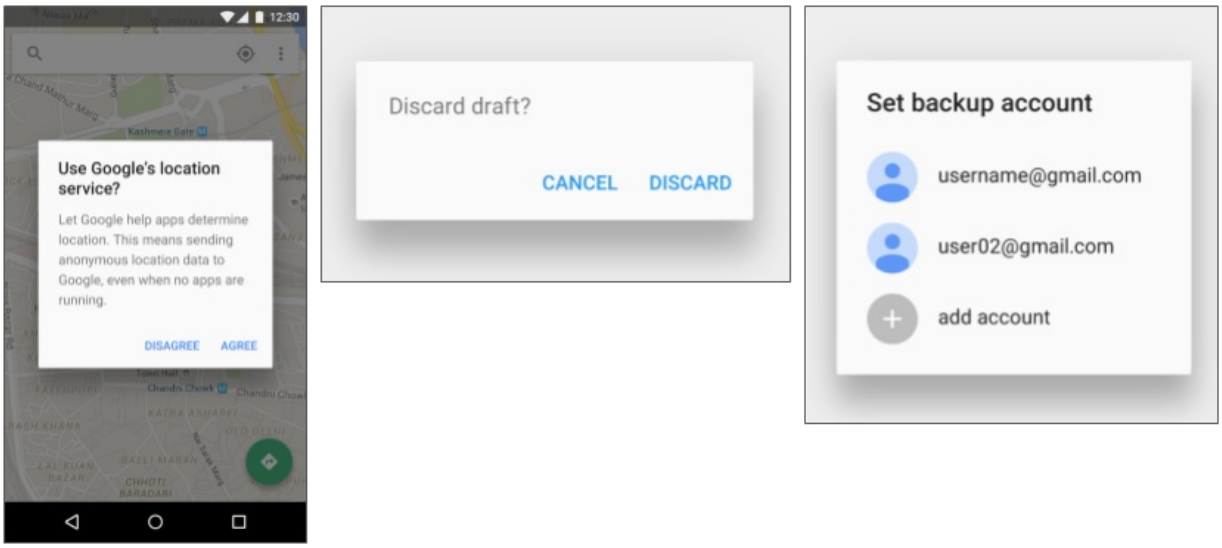
Catatan: Untuk membantu menyetel listener, lihat "Menetapkan Akses Masukan yang Menetapkan Tindakan Keyboard" dalam Bidang Teks. Untuk informasi selengkapnya tentang kelas EditorInfo, lihat Dokumentasi EditorInfo.

Menggunakan dialog dan picker

Dialog adalah jendela yang muncul di atas tampilan atau mengisi tampilan, menyela alur aktivitas. Dialog menginformasikan pengguna tentang tugas tertentu dan mungkin mengandung informasi penting, memerlukan keputusan, atau melibatkan beberapa tugas.

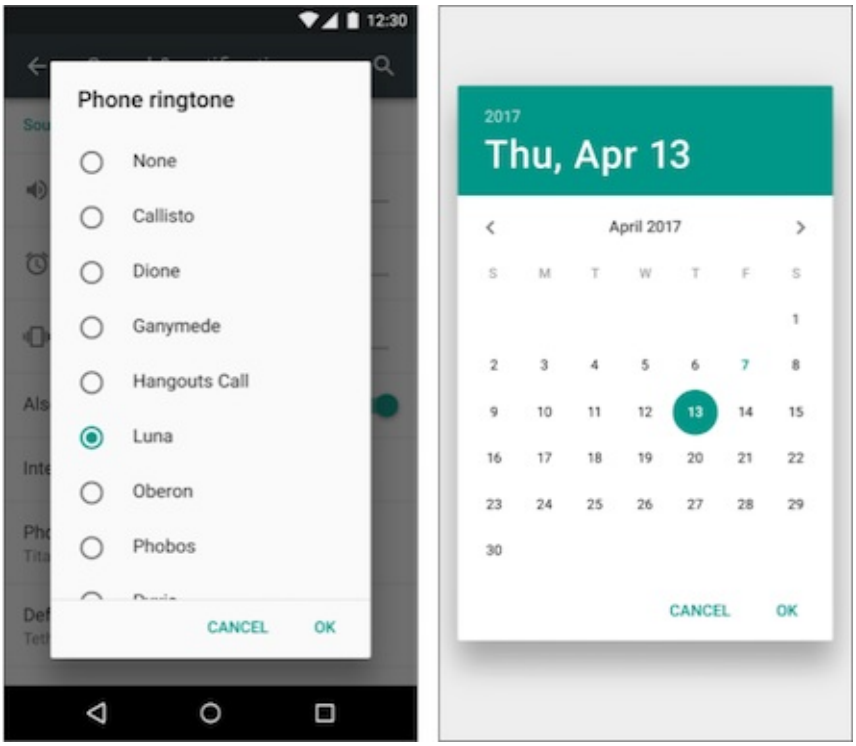
Misalnya, Anda ingin secara khusus menggunakan dialog untuk menampilkan peringatan yang mengharuskan pengguna mengetuk sebuah tombol untuk membuat keputusan, seperti **OK** atau **Cancel**. Dalam gambar di bawah ini, sisi kiri menampilkan peringatan dengan tombol **Disagree** dan **Agree**, sedangkan bagian tengah menampilkan peringatan dengan tombol **Cancel** dan **Discards**.

Anda juga bisa menggunakan dialog untuk menyediakan pilihan dalam gaya tombol radio, seperti yang ditampilkan pada sisi kanan gambar di bawah ini.



Kelas dasar untuk semua komponen dialog adalah Dialog. Ada sejumlah subkelas Dialog yang berguna untuk memperingatkan pengguna atas suatu kondisi, menampilkan status atau perkembangan, menampilkan informasi pada perangkat sekunder, atau memilih atau mengonfirmasi pilihan, seperti yang ditampilkan pada sisi kiri gambar di bawah ini. Android SDK juga menyediakan subkelas dialog yang siap digunakan seperti "picker" untuk memilih waktu atau tanggal,

seperti yang ditampilkan pada sisi kanan gambar di bawah ini. Picker memungkinkan pengguna untuk memasukkan informasi dalam format yang sudah ditentukan dan konsisten, yang mengurangi kemungkinan kesalahan masukan.



Dialog selalu menjaga fokus sampai ditutup atau aksi yang diperlukan telah dilakukan.

Tip: Praktik terbaik menyarankan untuk menggunakan sedikit dialog karena mengganggu alur kerja pengguna. Baca Panduan desain dialog untuk praktik desain terbaik tambahan, dan Dialog dalam dokumentasi developer Android untuk contoh kode.

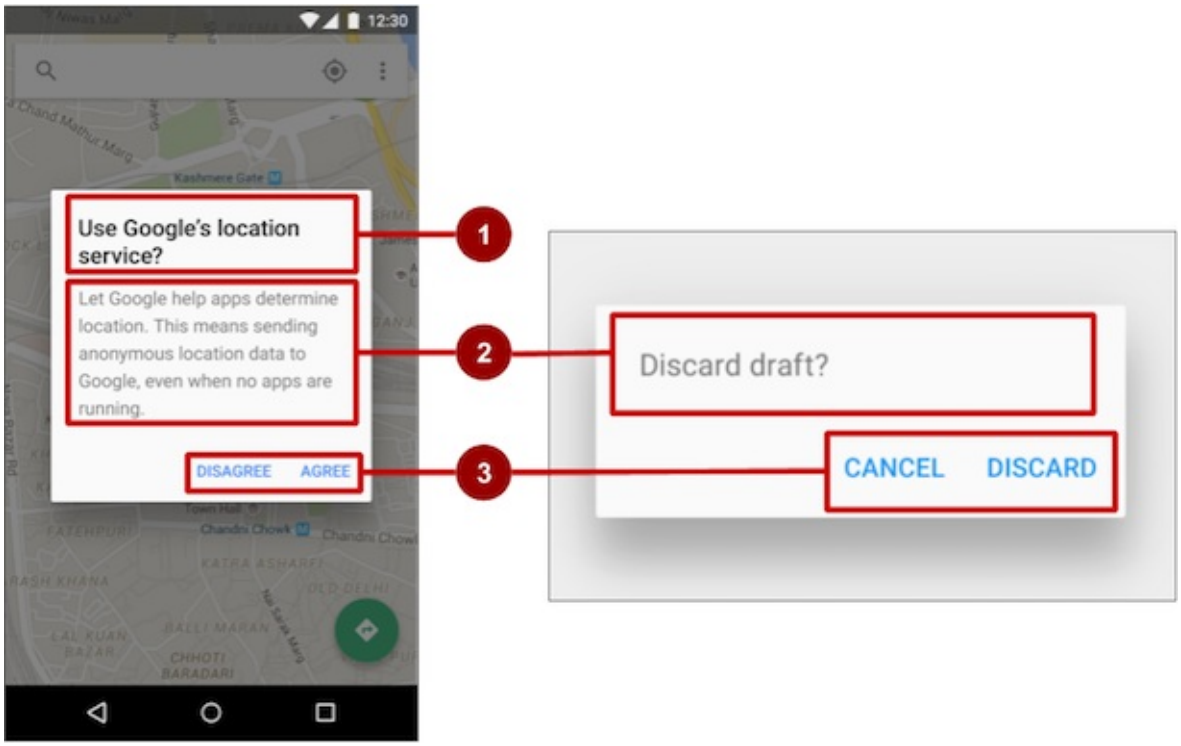
Kelas Dialog adalah kelas dasar untuk dialog, namun Anda harus menghindari membuat instance Dialog secara langsung, kecuali jika Anda sedang membuat dialog khusus. Untuk dialog Android standar, gunakan salah satu subkelas berikut:

- AlertDialog: Dialog yang bisa menampilkan judul, hingga tiga tombol, daftar item yang dapat dipilih, atau layout khusus.
- DatePickerDialog atau TimePickerDialog: Dialog berisi UI yang sudah didefinisikan, yang memungkinkan pengguna memilih tanggal atau waktu.

Menampilkan dialog peringatan

Peringatan adalah interupsi penting, memerlukan balasan atau tindakan, yang memberi tahu pengguna tentang situasi saat peringatan itu terjadi, atau aksi *sebelum* peringatan itu muncul (seperti saat membuang draf). Anda bisa menyediakan tombol dalam peringatan untuk membuat keputusan. Misalnya, sebuah dialog peringatan mungkin mengharuskan pengguna untuk mengeklik **Continue** setelah membacanya, atau memberi pilihan kepada pengguna untuk menyetujui suatu aksi dengan mengeklik tombol positif (seperti **OK** atau **Accept**), atau untuk tidak menyetujui dengan mengeklik tombol negatif (seperti **Disagree** atau **Cancel**).

Gunakan subkelas AlertDialog dari kelas Dialog untuk menampilkan dialog standar suatu peringatan. Kelas AlertDialog memungkinkan Anda untuk membangun berbagai desain dialog. Sebuah dialog peringatan bisa memiliki region berikut (lihat diagram di bawah ini):



1. Judul: Judul bersifat opsional. Sebagian besar peringatan tidak memerlukan judul. Jika Anda bisa meringkas keputusan dalam satu atau dua kalimat dengan mengajukan pertanyaan (seperti, "Buang draf?") Atau membuat pernyataan yang berkaitan dengan tombol aksi (seperti, "Klik OK untuk melanjutkan"), tidak perlu repot-repot memikirkan judul. Gunakan judul jika situasinya berisiko-tinggi, seperti potensi hilangnya konektivitas atau data, dan area materi ditempati oleh pesan terperinci, daftar, atau layout khusus.
2. Area materi: Area materi bisa menampilkan pesan, daftar, atau layout khusus lainnya.
3. Tombol aksi: Anda sebaiknya tidak menggunakan lebih dari tiga tombol aksi dalam dialog, dan kebanyakan hanya memiliki dua tombol.

Membangun AlertDialog

Kelas AlertDialog.Builder menggunakan pola desain *builder*, yang mempermudah dalam membangun objek dari kelas yang membutuhkan banyak atribut serta atribut opsional dan karena itu memerlukan banyak parameter untuk dibangun. Tanpa pola ini, Anda tentunya harus membuat konstruktor untuk kombinasi atribut yang diperlukan dan atribut opsional; dengan pola ini, kode lebih mudah dibaca dan dijaga. Untuk informasi selengkapnya tentang pola desain builder, lihat Pola builder

Gunakan `AlertDialog.Builder` untuk membangun dialog peringatan standar dan menyetel atribut pada dialog. Gunakan setTitle() untuk menyetel judulnya, setMessage() untuk menyetel pesannya, dan setPositiveButton() serta setNegativeButton() untuk menyetel tombolnya.

Catatan: Jika `AlertDialog.Builder` tidak dikenali saat memasukkannya, Anda mungkin perlu menambahkan pernyataan import berikut ke MainActivity.java:

```
import android.content.DialogInterface;
import android.support.v7.app.AlertDialog;
```

Yang berikut ini akan membuat objek dialog (`myAlertBuilder`) dan menyetel judul (sumber daya string yang disebut `alert_title`) dan pesan (sumber daya string yang disebut `alert_message`):

```
AlertDialog.Builder myAlertBuilder = new
                AlertDialog.Builder(MainActivity.this);
myAlertBuilder.setTitle(R.string.alert_title);
myAlertBuilder.setMessage(R.string.alert_message);
```

Menyetel tombol tindakan untuk dialog peringatan

Gunakan metode setPositiveButton() dan setNegativeButton() dari kelas AlertDialog.Builder untuk menyetel tombol tindakan bagi dialog peringatan. Metode ini memerlukan judul tombol (disediakan oleh sumber daya string) dan kelas DialogInterface.OnClickListener yang mendefinisikan aksi yang diambil bila pengguna menekan tombol:

```
myAlertBuilder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int which) {
           // User clicked OK button.
      }
});
myAlertBuilder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int which) {
         // User clicked the CANCEL button.
      }
   });
```

Anda bisa menambahkan salah satu saja dari setiap tipe tombol ke AlertDialog. Misalnya, Anda tidak bisa memiliki lebih dari satu tombol "positif".

Tip: Anda juga bisa menyetel tombol "netral" dengan setNeutralButton(). Tombol netral muncul di antara tombol positif dan tombol negatif. Gunakan tombol netral, seperti "Remind me later" jika Anda ingin pengguna dapat menghilangkan dialog dan memutuskan nanti.

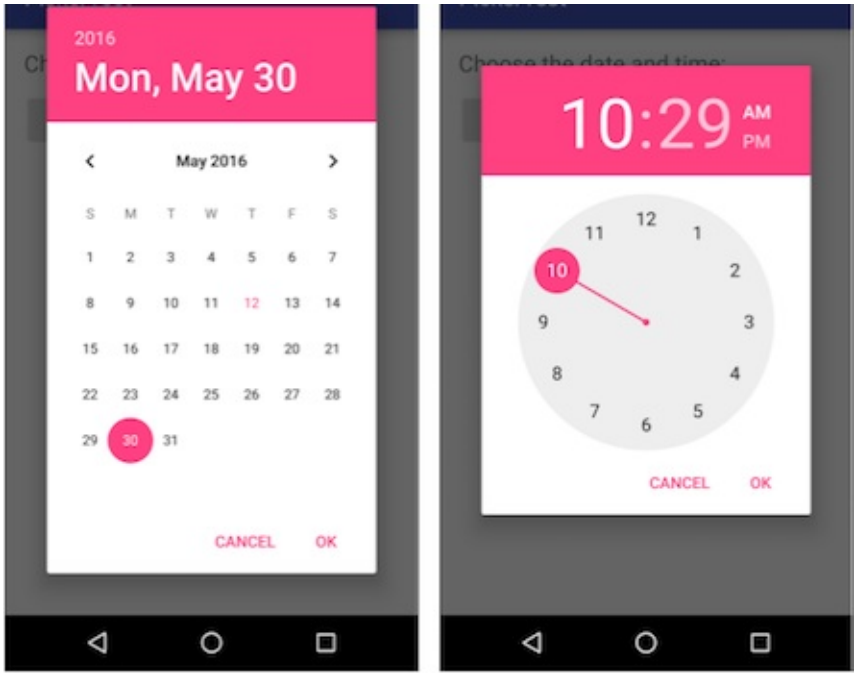
Menampilkan dialog

Untuk menampilkan dialog, panggil metode `show()` :

```
alertDialog.show();
```

Picker tanggal dan waktu

Android menyediakan dialog yang siap digunakan, disebut *picker*, untuk memilih waktu atau tanggal. Gunakan picker untuk memastikan pengguna Anda memilih waktu atau tanggal yang valid, yang diformat dengan benar dan disesuaikan dengan lokal pengguna. Setiap picker menyediakan kontrol untuk memilih setiap bagian waktu (jam, menit, AM/PM) atau tanggal (bulan, tanggal, tahun).



Saat menampilkan picker, Anda harus menggunakan instance DialogFragment, sebuah subkelas Fragment, yang menampilkan jendela dialog yang mengambang di atas jendela aktivitasnya. *Fragmen* adalah perilaku atau bagian antarmuka pengguna dalam suatu aktivitas. Fragmen seperti aktivitas mini di dalam aktivitas utama, dengan daur hidup individualnya sendiri. Suatu fragmen menerima kejadian masukannya sendiri, dan Anda bisa menambahkan atau membuangnya saat aktivitas utama berjalan. Anda mungkin mengombinasikan beberapa fragmen dalam suatu aktivitas tunggal untuk membangun antarmuka pengguna multipanel, atau menggunakan kembali suatu fragmen dalam beberapa aktivitas. Untuk mengetahui tentang fragmen, lihat Fragmen dalam Panduan API.

Salah satu manfaat menggunakan fragmen untuk picker adalah Anda bisa mengisolasi bagian kode untuk mengelola tanggal dan waktu setelah pengguna memilihnya dari picker. Anda juga bisa menggunakan DialogFragment untuk mengelola daur hidup dialog.

Tip: Manfaat lain menggunakan fragmen untuk picker adalah Anda bisa mengimplementasikan konfigurasi layout yang berbeda, seperti dialog dasar pada tampilan berukuran handset atau bagian layout yang disematkan pada tampilan yang besar.

Menambahkan fragmen

Untuk menambahkan fragmen bagi picker tanggal, buat fragmen kosong (**DatePickerFragment**) tanpa XML layout, dan tanpa metode pabrik atau callback antarmuka:

1. Luaskan **app > java > com.example.android.DateTimePickers** dan pilih **MainActivity**.
2. Pilih **File > New > Fragment > Fragment (Blank)**, dan beri nama fragmen **DatePickerFragment**. Kosongkan ketiga opsi kotak centang sehingga Anda *tidak* membuat XML layout, *jangan* sertakan metode pabrik fragmen, dan *jangan* sertakan callback antarmuka. Anda tidak perlu membuat layout untuk picker standar. Klik **Finish** untuk membuat fragmen.

Memperluas DialogFragment untuk picker

Langkah berikutnya adalah membuat picker standar dengan listener. Ikuti langkah-langkah ini:

1. Edit definisi kelas `DatePickerFragment` untuk memperluas `DialogFragment,` dan implementasikan `DatePickerDialog.OnDateSetListener` untuk membuat picker tanggal standar dengan listener:

   ```
   public class DatePickerFragment extends DialogFragment
                       implements DatePickerDialog.OnDateSetListener {
   ...
   }
   ```

   Android Studio secara otomatis menambahkan yang berikut ini dalam blok import di bagian atas:

   ```
   import android.app.DatePickerDialog.OnDateSetListener;
   import android.support.v4.app.DialogFragment;
   ```

2. Android Studio juga menampilkan ikon bola lampu merah di batas kiri, mengarahkan Anda untuk mengimplementasikan metode. Klik ikon dan, dengan **onDateSet** sudah dipilih serta opsi "Insert @Override" sudah dicentang, klik **OK** untuk membuat metode onDateSet().

   Kemudian Android Studio secara otomatis menambahkan yang berikut ini dalam blok import di bagian atas:

   ```
   import android.widget.DatePicker;
   ```

3. Ganti `onCreateView()` dengan `onCreateDialog()` :

   ```
   @Override
   public Dialog onCreateDialog(Bundle savedInstanceState) {
   }
   ```

Saat Anda memperluas `DialogFragment`, Anda harus mengganti metode callback onCreateDialog(), daripada `onCreateView`. Gunakan metode callback versi Anda untuk melakukan inisialisasi `year`, `month`, dan `day` untuk picker tanggal.

Menyetel default dan mengembalikan picker

Untuk menyetel tanggal default dalam picker dan mengembalikannya sebagai objek yang bisa Anda gunakan, ikuti langkah-langkah ini:

1. Tambahkan kode berikut pada metode `onCreateDialog()` untuk menyetel tanggal default bagi picker:

```
// Use the current date as the default date in the picker.
final Calendar c = Calendar.getInstance();
int year = c.get(Calendar.YEAR);
int month = c.get(Calendar.MONTH);
int day = c.get(Calendar.DAY_OF_MONTH);
```

   Saat Anda memasukkan `Calendar`, Anda diberi pilihan pustaka Kalender yang akan diimpor. Pilih ini:

```
import java.util.Calendar;
```

   Kelas Calendar menyetel tanggal default sebagai tanggal saat ini—kelas ini mengonversi antara instan tertentu dan serangkaian bidang kalender seperti `YEAR`, `MONTH`, `DAY_OF_MONTH`, `HOUR`, dan seterusnya. Kalender bersifat sensitif-lokal, dan metode kelasnya `getInstance()` mengembalikan objek Calendar yang bidang kalendernya telah diinisialisasi dengan tanggal dan waktu saat ini.

2. Tambahkan pernyataan berikut pada akhir metode untuk membuat instance picker tanggal yang baru dan mengembalikannya:

```
return new DatePickerDialog(getActivity(), this, year, month, day);
```

Menampilkan picker

Pada Aktivitas Utama, Anda perlu membuat metode untuk menampilkan picker tanggal. Ikuti langkah-langkah ini:

1. Buat metode untuk membuat instance fragmen dialog picker tanggal:

```
public void showDatePickerDialog(View v) {
    DialogFragment newFragment = new DatePickerFragment();
    newFragment.show(getSupportFragmentManager(), "datePicker");
}
```

2. Kemudian Anda bisa menggunakan `showDatePickerDialog()` dengan atribut `android:onClick` untuk tombol atau kontrol masukan lainnya:

```
        android:id="@+id/button_date"
        ...
        android:onClick="showDatePickerDialog"/>
```

Memproses pilihan picker pengguna

Metode `onDateSet()` secara otomatis dipanggil saat pengguna membuat pilihan dalam picker tangal, sehingga Anda bisa menggunakan metode ini untuk melakukan sesuatu dengan tanggal yang dipilih tersebut. Ikuti langkah-langkah ini:

1. Agar kode lebih terbaca, ubah parameter metode `onDateSet()` dari `int i`, `int i1`, dan `int i2` ke `int year`, `int month`, dan `int day`:

```
public void onDateSet(DatePicker view, int year, int month, int day) {
```

2. Buka **MainActivity** dan tambahkan penanda metode `processDatePickerResult()` yang mengambil `year`, `month`, dan `day` sebagai argumen:

```
public void processDatePickerResult(int year, int month, int day) {
}
```

3. Tambahkan kode berikut pada metode `processDatePickerResult()` untuk mengonversi `month`, `day`, dan `year` dan untuk memisahkan string:

```
String month_string = Integer.toString(month+1);
String day_string = Integer.toString(day);
String year_string = Integer.toString(year);
```

   Catatan: Integer `month` yang dikembalikan oleh picker tanggal mulai menghitung dari 0 untuk Januari, sehingga Anda perlu menambahkan 1 untuk mulai menampilkan bulan mulai dari 1.

4. Tambahkan yang berikut ini setelah kode di atas untuk menggabungkan tiga string dan menyertakan garis miring untuk format tanggal A.S.:

```
String dateMessage = (month_string + "/" +
                          day_string + "/" + year_string);
```

5. Tambahkan yang berikut ini setelah pernyataan di atas untuk menampilkan pesan Toast:

```
Toast.makeText(this, "Date: " + dateMessage,
                          Toast.LENGTH_SHORT).show();
```

6. Ekstrak string hard-code `"Date: "` menjadi sumber daya string bernama `date`. Hal ini secara otomatis mengganti string hard-code dengan `getString(R.string.date)`. Kode untuk metode `processDatePickerResult()` sekarang harus tampak seperti ini:

```
public void processDatePickerResult(int year, int month, int day) {
   String month_string = Integer.toString(month + 1);
   String day_string = Integer.toString(day);
   String year_string = Integer.toString(year);
   // Assign the concatenated strings to dateMessage.
   String dateMessage = (month_string + "/" +
                                day_string + "/" + year_string);
   Toast.makeText(this, getString(R.string.date) + dateMessage,
                                Toast.LENGTH_SHORT).show();
}
```

7. Buka **DatePickerFragment**, dan tambahkan yang berikut ini ke metode `onDateSet()` untuk memanggil metode `processDatePickerResult()` di `MainActivity` dan meneruskan `year`, `month`, dan `day` ke sana:

```
public void onDateSet(DatePicker view, int year, int month, int day) {
   // Set the activity to the Main Activity.
   MainActivity activity = (MainActivity) getActivity();
   // Invoke Main Activity's processDatePickerResult() method.
   activity.processDatePickerResult(year, month, day);
}
```

   Gunakan `getActivity()` yang, jika digunakan dalam suatu fragmen, akan mengembalikan aktivitas yang saat ini dikaitkan dengan fragmen. Anda memerlukannya karena Anda tidak bisa memanggil suatu metode dalam `MainActivity` tanpa konteks `MainActivity` (Anda seharusnya menggunakan `intent`, seperti yang Anda pelajari pada pembelajaran sebelumnya). Aktivitas mewarisi konteks, sehingga Anda bisa menggunakannya sebagai konteks untuk memanggil metode (seperti dalam `activity.processDatePickerResult`).

Menggunakan prosedur yang sama untuk picker waktu

Ikuti prosedur yang sama yang diuraikan di atas untuk picker tanggal:

1. Tambahkan fragmen kosong yang disebut **TimePickerFragment** yang memperluas `DialogFragment` dan mengimplementasikan `TimePickerDialog.OnTimeSetListener`:

```
public class TimePickerFragment extends DialogFragment
                implements TimePickerDialog.OnTimeSetListener {
```

2. Tambahkan dengan `@Override` metode `onTimeSet()` kosong:

   Android Studio juga menampilkan ikon bola lampu merah di batas kiri, yang meminta konfirmasi Anda untuk mengimplementasikan metode. Klik ikon dan, dengan **onTimeSet** sudah dipilih serta opsi "Insert @Override" sudah dicentang, klik **OK** untuk membuat metode onTimeSet(). Kemudian Android Studio secara otomatis menambahkan yang berikut ini dalam blok import di bagian atas:

```
import android.widget.TimePicker;
```

3. Gunakan `onCreateDialog()` untuk melakukan inisialisasi waktu dan mengembalikan dialog:

```
public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Use the current time as the default values for the picker.
        final Calendar c = Calendar.getInstance();
        int hour = c.get(Calendar.HOUR_OF_DAY);
        int minute = c.get(Calendar.MINUTE);

        // Create a new instance of TimePickerDialog and return it.
        return new TimePickerDialog(getActivity(), this, hour, minute,
                DateFormat.is24HourFormat(getActivity()));
    }
```

4. Tampilkan picker: Buka **MainActivity** dan buat metode untuk membuat instance fragmen dialog picker tanggal:

```
public void showDatePickerDialog(View v) {
        DialogFragment newFragment = new DatePickerFragment();
        newFragment.show(getSupportFragmentManager(), "datePicker");
}
```

   Gunakan `showDatePickerDialog()` dengan atribut `android:onClick` untuk tombol atau kontrol masukan lainnya:

```
        android:id="@+id/button_date"
        ...
        android:onClick="showDatePickerDialog"/>
```

5. Buat metode `processTimePickerResult()` di **MainActivity** untuk memproses hasil pemilihan dari picker waktu:

```
public void processTimePickerResult(int hourOfDay, int minute) {
   // Convert time elements into strings.
   String hour_string = Integer.toString(hourOfDay);
   String minute_string = Integer.toString(minute);
   // Assign the concatenated strings to timeMessage.
   String timeMessage = (hour_string + ":" + minute_string);
   Toast.makeText(this, getString(R.string.time) + timeMessage,
                                    Toast.LENGTH_SHORT).show();
}
```

6. Gunakan `onTimeSet()` untuk mendapatkan waktu dan meneruskannya ke metode `processTimePickerResult()` di MainActivity:

```
public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
   // Set the activity to the Main Activity.
   MainActivity activity = (MainActivity) getActivity();
   // Invoke Main Activity's processDatePickerResult() method.
   activity.processTimePickerResult(hourOfDay, minute);
}
```

Anda bisa membaca semua tentang mempersiapkan picker di Picker.

Mengenali isyarat

Isyarat sentuh terjadi saat seorang pengguna meletakkan satu atau beberapa jari di atas layar sentuh, dan aplikasi Anda akan menerjemahkan pola sentuhan sebagai isyarat khusus, seperti sentuhan lama, ketuk dua kali, gesek cepat, atau guliran.

Android menyediakan berbagai kelas dan metode untuk membantu Anda membuat dan mendeteksi isyarat. Meskipun aplikasi Anda seharusnya tidak bergantung pada isyarat sentuhan untuk perilaku dasar (karena isyarat mungkin tidak tersedia untuk semua pengguna dalam semua konteks), menambahkan interaksi berbasis sentuhan ke aplikasi Anda bisa sangat meningkatkan daya guna dan daya tariknya.

Untuk menyediakan pengalaman yang intuitif dan konsisten bagi pengguna, aplikasi Anda seharusnya mengikuti konvensi Android yang diterima untuk isyarat sentuh. Panduan desain isyarat menampilkan cara mendesain isyarat umum dalam aplikasi Android. Untuk detail dan contoh kode selengkapnya, lihat Menggunakan Isyarat Sentuh dalam dokumentasi developer Android.

Mendeteksi isyarat umum

Jika aplikasi Anda menggunakan isyarat umum seperti ketuk dua kali, tekan lama, lempar, dan seterusnya, Anda bisa memanfaatkan kelas GestureDetector untuk mendeteksi isyarat umum. Gunakan GestureDetectorCompat, yang disediakan sebagai implementasi kompatibilitas kelas GestureDetector kerangka kerja, yang menjamin perilaku menggulir titik fokus terbaru dari Jellybean MR1 pada semua versi platform. Kelas ini seharusnya hanya digunakan dengan kejadian gerakan yang dilaporkan untuk perangkat sentuh—jangan gunakan untuk trackball atau kejadian perangkat keras lainnya.

GestureDetectorCompat memungkinkan Anda mendeteksi isyarat umum tanpa harus memproses kejadian sentuh individual sendiri. GestureDetectorCompat mendeteksi beragam isyarat dan kejadian dengan menggunakan objek MotionEvent, yang melaporkan gerakan oleh jari (atau mouse, pena, atau trackball).

Cuplikan berikut menampilkan cara Anda menggunakan kelas `GestureDetectorCompat` dan `GestureDetector.SimpleOnGestureListener` .

1. Untuk menggunakan GestureDetectorCompat, buat instance (`mDetector` dalam cuplikan di bawah) dari kelas `GestureDetectorCompat` , dengan menggunakan metode `onCreate()` dalam aktivitas (seperti **MainActivity**):

```
public class MainActivity extends Activity {
    private GestureDetectorCompat mDetector;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mDetector = new GestureDetectorCompat(this, new
            MyGestureListener());
    }
    ...
}
```

Saat Anda membuat instance objek `GestureDetectorCompat` , salah satu parameter yang diambil adalah kelas yang harus Anda buat— `MyGestureListener` dalam cuplikan di atas—yang melakukan salah satu hal berikut:

- mengimplementasikan antarmuka GestureDetector.OnGestureListener, untuk mendeteksi semua isyarat standar, atau
- memperluas kelas GestureDetector.SimpleOnGestureListener, yang bisa Anda gunakan untuk memproses sedikit isyarat saja dengan mengganti metode yang Anda perlukan. Beberapa metode yang disediakannya antara lain onDown(), onLongPress(), onFling(), onScroll(), dan onSingleTapUp().

1. Buat kelas `MyGestureListener` sebagai aktivitas tersendiri (**MyGestureListener** `)` untuk memperluas `GestureDetector.SimpleOnGestureListener`, dan mengganti metode onFling() serta onDown() untuk menampilkan pernyataan log tentang kejadian:

```
class MyGestureListener
                   extends GestureDetector.SimpleOnGestureListener {
   private static final String DEBUG_TAG = "Gestures";

   @Override
   public boolean onDown(MotionEvent event) {
      Log.d(DEBUG_TAG,"onDown: " + event.toString());
      return true;
   }

   @Override
   public boolean onFling(MotionEvent event1, MotionEvent event2,
                  float velocityX, float velocityY) {
       Log.d(DEBUG_TAG, "onFling: " +
            event1.toString()+event2.toString());
       return true;
   }
}
```

2. Untuk mencegat kejadian sentuh, gantikan callback onTouchEvent() dari kelas GestureDetectorCompat di **MainActivity**:

```
@Override
public boolean onTouchEvent(MotionEvent event){
  this.mDetector.onTouchEvent(event);
  return super.onTouchEvent(event);
}
```

Mendeteksi semua isyarat

Untuk mendeteksi semua tipe isyarat, Anda perlu melakukan dua langkah penting:

1. Kumpulkan data tentang kejadian sentuh.
2. Menerjemahkan data untuk melihat apakah memenuhi kriteria salah satu isyarat yang didukung oleh aplikasi Anda.

Isyarat tersebut dimulai ketika pengguna pertama kali menyentuh layar, terus berlanjut ketika sistem melacak posisi jari pengguna, dan berakhir dengan menangkap kejadian terakhir saat jari pengguna meninggalkan layar. Selama interaksi ini, objek kelas MotionEvent dikirimkan ke onTouchEvent(), menyediakan detail setiap interaksi. Aplikasi Anda bisa menggunakan data yang disediakan oleh MotionEvent untuk menentukan apakah terjadi isyarat yang terkait dengannya.

Misalnya, saat pengguna pertama kali menyentuh layar, metode onTouchEvent() dipicu pada tampilan yang disentuh, dan objek MotionEvent() melaporkan gerakan oleh jari (atau mouse, atau pena, atau trackball) dalam hal:

- *Kode aksi:* Menetapkan perubahan keadaan yang terjadi, seperti jari mengetuk atau diangkat.
- *Serangkaian nilai sumbu:* Menjelaskan posisi pada koordinat sentuhan X dan Y serta informasi tentang tekanan, ukuran dan orientasi area kontak.

Jari individu atau objek lain yang menghasilkan jejak gerakan disebut dengan *pointer*. Beberapa perangkat bisa melaporkan beberapa jejak gerakan pada saat yang sama. Layar multi-sentuhan menampilkan satu jejak gerakan untuk setiap jari. Kejadian gerakan berisi informasi tentang semua pointer yang saat ini aktif, bahkan jika beberapa di antaranya tidak bergerak sejak kejadian pertama dikirimkan. Berdasarkan interpretasi objek MotionEvent, metode onTouchEvent() memicu callback yang sesuai pada antarmuka GestureDetector.OnGestureListener.

Setiap pointer MotionEvent memiliki ID unik yang diberikan saat pointer pertama kali turun (ditunjukkan oleh `ACTION_DOWN` atau `ACTION_POINTER_DOWN)` . ID pointer akan tetap valid hingga akhirnya pointer naik (ditunjukkan oleh `ACTION_UP` atau `ACTION_POINTER_UP`) atau bila isyarat dibatalkan (ditunjukkan oleh `ACTION_CANCEL`). Kelas MotionEvent menyediakan metode untuk menanyakan posisi dan properti pointer lainnya, seperti getX(int), getY(int), getAxisValue(int), getPointerId(int), dan getToolType(int).

Interpretasi materi MotionEvent sangat beragam bergantung pada kelas sumber perangkat. Pada layar sentuh, koordinat pointer menetapkan posisi absolut seperti koordinat X/Y tampilan. Setiap isyarat lengkap dinyatakan oleh urutan kejadian gerakan dengan tindakan yang menjelaskan transisi keadaan dan gerakan pointer.

Isyarat memulai kejadian gerakan dengan `ACTION_DOWN` yang menyediakan lokasi turunnya pointer untuk pertama kali. Karena setiap pointer tambahan turun atau naik, kerangka kerja menghasilkan kejadian gerakan dengan `ACTION_POINTER_DOWN` atau `ACTION_POINTER_UP` yang sesuai. Gerakan pointer dijelaskan oleh kejadian gerakan dengan `ACTION_MOVE` . Isyarat berakhir saat pointer terakhir naik sebagaimana dinyatakan oleh kejadian gerakan dengan `ACTION_UP` , atau bila isyarat dibatalkan dengan `ACTION_CANCEL` .

Untuk mencegat kejadian sentuh dalam suatu aktivitas atau tampilan, ganti callback `onTouchEvent()` seperti yang ditampilkan dalam cuplikan di bawah ini. Anda bisa menggunakan metode getActionMasked() dari kelas MotionEventCompat untuk mengekstrak aksi yang dilakukan pengguna dari parameter kejadian. (MotionEventCompat adalah penunjang untuk mengakses fitur dalam MotionEvent, yang diperkenalkan setelah API level 4 yang kompatibel dengan versi sebelumnya.) Hal ini akan memberi data mentah yang Anda butuhkan jika isyarat yang terkait dengan Anda terjadi:

```
public class MainActivity extends Activity {
...
// This example shows an Activity, but you would use the same approach if
// you were subclassing a View.
   @Override
   public boolean onTouchEvent(MotionEvent event){

       int action = MotionEventCompat.getActionMasked(event);

       switch(action) {
          case (MotionEvent.ACTION_DOWN) :
             Log.d(DEBUG_TAG,"Action was DOWN");
             return true;
          case (MotionEvent.ACTION_MOVE) :
             Log.d(DEBUG_TAG,"Action was MOVE");
             return true;
          case (MotionEvent.ACTION_UP) :
             Log.d(DEBUG_TAG,"Action was UP");
             return true;
          case (MotionEvent.ACTION_CANCEL) :
             Log.d(DEBUG_TAG,"Action was CANCEL");
             return true;
          case (MotionEvent.ACTION_OUTSIDE) :
             Log.d(DEBUG_TAG,"Movement occurred outside bounds " +
                    "of current screen element");
             return true;
          default :
             return super.onTouchEvent(event);
    }
}
```

Selanjutnya Anda bisa melakukan pemrosesan sendiri terhadap kejadian ini untuk menentukan apakah terjadi isyarat.

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Menggunakan Keyboard, Kontrol Masukan, Peringatan, dan Picker

Ketahui selengkapnya

- Panduan Android API, bagian "Kembangkan":
  - Menetapkan Tipe Metode Masukan
  - Menangani Masukan Keyboard
  - Bidang Teks
  - EditorInfo
  - Kontrol Masukan
  - Tombol
  - Gaya dan Tema
  - Spinner
  - Dialog
  - Fragmen
  - Kejadian Masukan
  - Picker
  - DateFormat
  - Menggunakan Isyarat Sentuh
- Spesifikasi Desain Material:
  - Komponen - Tombol
  - Panduan desain dialog
  - Panduan desain isyarat
- Blog Developer:
  - "Holo Everywhere"
  - Mengimplementasikan Desain Material pada Aplikasi Android Anda

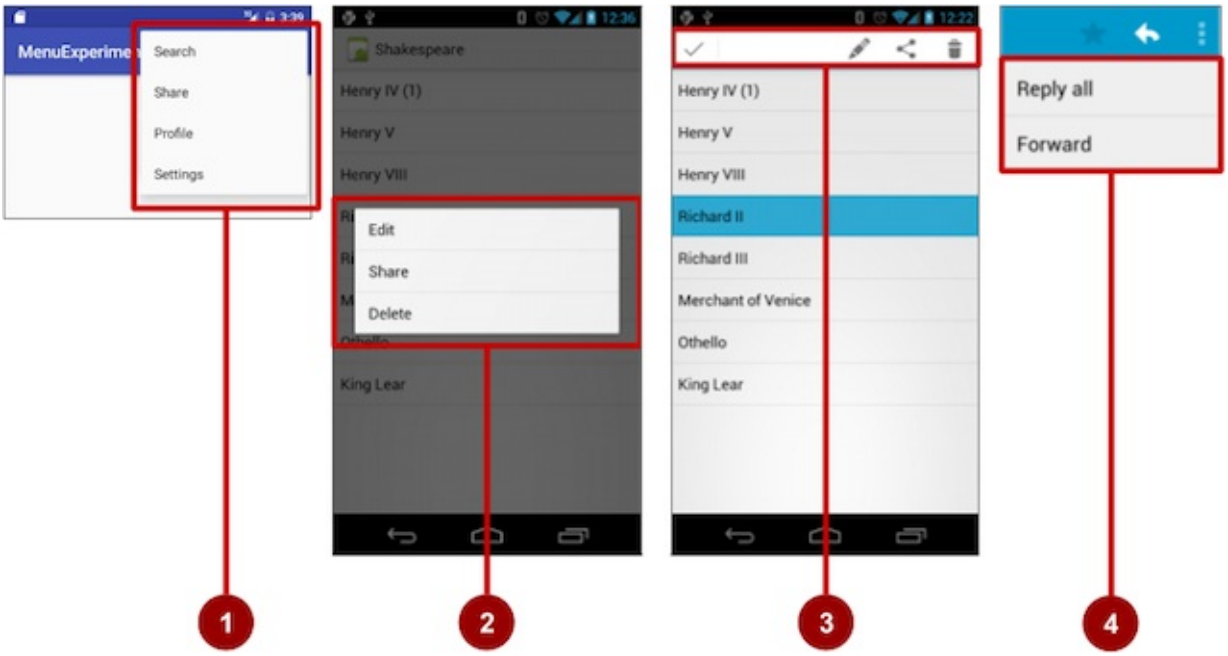
4.2: Menu

Materi:

- Tipe menu
- Bilah aplikasi dan menu opsi
- Menu kontekstual
- Menu munculan
- Praktik terkait
- Ketahui selengkapnya

Tipe menu

Menu adalah serangkaian opsi yang bisa dipilih pengguna untuk menjalankan suatu fungsi, seperti menelusuri informasi, menyimpan informasi, mengedit informasi, atau mengarahkan ke suatu layar. Android menawarkan tipe menu berikut, yang berguna untuk berbagai situasi (lihat gambar di bawah ini):



1. *Menu opsi:* Muncul dalam bilah aplikasi dan menyediakan opsi utama yang memengaruhi penggunaan aplikasi itu sendiri. Contoh opsi menu: **Search** untuk melakukan penelusuran, **Bookmark** untuk menyimpan tautan ke suatu layar, dan **Setting** untuk mengarahkan ke layar Settings.
2. *Menu konteks:* Muncul sebagai daftar pilihan yang mengambang bila pengguna melakukan ketukan lama pada suatu elemen di layar. Contoh opsi menu: **Edit** untuk mengedit elemen, **Delete** untuk menghapusnya, dan **Share** untuk membagikannya melalui media sosial.
3. *Bilah aksi kontekstual:* Muncul di bagian atas layar yang menghamparkan bilah aplikasi, dengan item aksi yang memengaruhi elemen yang dipilih. Contoh opsi menu: **Edit**, **Share**, dan **Delete** untuk satu atau beberapa elemen yang dipilih.
4. *Menu munculan:* Muncul dengan ditambatkan pada suatu tampilan seperti ImageButton, dan menyediakan luapan aksi atau bagian ke dua dari perintah dua bagian. Contoh menu munculan: aplikasi Gmail menambatkan menu munculan pada bilah aplikasi untuk tampilan pesan dengan **Reply**, **Reply All**, dan **Forward**.

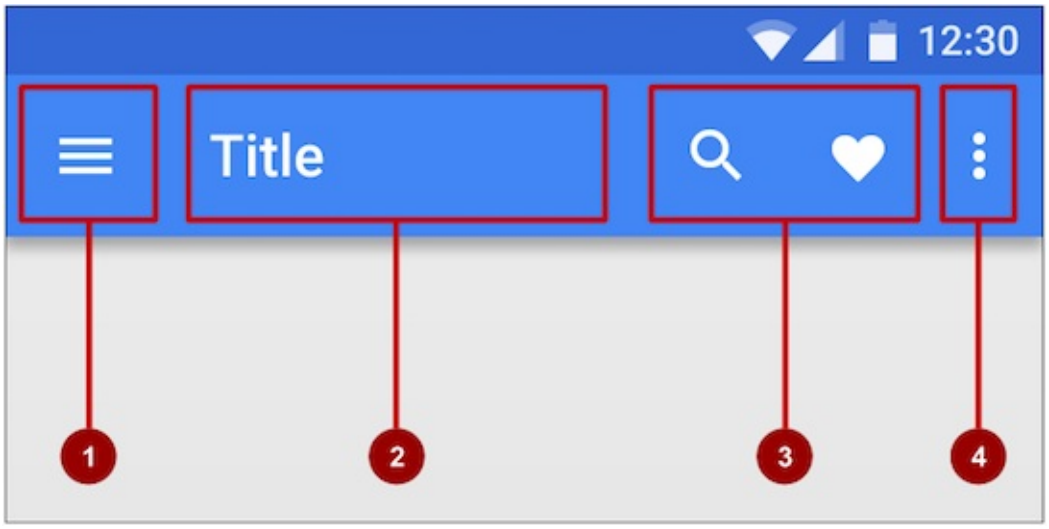
Bilah aplikasi dan menu opsi

Bilah aplikasi (disebut juga *bilah aksi*) adalah ruang khusus di bagian atas setiap layar aktivitas. Bila Anda membuat aktivitas dari suatu template (seperti Empty Template), bilah aplikasi secara otomatis disertakan untuk aktivitas dalam grup tampilan akar CoordinatorLayout di bagian atas hierarki tampilan.

Bilah aplikasi secara default menampilkan judul aplikasi, atau nama yang didefinisikan dalam AndroidManifest.xml dengan atribut `android:label` untuk aktivitas. Bilah aplikasi juga menyertakan tombol **Naik** untuk mengarahkan ke atas ke aktivitas induk, yang dijelaskan di bab berikutnya.

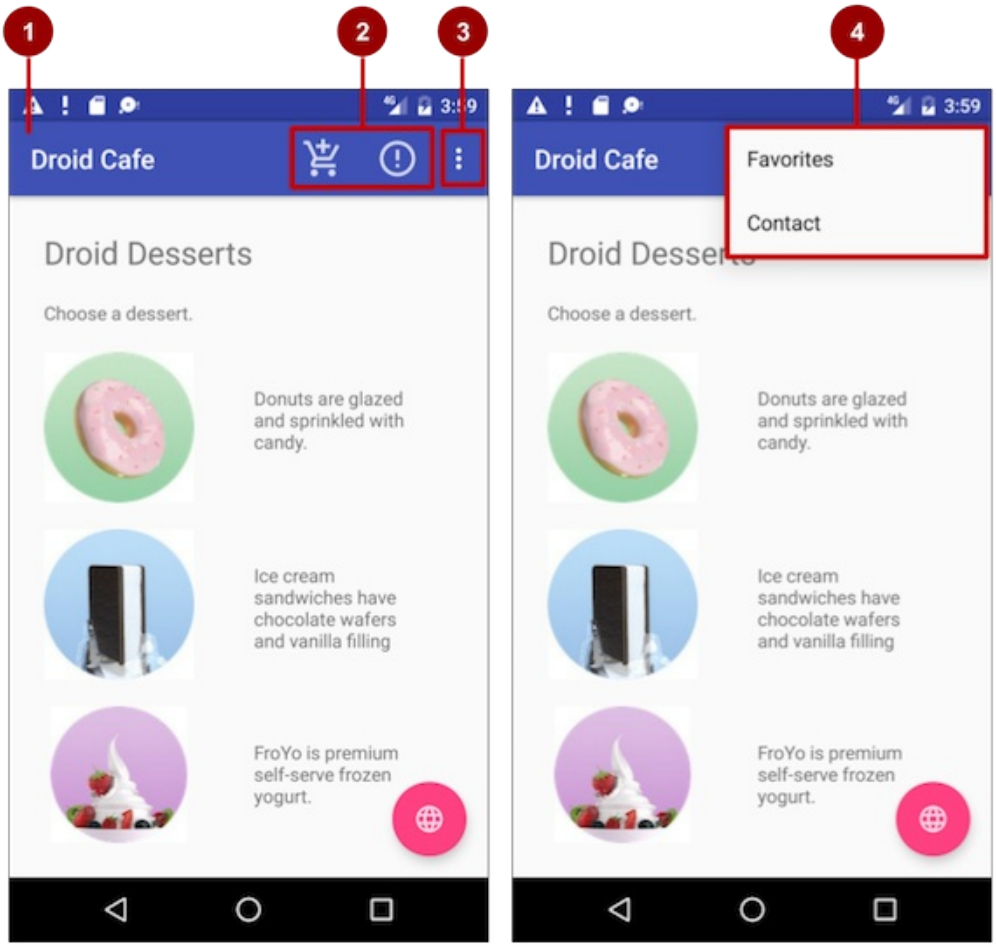
Menu opsi dalam bilah aplikasi menyediakan navigasi ke aktivitas lain dalam aplikasi, atau opsi utama yang memengaruhi penggunaan aplikasi itu sendiri— namun bukan yang menjalankan aksi pada elemen di layar. Misalnya, menu opsi Anda mungkin menyediakan pilihan pengguna untuk mengarahkan ke aktivitas lain, seperti menempatkan urutan, atau untuk aksi yang memiliki efek global pada aplikasi, seperti mengubah setelan atau informasi profil.

Menu opsi muncul di sudut kanan bilah aplikasi. Bilah aplikasi terbagi menjadi empat area fungsional berbeda yang berlaku untuk sebagian besar aplikasi:



1. *Tombol navigasi atau Tombol Naik:* Gunakan tombol navigasi dalam ruang ini untuk membuka panel samping navigasi, atau menggunakan tombol Naik untuk mengarahkan ke atas melalui hierarki layar aplikasi Anda ke aktivitas induk. Keduanya akan dijelaskan di bab berikutnya.
2. *Judul:* Judul dalam bilah aplikasi adalah judul aplikasi, atau nama yang didefinisikan dalam AndroidManifest.xml oleh atribut `android:label` untuk aktivitas.
3. *Ikon aksi untuk menu opsi:* Setiap ikon aksi muncul dalam bilah aplikasi dan menyatakan salah satu item pada menu opsi yang paling sering digunakan. Item menu opsi yang lebih jarang digunakan akan muncul di menu opsi luapan.
4. *Menu opsi luapan:* Ikon luapan membuka munculan bersama item menu opsi yang tidak ditampilkan seperti ikon dalam bilah aplikasi.

Item menu opsi yang sering digunakan akan muncul sebagai ikon dalam bilah aplikasi. Menu opsi luapan menampilkan menu selebihnya:



Pada gambar di atas:

1. **Bilah aplikasi**. Bilah aplikasi menyertakan judul aplikasi, menu opsi, dan tombol luapan.
2. **Ikon aksi menu opsi**. Dua item menu opsi pertama muncul sebagai ikon dalam bilah aplikasi.
3. **Tombol luapan**. Tombol luapan (tiga titik vertikal) membuka menu yang menampilkan item menu opsi selengkapnya.
4. **Menu luapan opsi**. Setelah mengeklik tombol luapan, item menu opsi selengkapnya akan muncul dalam menu luapan.

Menambahkan bilah aplikasi

Setiap aktivitas yang menggunakan tema default juga memiliki ActionBar sebagai bilah aplikasinya. Beberapa tema juga mempersiapkan ActionBar sebagai bilah aplikasi secara default. Jika Anda memulai suatu aplikasi dari template seperti Empty Activity, sebuah ActionBar akan muncul sebagai bilah aplikasi.

Akan tetapi, karena fitur ditambahkan ke ActionBar asli pada beragam rilis Android, ActionBar asli akan berperilaku berbeda-beda bergantung pada versi Android yang berjalan pada perangkat tersebut. Oleh karena itu, jika Anda menambahkan menu opsi, Anda harus menggunakan Toolbar pustaka dukungan v7 appcompat sebagai bilah aplikasi. Menggunakan Toolbar akan memudahkan dalam mempersiapkan bilah aplikasi yang bekerja pada beragam perangkat, serta memberi Anda ruang untuk menyesuaikan bilah aplikasi bila nanti aplikasi Anda berkembang. Bilah alat menyertakan berbagai fitur terbaru, dan bekerja pada perangkat apa pun yang bisa menggunakan pustaka dukungan.

Untuk menggunakan Bilah Alat sebagai bilah aplikasi (daripada ActionBar default) untuk aktivitas, lakukan salah satu hal berikut:

- Mulai proyek Anda dengan template Basic Activity, yang mengimplementasikan Bilah Alat untuk aktivitas sekaligus menu opsi dasar (dengan satu item, **Settings**). Anda bisa melewati bagian ini.
- Lakukan sendiri, seperti yang ditampilkan dalam bagian ini:

1. Tambahkan pustaka dukungan: appcompat dan desain.
2. Gunakan tema dan gaya NoActionBar untuk bilah aplikasi dan latar belakang.
3. Tambahkan AppBarLayout dan Toolbar pada layout.
4. Tambahkan kode pada aktivitas untuk mempersiapkan bilah aplikasi.

Menambahkan pustaka dukungan

Jika Anda memulai suatu proyek aplikasi menggunakan template Basic Activity, template akan menambahkan pustaka dukungan berikut untuk Anda, sehingga Anda bisa melewati langkah ini.

Jika Anda *tidak* menggunakan template Basic Activity, tambahkan pustaka dukungan appcompat (versi saat ini adalah v7) untuk kelas Toolbar, dan pustaka desain untuk tema NoActionBar, ke proyek Anda:

1. Pilih **Tools > Android > SDK Manager** untuk memeriksa apakah Android Support Repository telah dipasang; jika belum, pasanglah.
2. Buka file build.gradle untuk aplikasi Anda, dan tambahkan identifier proyek fitur pustaka dukungan ke bagian `dependencies` . Misalnya, untuk menyertakan `support:appcompat` dan `support:design` , tambahkan:

   ```
   compile 'com.android.support:appcompat-v7:23.4.0'
   compile 'com.android.support:design:23.4.0'
   ```

   Catatan: Perbarui nomor versi untuk dependensi jika perlu. Jika nomor versi yang Anda tetapkan lebih rendah dari nomor versi pustaka yang tersedia saat ini, Android Studio akan memperingatkan Anda ("a newer version of com.android.support:design is available"). Perbarui nomor versi ke versi yang disebutkan Android Studio untuk digunakan.

Menggunakan tema untuk mendesain bilah aplikasi

Jika Anda memulai suatu proyek aplikasi menggunakan template Basic Activity, template tersebut akan menambahkan tema untuk menggantikan ActionBar dengan `Toolbar` , sehingga Anda bisa melewati langkah ini.

Jika Anda *tidak* menggunakan template Basic Activity, Anda bisa menggunakan kelas `Toolbar` untuk bilah aplikasi dengan mematikan ActionBar default menggunakan tema NoActionBar untuk aktivitas tersebut. Tema dalam Android serupa dengan gaya, kecuali bahwa tema diterapkan ke seluruh aplikasi atau aktivitas, bukan ke tampilan tertentu.

Bila Anda membuat proyek baru di Android Studio, sebuah tema aplikasi secara otomatis dibuat untuk Anda. Misalnya, jika Anda memulai suatu proyek aplikasi dengan template Basic Activity atau Empty Activity, tema `AppTheme` disediakan dalam direktori **styles.xml** dalam **res > values**.

Tip: Anda akan mengetahui selengkapnya tentang tema dalam bab mengenai sumber daya dapat digambar, gaya, dan tema.

Anda bisa memodifikasi tema untuk menyediakan gaya bagi bilah aplikasi dan latar belakang aplikasi, sehingga bilah aplikasi akan terlihat dan menonjol dibandingkan latar belakangnya. Ikuti langkah-langkah ini:

1. Buka file **styles.xml**. Anda seharusnya sudah memiliki yang berikut ini dalam file:

   ```
   <resources>
      <!-- Base application theme. -->
      <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>
   ```

`AppTheme` "mewarisi"—menggunakan semua gaya—dari induk yang disebut `Theme.AppCompat.Light.DarkActionBar`, yaitu tema standar yang disediakan bersama Android. Akan tetapi, Anda bisa mengganti gaya yang diwarisi bersama gaya lain dengan menambahkan gaya lain ke **styles.xml**.

2. Tambahkan gaya `AppTheme.NoActionBar`, `AppTheme.AppBarOverlay`, dan `AppTheme.PopupOverlay` pada gaya `AppTheme`, seperti yang ditampilkan di bawah ini. Semua gaya ini akan menggantikan atribut gaya dengan nama yang sama dalam `AppTheme`, memengaruhi penampilan bilah aplikasi dan latar belakang aplikasi:

```
<resources>
   <!-- Base application theme. -->
   <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
      ...
    </style>

  <style name="AppTheme.NoActionBar">
      <item name="windowActionBar">false</item>
      <item name="windowNoTitle">true</item>
  </style>

  <style name="AppTheme.AppBarOverlay"
            parent="ThemeOverlay.AppCompat.Dark.ActionBar" />

  <style name="AppTheme.PopupOverlay"
            parent="ThemeOverlay.AppCompat.Light" />
   ...
</resources>
```

3. In the AndroidManifest.xml file, add the NoActionBar theme in appcompat to the `<application>` element. Using this theme prevents the app from using the native ActionBar class to provide the app bar:

```
<activity
   ...
   android:theme="@style/AppTheme.NoActionBar">
</activity>
```

Menambahkan AppBarLayout dan Bilah Alat pada layout

Jika Anda memulai suatu proyek aplikasi menggunakan template Basic Activity, template tersebut akan menambahkan tema `AppBarLayout` dan `Toolbar` untuk Anda, sehingga Anda bisa melewati langkah ini.

Jika Anda *tidak* menggunakan template Basic Activity, Anda bisa menyertakan Toolbar dalam layout aktivitas dengan menambahkan elemen `AppBarLayout` dan `Toolbar` .AppBarLayout adalah LinearLayout vertikal yang mengimplementasikan banyak fitur konsep bilah aplikasi desain material, seperti isyarat menggulir. Ingat hal berikut:

- `AppBarLayout` harus merupakan anak langsung dalam grup tampilan akar `CoordinatorLayout`, dan `Toolbar` harus merupakan anak langsung dalam `AppBarLayout`, seperti yang ditampilkan di bawah ini:

```
<android.support.design.widget.CoordinatorLayout
   ... >
   <android.support.design.widget.AppBarLayout
      ...>
      <android.support.v7.widget.Toolbar
            ...
            />
   </android.support.design.widget.AppBarLayout>
   ...
</android.support.design.widget.CoordinatorLayout>
```

- Posisikan `Toolbar` di bagian atas layout aktivitas, karena Anda menggunakannya sebagai bilah aplikasi.
- `AppBarLayout` juga memerlukan layout materi tersendiri yang seinduk untuk materi yang digulir di bawah bilah aplikasi. Anda bisa menambahkan saudara ini sebagai grup tampilan (seperti RelativeLayout atau LinearLayout) seperti berikut:
  - Di dalam file layout yang sama untuk aktivitas (seperti dalam **activity_main.xml**)
  - Dalam file layout tersendiri, seperti **content_main.xml**, yang kemudian bisa Anda tambahkan ke file layout

aktivitas dengan pernyataan include:

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

- Anda perlu menyetel perilaku menggulir materi yang seinduk, seperti yang ditampilkan di bawah ini dengan grup RelativeLayout, untuk menjadi instance dari `AppBarLayout.ScrollingViewBehavior`:

```
<RelativeLayout
   ...
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   app:layout_behavior="@string/appbar_scrolling_view_behavior"
   ... >

</RelativeLayout>
```

Perilaku layout untuk `RelativeLayout` disetel ke sumber daya string `@string/appbar_scrolling_view_behavior`, yang mengontrol perilaku gulir layar sehubungan dengan bilah aplikasi di bagian atas. Sumber daya string menyatakan string berikut, yang didefinisikan dalam file values.xml yang tidak boleh diedit:

```
android.support.design.widget.AppBarLayout$ScrollingViewBehavior
```

Perilaku ini didefinisikan oleh kelas AppBarLayout.ScrollingViewBehavior. Perilaku ini harus digunakan oleh Tampilan yang bisa menggulir secara vertikal—perilaku ini mendukung pengguliran tersarang untuk menggulir AppBarLayout yang seinduk secara otomatis.

Menambahkan kode untuk mempersiapkan bilah aplikasi

Jika Anda memulai suatu proyek aplikasi menggunakan template Basic Activity, template tersebut akan menambahkan kode yang diperlukan untuk mempersiapkan bilah aplikasi, sehingga Anda bisa melewati langkah ini.

Jika *tidak* menggunakan template Basic Activity, Anda bisa mengikuti langkah-langkah ini untuk mempersiapkan bilah aplikasi dalam aktivitas:

1. Pastikan bahwa aktivitas tempat Anda ingin menampilkan bilah aplikasi memperluas AppCompatActivity:

```
public class MyActivity extends AppCompatActivity {
...
}
```

2. In the activity's `onCreate()` method, call the activity's setSupportActionBar() method, and pass the activity's toolbar (assuming the Toolbar element's id is `toolbar`). The `setSupportActionBar()` method sets the toolbar as the app bar for the activity:

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

Aktivitas sekarang menampilkan bilah aplikasi. Secara default, bilah aplikasi berisi nama aplikasi saja.

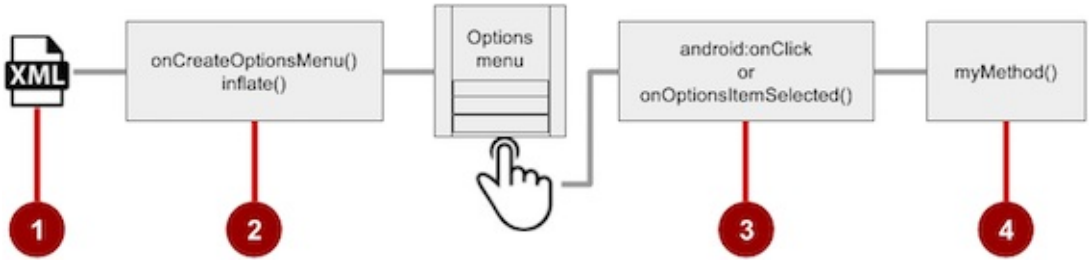
Menambahkan menu opsi

Android menyediakan format XML standar untuk mendefinisikan item menu opsi. Sebagai ganti membangun menu dalam kode aktivitas, Anda bisa mendefinisikan menu dan semua itemnya dalam sumber daya menu XML. Sumber daya menu mendefinisikan menu aplikasi (menu opsi, menu konteks, atau menu munculan) yang bisa dimekarkan dengan

MenuInflater, yang memuat sumber daya sebagai objek Menu dalam aktivitas atau fragmen.

Jika Anda memulai suatu proyek aplikasi menggunakan template Basic Activity, template akan menambahkan sumber daya menu untuk Anda dan memekarkan menu opsi dengan MenuInflater, sehingga Anda bisa melewati langkah ini dan langsung ke "Mendefiniskan cara item menu muncul".

Jika Anda *tidak* menggunakan template Basic Activity, gunakan pola desain pemekaran-sumber-daya, yang memudahkan dalam membuat menu opsi. Ikuti langkah-langkah ini (lihat gambar di bawah ini):



1. **Sumber daya menu XML**. Buat file sumber daya menu XML untuk item menu, dan tetapkan atribut penampilan dan posisi seperti yang dijelaskan di bagian berikutnya.
2. **Memekarkan menu**. Ganti metode onCreateOptionsMenu() dalam aktivitas atau fragmen Anda untuk memekarkan menu.
3. **Menangani klik item menu**. Item menu adalah tampilan, sehingga Anda bisa menggunakan atribut `android:onClick` untuk setiap item menu. Akan tetapi, metode onOptionsItemSelected() bisa menangani semua klik item menu pada satu tempat, dan menentukan item menu mana yang diklik, yang membuat kode Anda mudah dipahami.
4. **Melakukan aksi**. Buat metode untuk melakukan aksi bagi setiap item menu opsi.

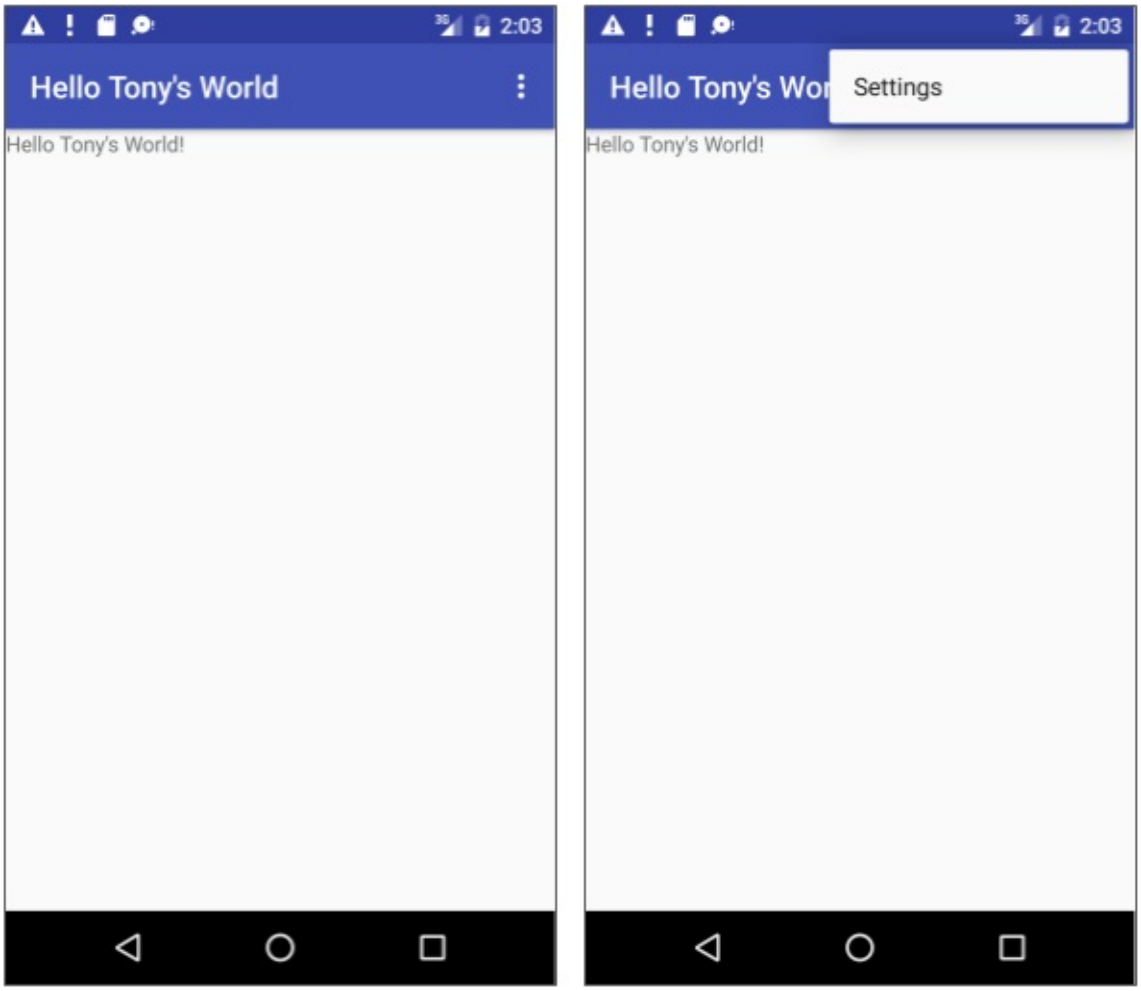
Membuat sumber daya XML untuk menu

Ikuti langkah-langkah ini untuk menambahkan item menu ke sumber daya menu XML:

1. Klik direktori **res**, dan pilih **File > New > Android resource directory**, pilih **menu** dalam menu tarik-turun Resource type, dan klik **OK**.
2. Klik direktori **menu** baru, dan pilih **File > New > Menu resource file**, masukkan nama file sebagai **menu_main** atau yang semacamnya, dan klik **OK**. File menu_main.xml sekarang ada di dalam direktori menu.
3. Buka file **menu_main.xml** (jika belum terbuka), dan klik tab **Text** di sebelah tab Design di bagian bawah panel untuk menampilkan file.
4. Tambahkan item menu opsi pertama menggunakan tag `<item … />` . Dalam contoh ini, item tersebut adalah **Settings**:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
   ...>
   <item
      android:id="@+id/action_settings"
      android:title="@string/settings" />
</menu>
```

Setelah mempersiapkan dan memekarkan sumber daya XML dalam kode aktivitas atau fragmen, ikon luapan dalam bilah aplikasi yang diklik akan menampilkan menu opsi hanya dengan satu opsi (**Settings**):



Mendefinisikan cara item menu muncul

Jika Anda memulai suatu proyek aplikasi menggunakan template Basic Activity, template tersebut akan menambahkan menu opsi dengan satu opsi: **Settings**.

Untuk menambahkan lebih banyak item menu opsi, tambahkan lebih banyak tag `<item … />` dalam file **menu_main.xml**. Misalnya, dua item menu opsi didefinisikan dalam cuplikan berikut: `@string/settings` (**Settings**) dan `@string/action_order` (**Order**):

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
   ...>
   <item
      android:id="@+id/action_settings"
      android:title="@string/settings" />
   <item
      android:id="@+id/action_order"
      android:icon="@drawable/ic_order_white"
      android:title="@string/action_order"/>
</menu>
```

Dalam setiap tag `<item … />` Anda bisa menambahkan atribut untuk mendefinisikan cara item menu muncul, seperti urutan penampilannya terkait item lain, dan apakah item tersebut bisa muncul sebagai ikon dalam bilah aplikasi. Item yang Anda setel menjadi *tidak* muncul dalam bilah aplikasi (atau tidak pas dalam bilah aplikasi karena orientasi tampilan) ditempatkan berurutan dalam menu luapan).

Jika memungkinkan, Anda bisa menampilkan aksi yang paling sering digunakan menggunakan ikon dalam bilah aplikasi, sehingga pengguna bisa mengekliknya tanpa harus mengeklik tombol luapan terlebih dahulu.

Menambahkan ikon untuk item menu

Untuk menetapkan ikon untuk aksi, Anda perlu terlebih dulu menambahkan ikon sebagai aset gambar ke folder **drawable** dengan mengikuti langkah-langkah ini (lihat Image Asset Studio untuk keterangan lengkap):

1. Luaskan **res** dalam tampilan Project, dan klik-kanan (atau klik Command) **drawable**.
2. Pilih **New > Image Asset**. Dialog Configure Image Asset akan muncul.
3. Pilih **Action Bar and Tab Items** dalam menu tarik-turun.
4. Edit nama ikon (misalnya, **ic_order_white** untuk item menu Order).
5. Klik gambar clipart (logo Android) untuk memilih gambar clipart sebagai ikon. Laman ikon akan muncul. Klik ikon yang ingin Anda gunakan.
6. (Opsional) Pilih **HOLO_DARK** dari menu tarik-turun Tema. Ini akan menyetel ikon menjadi putih dengan latar belakang berwarna gelap (atau hitam). Klik **Next**.
7. Klik **Finish**, dalam dialog Konfirmasi Path Ikon.

Atribut penampilan dan ikon

Gunakan atribut berikut untuk menentukan penampilan item menu:

- `android:icon` : Gambar yang digunakan sebagai ikon item menu. Misalnya, item menu berikut mendefinisikan `ic_order_white` sebagai ikonnya:

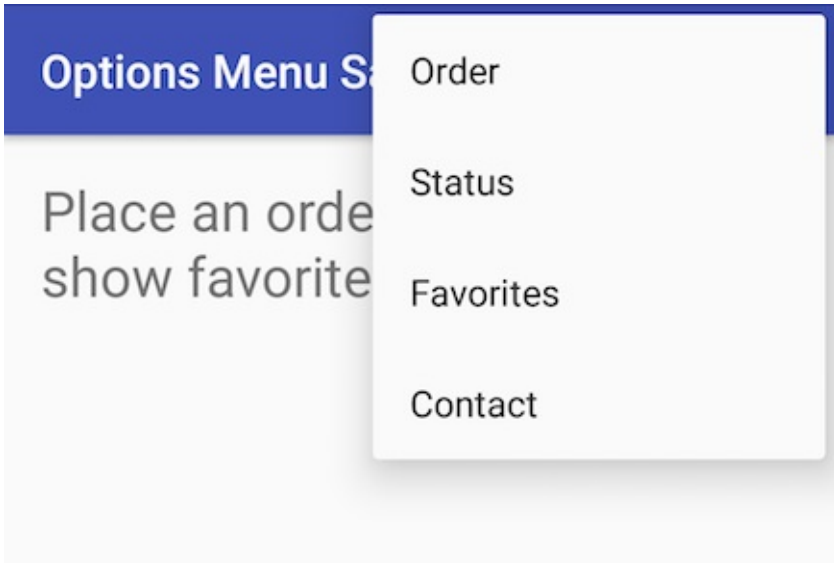
```
<item
    android:id="@+id/action_order"
    android:icon="@drawable/ic_order_white"
    android:title="@string/action_order"/>
```

- `android:title` : String untuk judul item menu.
- `android:titleCondensed` : String yang digunakan sebagai judul singkat untuk situasi di mana judul normal terlalu panjang.

Posisikan atribut

Gunakan atribut `android:orderInCategory` untuk menetapkan urutan kemunculan item menu di menu, dengan angka terkecil muncul lebih tinggi dalam menu. Ini biasanya urutan kepentingan item di dalam menu. Misalnya, jika Anda ingin **Order** sebagai yang pertama, diikuti oleh **Status**, **Favorites**, dan **Contact**, tabel berikut menampilkan prioritas item ini dalam menu:

Item menu	`orderInCategory attribute`
Order:	10
Status	20
Favorites	40
Contact	100

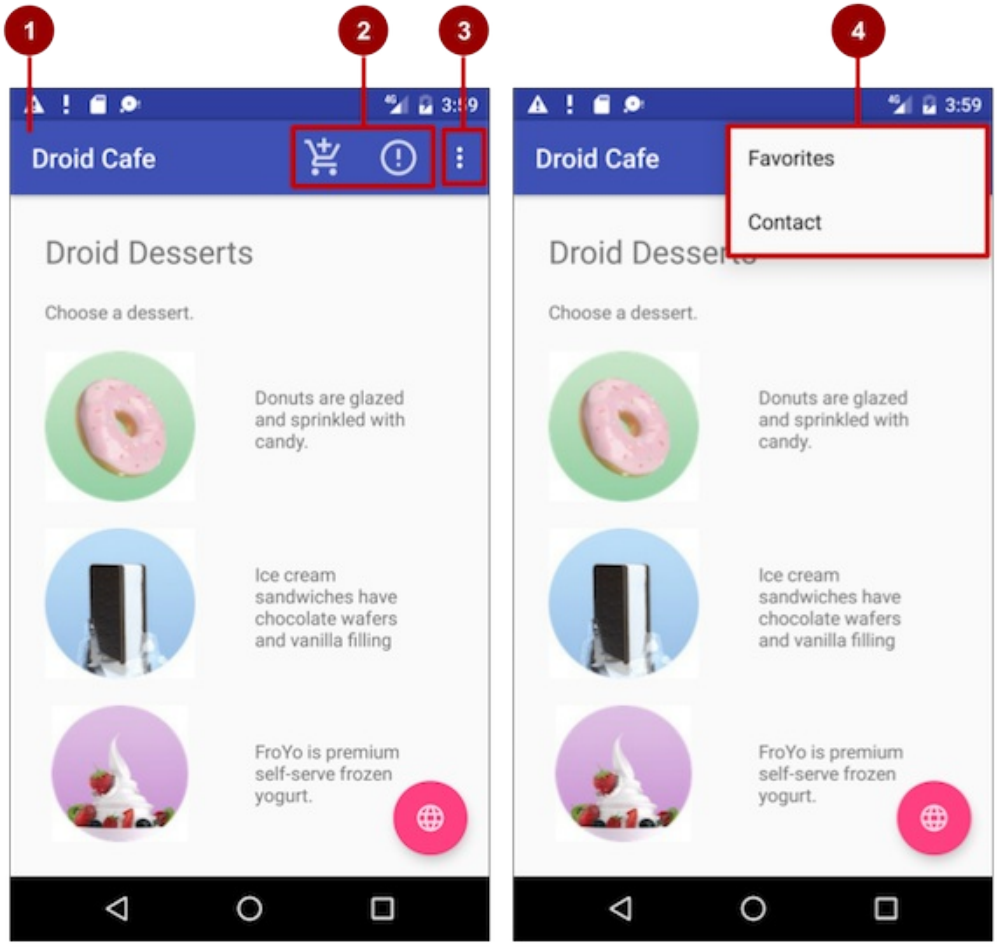


Catatan: Sementara angka 1, 2, 3, dan 4 juga bekerja dalam contoh di atas, angka 10, 20, 40, dan 100 meninggalkan ruangan untuk item menu tambahan yang akan ditambahkan di antara mereka nantinya.
Gunakan atribut `app:showAsAction` untuk menampilkan item menu sebagai ikon dalam bilah aplikasi, dengan nilai sebagai berikut:

- `"always"` : Selalu tempatkan item ini dalam bilah aplikasi. Gunakan ini hanya jika item tersebut penting untuk muncul dalam bilah aplikasi (seperti ikon Telusur). Jika Anda menyetel beberapa item untuk selalu muncul dalam bilah aplikasi, mereka bisa menindih item lain dalam bilah aplikasi, seperti judul aplikasi.
- `"ifRoom"` : Hanya tempatkan item ini dalam bilah aplikasi jika ada ruang untuk item tersebut. Jika tidak ada cukup ruang untuk semua item yang ditandai `"ifRoom"` , item dengan nilai `orderInCategory` terendah akan ditampilkan dalam bilah aplikasi, dan item lainnya akan ditampilkan dalam menu luapan.
- `"never"` : Jangan pernah tempatkan item ini dalam bilah aplikasi. Sebagai gantinya, cantumkan item dalam menu luapan bilah aplikasi.
- `"withText"` : Sertakan juga teks judul (didefinisikan oleh `android:title`) dengan item tersebut. Teks judul akan muncul jika item muncul dalam menu luapan, sehingga atribut digunakan terutama menyertakan judul dengan ikon dalam bilah aplikasi.

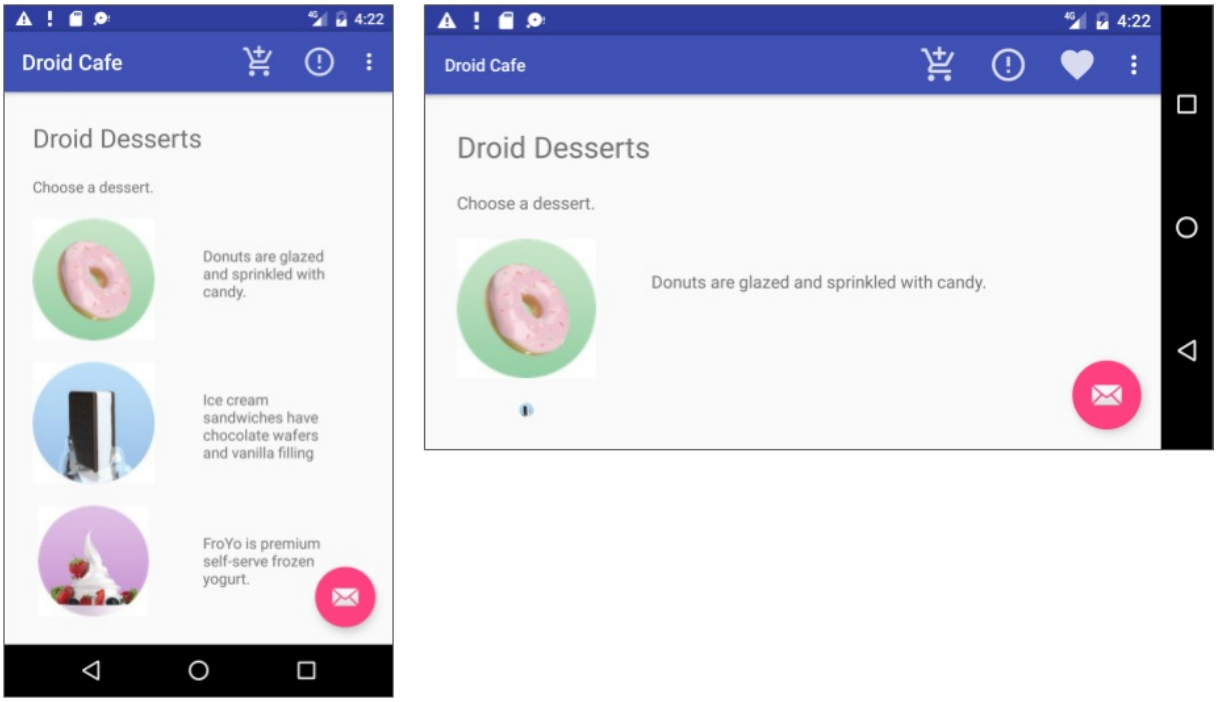
Misalnya, ikon item menu berikut muncul dalam bilah aplikasi jika ada ruang untuk ikon tersebut:

```
<item
   android:id="@+id/action_favorites"
   android:icon="@drawable/ic_favorites_white"
   android:orderInCategory="40"
   android:title="@string/action_favorites"
   app:showAsAction="ifRoom" />
```



Dalam gambar di atas:

1. **Ikon aksi menu opsi**. Dua item menu opsi pertama muncul sebagai ikon aksi dalam bilah aplikasi: **Order** (ikon keranjang belanja) dan **Info** (ikon "**i**").
2. **Tombol luapan**. Mengeklik tombol luapan akan menampilkan menu luapan.
3. **Menu luapan opsi**. Menu luapan menampilkan lebih banyak menu opsi: **Favorites** dan **Contact**. **Favorites** (ikon hati) tidak pas dalam bilah aplikasi dengan orientasi vertikal, namun muncul dalam orientasi horizontal pada ponsel cerdas, atau dalam kedua orientasi pada tablet, seperti yang ditampilkan di bawah ini.



Memekarkan sumber daya menu

Jika Anda memulai suatu proyek aplikasi menggunakan template Basic Activity, template tersebut akan menambahkan kode untuk memekarkan menu opsi dengan MenuInflater, sehingga Anda bisa melewati langkah ini.

Jika Anda *tidak* menggunakan template Basic Activity, mekarkan sumber daya menu dalam aktivitas Anda dengan menggunakan metode onCreateOptionsMenu() (dengan anotasi `Override`) dengan metode getMenuInflater() dari kelas Activity.

Metode `getMenuInflater()` akan mengembalikan MenuInflater, yaitu kelas yang digunakan untuk membuat instance file XML menu menjadi objek Menu. Kelas MenuInflater menyediakan metode inflate(), yang memerlukan `id` sumber daya untuk sumber daya layout XML yang akan dimuat (`R.menu.menu_main` dalam contoh berikut), dan Menu dimekarkan (`menu` dalam contoh berikut):

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   getMenuInflater().inflate(R.menu.menu_main, menu);
   return true;
}
```

Menangani klik item menu

Sebagaimana dengan tombol, atribut `android:onClick` mendefinisikan metode untuk memanggil saat item menu ini diklik. Anda harus mendeklarasikan metode dalam aktivitas ini sebagai `public` dan menerima `MenuItem` sebagai satu-satunya parameter, yang menunjukkan item yang diklik.

Misalnya, Anda bisa mendefinisikan item **Favorites** dalam file sumber daya menu untuk menggunakan atribut `android:onClick` untuk memanggil metode `onFavoritesClick()`:

```
<item
        android:id="@+id/action_favorites"
        android:icon="@drawable/ic_favorites_white"
        android:orderInCategory="40"
        android:title="@string/action_favorites"
        app:showAsAction="ifRoom"
        android:onClick="onFavoritesClick" />
```

Anda harus mendeklarasikan metode `onFavoritesClick()` dalam aktivitas:

```
public void onFavoritesClick(MenuItem item) {
    // The item parameter indicates which item was clicked.
    // Add code to handle the Favorites click.
}
```

Akan tetapi, metode onOptionsItemSelected() dari kelas aktivitas bisa menangani semua klik item menu pada satu tempat, dan menentukan item menu mana yang telah diklik, sehingga membuat kode Anda lebih mudah dipahami. Template Basic Activity menyediakan implementasi metode `onOptionsItemSelected()` dengan sebuah blok `switch case` untuk memanggil metode yang sesuai (seperti `showOrder`) berdasarkan `id` item menu, yang bisa Anda ambil menggunakan metode getItemId() dari kelas Adapter:

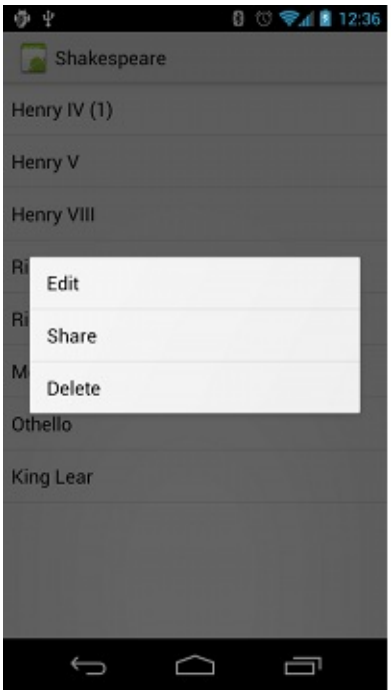
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   switch (item.getItemId()) {
      case R.id.action_order:
         showOrder();
         return true;
      case R.id.action_status:
         showStatus();
         return true;
      case R.id.action_contact:
         showContact();
         return true;
      default:
         // Do nothing
   }
   return super.onOptionsItemSelected(item);
}
```

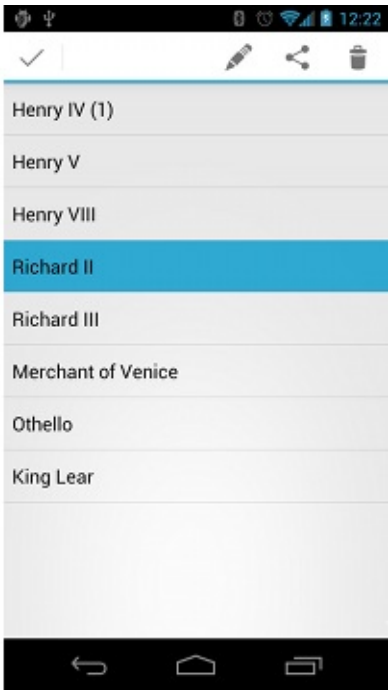
Menu kontekstual

Gunakan *menu kontekstual* untuk memungkinkan pengguna melakukan aksi pada tampilan yang dipilih. Anda bisa menyediakan menu konteks untuk setiap Tampilan, namun menu konteks paling sering digunakan untuk item dalam RecyclerView, GridView, atau kumpulan tampilan lainnya, di mana pengguna bisa melakukan tindakan langsung pada setiap item.

Android menyediakan dua jenis menu kontekstual:

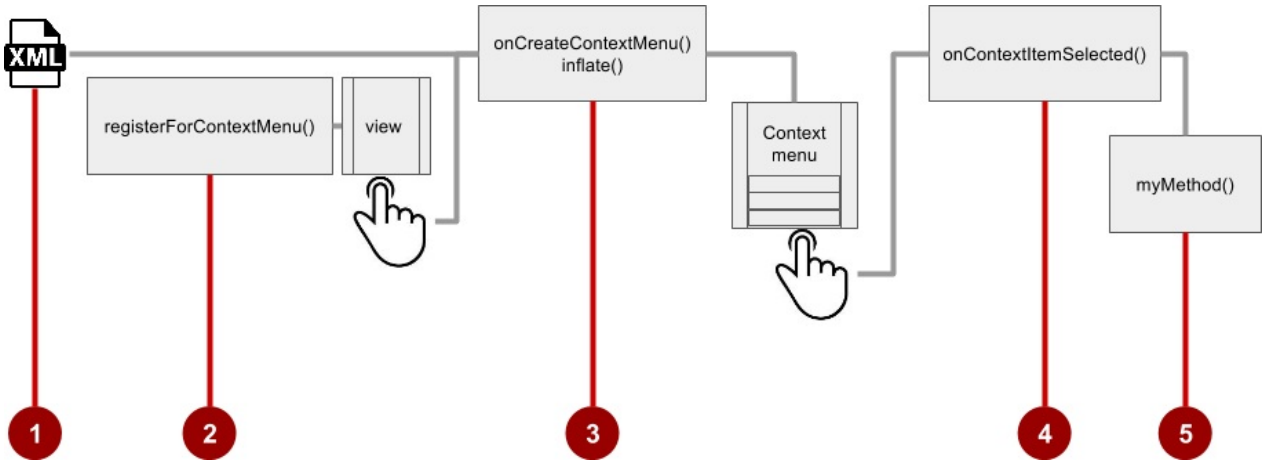
- *Menu konteks*, ditampilkan pada sisi kiri dalam gambar di bawah ini, muncul sebagai daftar item menu mengambang bila pengguna melakukan ketukan lama pada elemen tampilan di layar. Menu konteks umumnya digunakan untuk memodifikasi elemen tampilan atau menggunakannya dalam beberapa cara. Misalnya, menu konteks dapat menyertakan **Edit** untuk mengedit elemen tampilan, **Delete** untuk menghapusnya, dan **Share** untuk membagikannya melalui media sosial. Pengguna bisa melakukan aksi kontekstual pada satu elemen tampilan untuk setiap kalinya.
- *Bilah aksi kontekstual*, yang ditampilkan di sisi kanan gambar di bawah ini, muncul di bagian atas layar di tempat bilah aplikasi atau di bawah bilah aplikasi, bersama item aksi yang memengaruhi elemen tampilan yang dipilih. Pengguna bisa melakukan aksi pada beberapa elemen tampilan sekaligus (jika aplikasi Anda memungkinkannya).





Menu konteks mengambang

Pola desain pemekaran-sumber-daya yang familier digunakan untuk membuat menu konteks mengambang, yang dimodifikasi untuk menyertakan pendaftaran (asosiasi) menu konteks dengan suatu tampilan:



Ikuti langkah-langkah ini untuk membuat menu konteks mengambang bagi satu atau beberapa elemen tampilan (lihat gambar di atas):

1. Buat file sumber daya menu XML untuk item menu, tetapkan atribut penampilan dan posisi (seperti yang dijelaskan di bagian sebelumnya).
2. Daftarkan tampilan ke menu konteks dengan menggunakan metode registerForContextMenu() dari kelas Activity.
3. Implementasikan metode onCreateContextMenu() dalam aktivitas atau fragmen Anda untuk memekarkan menu.
4. Implementasikan metode onContextItemSelected() dalam aktivitas atau fragmen Anda untuk menangani klik item menu.
5. Buat metode untuk melakukan aksi bagi setiap item menu konteks.

Membuat file sumber daya XML

Buat file dan direktori sumber daya menu XML dengan mengikuti langkah-langkah di bagian sebelumnya. Gunakan nama yang sesuai untuk file tersebut, seperti `menu_context`. Tambahkan item menu konteks (dalam contoh ini, item menu adalah **Edit**, **Share**, dan **Delete**):

```
<item
   android:id="@+id/context_edit"
   android:title="@string/edit"
   android:orderInCategory="10"/>

<item
   android:id="@+id/context_share"
   android:title="@string/share"
   android:orderInCategory="20"/>

<item
   android:id="@+id/context_delete"
   android:title="@string/delete"
   android:orderInCategory="30"/>
```

Mendaftarkan tampilan ke menu konteks

Daftarkan tampilan ke menu konteks dengan memanggil metode registerForContextMenu() dan meneruskan tampilan padanya. Mendaftarkan menu konteks untuk tampilan akan menyetel View.OnCreateContextMenuListener pada tampilan untuk aktivitas ini, sehingga onCreateContextMenu() akan dipanggil saat tiba waktunya menampilkan menu konteks. (Anda akan mengimplementasikan `onCreateContextMenu` di bagian berikutnya.)

Misalnya, dalam metode `onCreate()` untuk aktivitas, tambahkan pernyataan `registerForContextMenu()` :

```
...
// Registering the context menu to the text view of the article.
TextView article_text = (TextView) findViewById(R.id.article);
registerForContextMenu(article_text);
...
```

Beberapa tampilan bisa didaftarkan ke menu konteks yang sama. Jika Anda ingin setiap item dalam ListView atau GridView menyediakan menu konteks yang sama, daftarkan semua item untuk menu konteks dengan meneruskan ListView atau GridView ke registerForContextMenu().

Mengimplementasikan metode onCreateContextMenu()

Bila tampilan yang telah didaftarkan menerima kejadian klik-lama, sistem akan memanggil metode onCreateContextMenu() yang bisa Anda ganti dalam aktivitas atau fragmen. Inilah tempat Anda mendefinisikan item menu, biasanya dengan memekarkan sumber daya menu.

Misalnya:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                        ContextMenu.ContextMenuInfo menuInfo) {
   super.onCreateContextMenu(menu, v, menuInfo);
   MenuInflater inflater = getMenuInflater();
   inflater.inflate(R.menu.menu_context, menu);
}
```

Dalam kode di atas:

- Metode `menu` untuk onCreateContextMenu() adalah menu konteks yang akan dibangun.
- Metode `v` adalah tampilan yang didaftarkan untuk menu konteks.
- Metode `menuInfo` adalah informasi ekstra tentang tampilan yang didaftarkan untuk menu konteks. Informasi ini bervariasi, bergantung pada kelas **v**, yang bisa berupa RecyclerView atau GridView. Jika Anda mendaftarkan RecyclerView atau GridView, Anda harus membuat instance objek ContextMenu.ContextMenuInfo untuk menyediakan informasi tentang item yang dipilih, dan meneruskannya sebagai `menuInfo`, seperti ID baris, posisi, atau tampilan anak.

Kelas MenuInflater menyediakan metode inflate(), yang memerlukan ID sumber daya sebagai parameter, untuk sumber daya layout XML yang akan dimuat (`menu_context` dalam contoh di atas), dan Menu dimekarkan menjadi (`menu` dalam contoh di atas).

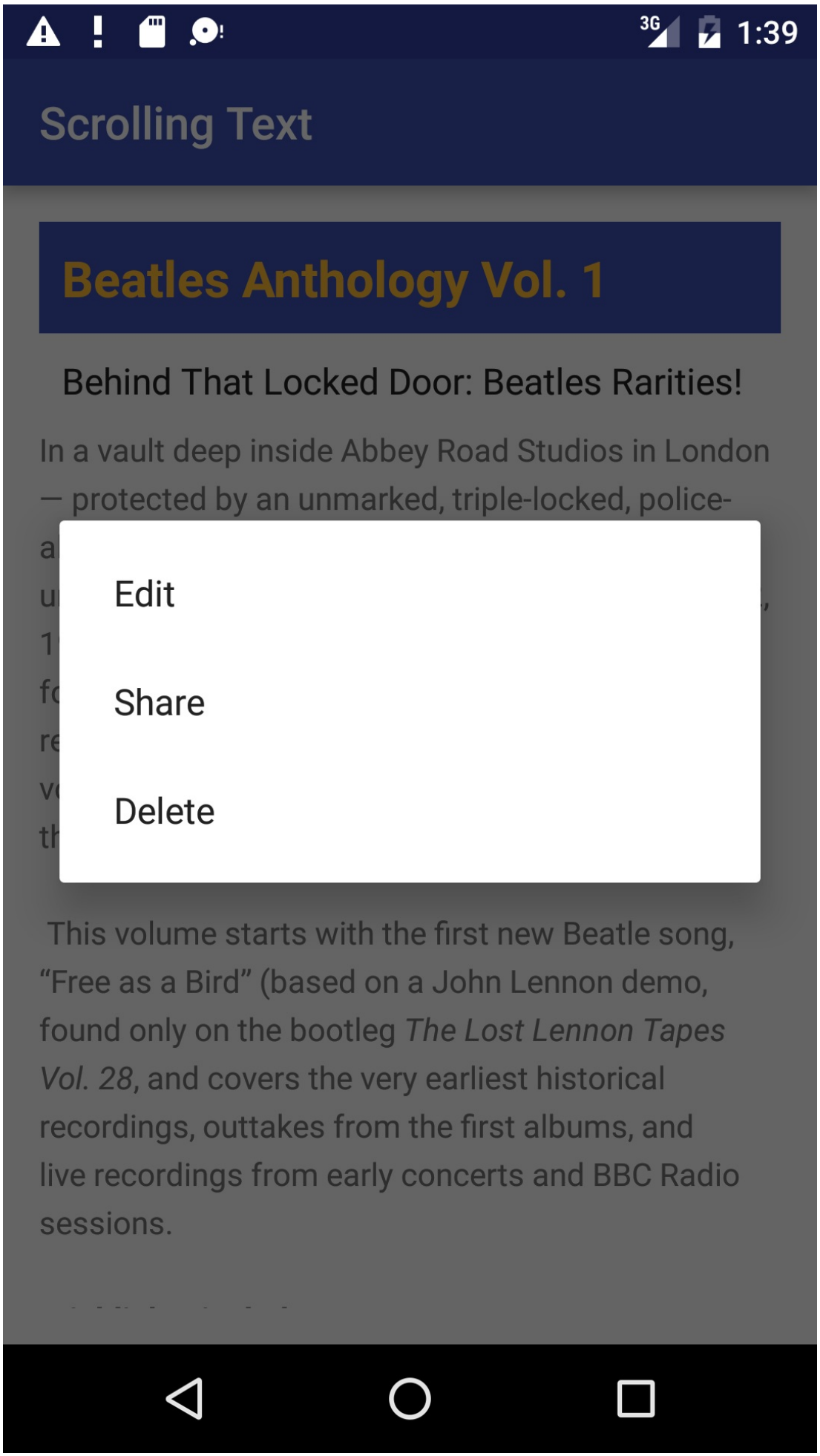
Mengimplementasikan metode onContextItemSelected()

Bila pengguna mengeklik sebuah item menu, sistem akan memanggil metode onContextItemSelected(). Anda bisa mengganti metode ini dalam aktivitas atau fragmen untuk menentukan item menu mana yang diklik, dan untuk tampilan mana menu tersebut dimunculkan. Anda juga bisa menggunakannya untuk mengimplementasikan aksi yang sesuai untuk item menu, seperti `editNote()` dan `shareNote()` di bawah untuk item menu **Edit** dan **Share**. Misalnya:

```
@Override
public boolean onContextItemSelected(MenuItem item) {
   switch (item.getItemId()) {
      case R.id.context_edit:
         editNote();
         return true;
      case R.id.context_share:
         shareNote();
         return true;
      default:
         return super.onContextItemSelected(item);
   }
}
```

Contoh di atas menggunakan metode getItemId() untuk mendapatkan `id` bagi item menu yang dipilih, dan menggunakannya dalam blok `switch case` untuk menentukan aksi mana yang akan dilakukan. Metode `id` adalah atribut `android:id` yang ditetapkan untuk item menu dalam file sumber daya menu XML.

Bila pengguna melakukan klik-lama pada artikel dalam tampilan teks, menu konteks mengambang akan muncul dan pengguna bisa mengeklik item menu.


1:39
Scrolling Text
Beatles Anthology Vol. 1
Behind That Locked Door: Beatles Rarities!
In a vault deep inside Abbey Road Studios in London — protected by an unmarked, triple-locked, police-
Edit
Share
Delete
This volume starts with the first new Beatle song, “Free as a Bird” (based on a John Lennon demo, found only on the bootleg *The Lost Lennon Tapes Vol. 28*, and covers the very earliest historical recordings, outtakes from the first albums, and live recordings from early concerts and BBC Radio sessions.

Jika Anda menggunakan informasi `menuInfo` untuk RecyclerView atau GridView, Anda bisa menambahkan pernyataan sebelum blok kasus peralihan untuk mengumpulkan informasi spesifik tentang tampilan yang dipilih (untuk `info`) dengan menggunakan AdapterView.AdapterContextMenuInfo:

```
AdapterView.AdapterContextMenuInfo info =
            (AdapterView.AdapterContextMenuInfo) item.getMenuInfo();
```

Bilah aksi kontekstual

Bilah aksi kontekstual muncul di bagian atas layar untuk menyajikan tindakan yang bisa dilakukan pengguna pada tampilan setelah mengeklik lama tampilan, seperti yang ditampilkan dalam gambar di bawah ini.

Dalam gambar di atas:

1. **Bilah aksi kontekstual**. Bilah ini menawarkan tiga tindakan pada sisi kanan (**Edit**, **Share**, dan **Delete**) serta tombol **Done** (ikon panah ke kiri) pada sisi kiri.
2. **Tampilan**. Tampilan di mana klik-lama akan memicu bilah aksi kontekstual.

Bilah aksi kontekstual muncul hanya saat *mode aksi kontekstual*, implementasi sistem ActionMode, terjadi akibat pengguna mengeklik lama pada Tampilan.

ActionMode menyatakan mode antarmuka pengguna (UI) untuk menyediakan interaksi alternatif, sehingga menggantikan bagian UI normal sampai selesai. Misalnya, pemilihan teks diimplementasikan sebagai ActionMode, sebagaimana tindakan kontekstual yang bekerja pada item yang dipilih di layar. Memilih bagian teks atau mengeklik lama suatu tampilan akan memicu ActionMode.

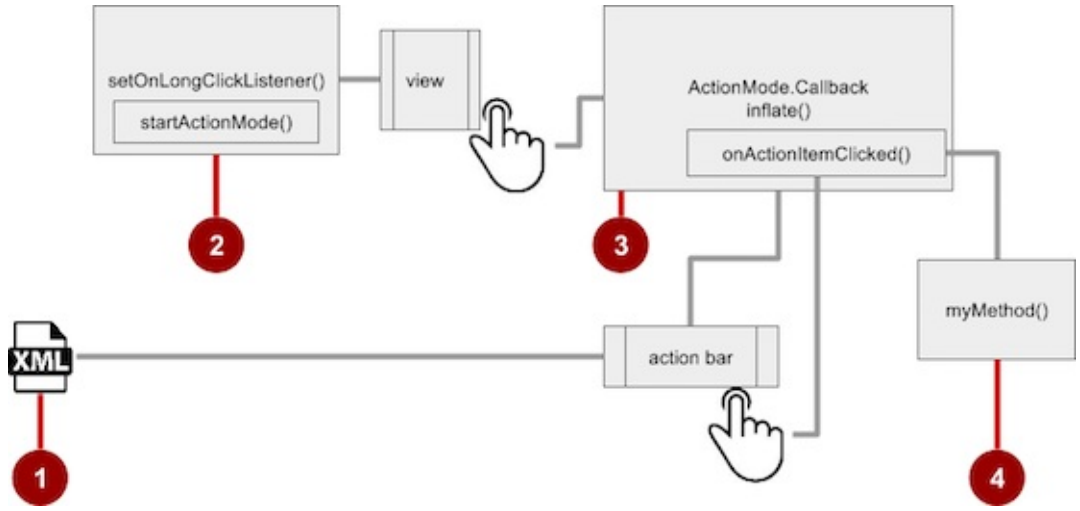
Saat mode ini diaktifkan, pengguna bisa memilih beberapa item (jika aplikasi Anda memungkinkannya), menghapus pilihan item, dan melanjutkan navigasi dalam aktivitas. Mode aksi dinonaktifkan dan bilah aksi kontekstual menghilang saat pengguna menghapus pilihan semua item, menekan tombol Return, atau mengetuk **Done** (ikon panah ke kiri) pada sisi kiri bilah.

Ikuti langkah-langkah ini untuk membuat bilah aksi kontekstual (lihat gambar di bawah ini):

1. Buat file sumber daya menu XML untuk item menu, dan tetapkan ikon untuk setiap item (seperti yang dijelaskan dalam bagian sebelumnya).
2. Setel listener klik-lama ke tampilan yang harus memicu bilah aksi kontekstual menggunakan metode

setOnLongClickListener(). Panggil startActionMode() dalam metode setOnLongClickListener() bila pengguna melakukan ketukan lama pada tampilan.

3. Implementasikan antarmuka ActionMode.Callback untuk menangani daur hidup ActionMode. Sertakan dalam antarmuka ini, aksi untuk merespons klik item menu dalam metode callback onActionItemClicked().
4. Buat metode untuk melakukan aksi bagi setiap item menu konteks.



Membuat file sumber daya XML

Buat file dan direktori sumber daya menu XML dengan mengikuti langkah-langkah di bagian sebelumnya. Gunakan nama yang sesuai untuk file tersebut, seperti `menu_context`. Tambahkan ikon untuk item menu konteks (dalam contoh ini, item menu adalah **Edit**, **Share**, dan **Delete**). Misalnya, item menu Edit akan memiliki atribut ini:

```
<item
   android:id="@+id/context_edit"
   android:orderInCategory="10"
   android:icon="@drawable/ic_action_edit_white"
   android:title="@string/edit" />
```

Bilah aksi kontekstual standar memiliki latar belakang gelap. Gunakan warna terang atau putih untuk ikon. Jika Anda menggunakan ikon clipart, pilih **HOLO_DARK** untuk menu tarik-turun Theme saat membuat aset gambar baru.

Menyetel listener klik-lama

Gunakan setOnLongClickListener() untuk menyetel listener klik-lama ke Tampilan yang harus memicu bilah aksi kontekstual. Tambahkan kode untuk menyetel listener klik-lama ke kelas aktivitas (seperti **MainActivity**) menggunakan metode `onCreate()` aktivitas. Ikuti langkah-langkah ini:

1. Deklarasikan variabel anggota `mActionMode` dalam definisi kelas untuk aktivitas:

   ```
   private ActionMode mActionMode;
   ```

   Anda harus memanggil startActionMode() untuk mengaktifkan ActionMode, yang akan mengembalikan ActionMode yang dibuat. Dengan menyimpannya dalam variabel anggota (`mActionMode` , Anda bisa membuat perubahan pada bilah aksi kontekstual sebagai respons terhadap kejadian lainnya.

2. Persiapkan listener bilah aksi kontekstual dalam metode `onCreate()` , menggunakan `View` sebagai tipe tampilan untuk menggunakan `setOnLongClickListener` :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   ...
   View articleView = findViewById(article);
   articleView.setOnLongClickListener(new View.OnLongClickListener() {
      ...
      // Add method here to start ActionMode after long-click.
      ...
   });
}
```

Mengimplementasikan antarmuka ActionMode.Callback

Sebelum Anda bisa menambahkan kode ke `onCreate()` untuk memulai ActionMode, Anda harus mengimplementasikan antarmuka ActionMode.Callback untuk mengelola daur hidup mode aksi. Dalam metode callback-nya, Anda bisa menetapkan aksi untuk bilah aksi kontekstual, dan merespons klik pada item aksi.

1. Tambahkan metode berikut ke kelas aktivitas (seperti **MainActivity**) untuk mengimplementasikan antarmuka:

   ```
   public ActionMode.Callback mActionModeCallback = new
                                      ActionMode.Callback() {
      ...
      // Add code to create action mode here.
      ...
   }
   ```

2. Add the `onCreateActionMode()` code within the brackets of the above method to create action mode (the full code is provided at the end of this section):

   ```
   @Override
   public boolean onCreateActionMode(ActionMode mode, Menu menu) {
     // Inflate a menu resource providing context menu items
     MenuInflater inflater = mode.getMenuInflater();
     inflater.inflate(R.menu.menu_context, menu);
     return true;
   }
   ```

   Metode onCreateActionMode() akan memekarkan menu dengan menggunakan pola yang sama dengan yang digunakan untuk menu konteks mengambang. Namun pemekaran ini *hanya* terjadi bila ActionMode telah dibuat, yaitu bila pengguna melakukan klik-lama. Kelas MenuInflater menyediakan metode inflate(), yang memerlukan `id` sumber daya untuk sumber daya layout XML yang akan dimuat (`menu_context` dalam contoh di atas), dan Menu dimekarkan menjadi (`menu` dalam contoh di atas).

3. Tambahkan metode onActionItemClicked() dengan penangan Anda untuk setiap item menu:

   ```
   @Override
   public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
   switch (item.getItemId()) {
     case R.id.context_edit:
        editNote();
        mode.finish();
        return true;
     case R.id.context_share:
        shareNote();
        mode.finish();
        return true;
     default:
        return false;
   }
   ```

Kode di atas menggunakan metode getItemId() untuk mendapatkan `id` bagi item menu yang dipilih, dan menggunakannya dalam blok `switch case` untuk menentukan aksi mana yang akan dilakukan. Metode `id` dalam setiap pernyataan `case` adalah atribut `android:id` yang ditetapkan untuk item menu dalam file sumber daya menu XML.

Aksi yang ditampilkan adalah metode `editNote()` dan `shareNote()` , yang bisa Anda buat dalam aktivitas yang sama. Setelah aksi dipilih, Anda bisa menggunakan metode `mode.finish()` untuk menutup bilah aksi kontekstual.

4. Tambahkan metode onPrepareActionMode() dan onDestroyActionMode(), yang mengelola daur hidup ActionMode:

```
@Override
public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
      return false; // Return false if nothing is done.
}
```

Metode `onPrepareActionMode()` yang ditampilkan di atas dipanggil setiap kali ActionMode terjadi, dan selalu dipanggil setelah `onCreateActionMode()` .

```
@Override
public void onDestroyActionMode(ActionMode mode) {
      mActionMode = null;
}
```

Metode `onDestroyActionMode()` yang ditampilkan di atas dipanggil bila pengguna keluar dari ActionMode dengan mengeklik **Done** dalam bilah aksi kontekstual, atau mengeklik pada tampilan yang berbeda.

5. Tinjau kode lengkap untuk implementasi antarmuka ActionMode.Callback:

```
public ActionMode.Callback mActionModeCallback = new
                                        ActionMode.Callback() {
   @Override
   public boolean onCreateActionMode(ActionMode mode, Menu menu) {
      // Inflate a menu resource providing context menu items
      MenuInflater inflater = mode.getMenuInflater();
      inflater.inflate(R.menu.menu_context, menu);
      return true;
   }

   // Called each time ActionMode is shown. Selalu dipanggil setelah
   // onCreateActionMode.
   @Override
   public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
      return false; // Return false if nothing is done
   }

   // Called when the user selects a contextual menu item
   @Override
   public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
      switch (item.getItemId()) {
         case R.id.context_edit:
            editNote();
            mode.finish();
            return true;
         case R.id.context_share:
            shareNote();
            mode.finish();
            return true;
         default:
            return false;
      }
   }

   // Called when the user exits the action mode
   @Override
   public void onDestroyActionMode(ActionMode mode) {
      mActionMode = null;
   }
};
```

Memulai ActionMode

Anda menggunakan startActionMode() untuk memulai ActionMode setelah pengguna melakukan klik-lama.

1. Untuk memulai ActionMode, tambahkan metode `onLongClick()` dalam tanda kurung untuk metode `setOnLongClickListener` dalam `onCreate()`:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   ...
   articleView.setOnLongClickListener(new View.OnLongClickListener() {
      // Called when the user long-clicks on articleView
      public boolean onLongClick(View view) {
         if (mActionMode != null) return false;
         // Start the contextual action bar
         // using the ActionMode.Callback.
         mActionMode =
               MainActivity.this.startActionMode(mActionModeCallback);
         view.setSelected(true);
         return true;
      }
   });
}
```

   Kode di atas terlebih dahulu memastikan bahwa instance `ActionMode` tidak dibuat lagi jika sudah aktif dengan memeriksa apakah `mActionMode` nol sebelum memulai mode aksi:

```
if (mActionMode != null) return false;
```

Saat pengguna melakukan klik-lama, panggilan akan dibuat ke `startActionMode()` dengan menggunakan antarmuka ActionMode.Callback, dan bilah aksi kontekstual muncul di bagian atas tampilan. Metode setSelected() mengubah keadaan tampilan ini menjadi dipilih (disetel ke `true`).

2. Tinjau kode untuk metode `onCreate()` dalam aktivitas, yang kini menyertakan `setOnLongClickListener()` dan `startActionMode()` :

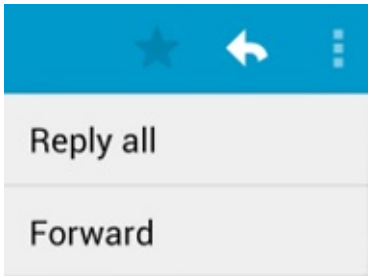
```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);

   // set up the contextual action bar listener
   View articleView = findViewById(article);
   articleView.setOnLongClickListener(new View.OnLongClickListener() {
      // Called when the user long-clicks on articleView
      public boolean onLongClick(View view) {
         if (mActionMode != null) return false;
         // Start the contextual action bar
         // using the ActionMode.Callback.
         mActionMode =
                MainActivity.this.startActionMode(mActionModeCallback);
         view.setSelected(true);
         return true;
      }
   });
}
```

Menu munculan

PopupMenu adalah daftar vertikal untuk item yang ditambatkan pada sebuah Tampilan. Munculnya di bawah tampilan jangkar jika ada ruang, atau di atas tampilan jika tidak ada.

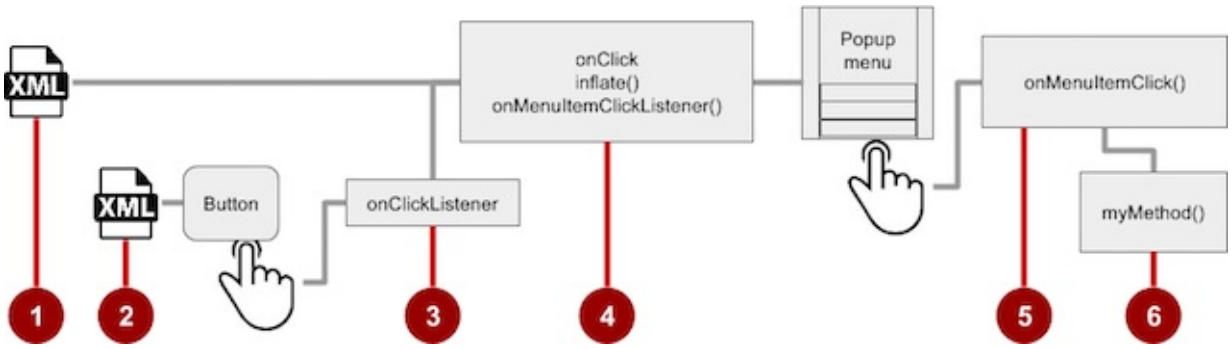
Menu munculan biasanya digunakan untuk menyediakan luapan aksi (serupa dengan ikon aksi luapan untuk menu opsi) atau bagian kedua dari perintah dua-bagian. Gunakan menu munculan untuk tindakan diperluas yang terkait dengan region materi dalam aktivitas Anda. Tidak seperti menu konteks, menu munculan yang ditambatkan ke sebuah Tombol (Tampilan), selalu tersedia, dan tindakannya secara umum tidak memengaruhi materi Tampilan.

Misalnya, aplikasi Gmail menggunakan menu munculan yang ditambatkan ke ikon luapan dalam bilah aplikasi saat menampilkan pesan email. Item menu munculan **Reply**, **Reply All**, dan **Forward** *terkait* dengan pesan email, namun tidak *memengaruhi* atau *berfungsi pada* pesan. Tindakan dalam menu munculan tidak boleh secara langsung memengaruhi materi yang bersangkutan (gunakan menu kontekstual untuk langsung memengaruhi materi yang dipilih). Seperti yang ditampilkan di bawah ini, sebuah munculan bisa ditambatkan ke tombol aksi luapan dalam bilah aksi.



Membuat menu munculan

Ikuti langkah-langkah ini untuk membuat menu munculan (lihat gambar di bawah ini):



1. Buat file sumber daya menu XML untuk item menu munculan, dan tetapkan atribut penampilan dan posisi (seperti yang dijelaskan di bagian sebelumnya).
2. Tambahkan ImageButton untuk ikon menu munculan dalam file layout aktivitas XML.
3. Tetapkan onClickListener() pada tombol.
4. Ganti metode `onClick()` untuk memekarkan menu munculan dan mendaftarkannya dengan PopupMenu.OnMenuItemClickListener.
5. Implementasikan metode onMenuItemClick().
6. Buat metode untuk melakukan aksi bagi setiap item menu munculan.

Membuat file sumber daya XML

Buat file dan direktori sumber daya menu XML dengan mengikuti langkah-langkah di bagian sebelumnya. Gunakan nama yang sesuai untuk file tersebut, seperti `menu_popup` .

Menambahkan ImageButton untuk ikon yang akan diklik

Gunakan ImageButton dalam layout aktivitas untuk ikon yang memicu menu munculan. Menu munculan ditambatkan ke tampilan dalam aktivitas, sebagai ImageButton. Pengguna mengekliknya untuk melihat menu.

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/button_popup"
    android:src="@drawable/@drawable/ic_action_popup"/>
```

Menetapkan onClickListener ke tombol

1. Buat variabel anggota (`mButton`) dalam definisi kelas aktivitas:

   ```
   public class MainActivity extends AppCompatActivity {
      private ImageButton mButton;
      ...
   }
   ```

2. In the `onCreate()` method for the same activity, assign the `ImageButton` in the layout to the member variable, and assign onClickListener() to the button:

   ```
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      ...
      mButton = (ImageButton) findViewById(R.id.button_popup);
      mButton.setOnClickListener(new View.OnClickListener() {
      ...
      // define onClick here
      ...
   });
   ```

Memekarkan menu munculan

Sebagai bagian dari metode `setOnClickListener()` dalam `onCreate()`, tambahkan metode `onClick()` untuk memekarkan menu munculan dan mendaftarkannya dengan PopupMenu.OnMenuItemClickListener:

```
@Override
public void onClick(View v) {
   //Creating the instance of PopupMenu
   PopupMenu popup = new PopupMenu(MainActivity.this, mButton);
   //Inflating the Popup using xml file
   popup.getMenuInflater().inflate(R.menu.menu_popup, popup.getMenu());
   //registering popup with OnMenuItemClickListener
   popup.setOnMenuItemClickListener(new
                              PopupMenu.OnMenuItemClickListener() {
      ...
      // Add onMenuItemClick here
      ...
            // Perform action here
      ...
}
```

Setelah membuat instance objek PopupMenu (`popup` dalam contoh di atas), metode menggunakan kelas MenuInflater dan metode inflate()-nya, yang memerlukan sebagai parameter:

- `id` sumber daya untuk sumber daya layout XML yang akan dimuat (`menu_popup` dalam contoh di atas)
- Menu tempat memekarkan menjadi (`popup.getMenu()` dalam contoh di atas).

Kode kemudian mendaftarkan munculan dengan listener, PopupMenu.OnMenuItemClickListener.

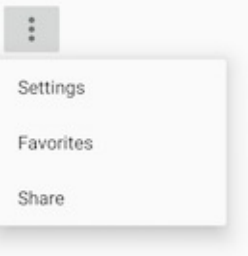
Mengimplementasikan onMenuItemClick

Untuk melakukan aksi bila pengguna memilih item menu munculan, implementasikan callback onMenuItemClick() dalam metode `setOnClickListener()` di atas, dan selesaikan metode dengan `popup.show` untuk menampilkan menu munculan:

```
         public boolean onMenuItemClick(MenuItem item) {
            // Perform action here
            return true;
         }
      });
      popup.show();  //show the popup menu
   }
});// close the setOnClickListener method
```

Sebagai rangkuman, keseluruhan metode `onCreate()` kini seharusnya tampak seperti ini:

```
private ImageButton mButton;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   ...
   // popup button setup
   mButton = (ImageButton) findViewById(R.id.button_popup);
   mButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
         //Creating the instance of PopupMenu
         PopupMenu popup = new PopupMenu(MainActivity.this, mButton);
         //Inflating the Popup using xml file
         popup.getMenuInflater().inflate(R.menu.menu_popup, popup.getMenu());

         //registering popup with OnMenuItemClickListener
         popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
            public boolean onMenuItemClick(MenuItem item) {
               // Perform action here
               return true;
            }
         });
         popup.show();//show the popup menu
      }
   });//close the setOnClickListener method
 }
```



Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Menggunakan Menu Opsi

Ketahui selengkapnya

- Panduan Android API, bagian "Kembangkan":
  - Menambahkan Bilah Aplikasi
  - Gaya dan Tema
  - Menu
  - Sumber Daya Menu
- Lainnya:
  - Panduan Android API, bagian "Desain":Ikon dan sumber daya yang bisa diunduh lainnya
  - Panduan Pengguna Android Studio:Image Asset Studio
  - Blog Developer Android: "Holo Everywhere"

4.3: Navigasi Layar

Materi:

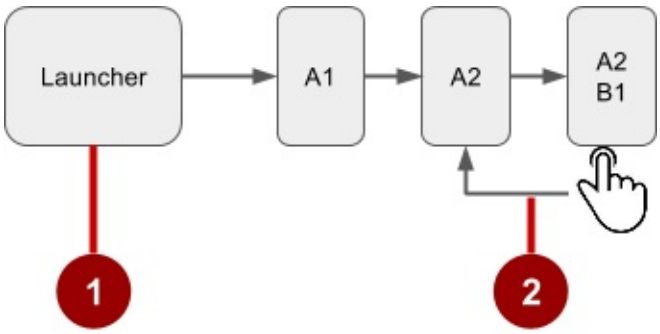
Menyediakan jalur bagi pengguna melalui aplikasi Anda

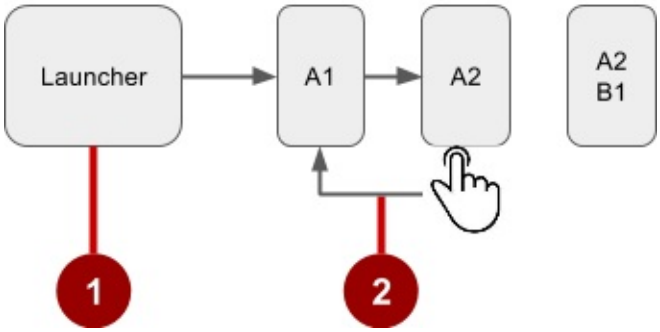
Pada tahap awal pengembangan aplikasi, Anda harus menentukan jalur yang harus digunakan pengguna melalui aplikasi Anda untuk melakukan sesuatu, seperti menyerahkan pesanan atau menjelajahi materi. Setiap jalur memungkinkan pengguna melakukan navigasi ke seluruh bagian, ke dalam, dan kembali dari tugas serta potongan materi yang berbeda-beda dalam aplikasi.

Dalam banyak kasus, Anda akan memerlukan sejumlah jalur berbeda melalui aplikasi Anda yang menawarkan tipe navigasi berikut:

- Navigasi *kembali*: Pengguna bisa mengarahkan kembali ke layar sebelumnya menggunakan tombol Kembali.
- Navigasi *hierarkis*: Pengguna bisa mengarahkan melalui hierarki layar yang disusun dengan layar *induk* untuk setiap rangkaian layar *anak*.

Navigasi tombol-kembali





Dalam gambar di atas:

1. Mulai dari Launcher.
2. Mengeklik tombol Kembali untuk mengarahkan ke layar sebelumnya.

Anda tidak harus mengelola tombol Kembali di aplikasi. Sistem akan menangani tugas dan *back-stack*—yakni daftar layar sebelumnya—secara otomatis. Tombol Kembali secara default cukup melintasi daftar layar ini, yang membuang layar saat ini dari daftar saat pengguna menekannya.

Akan tetapi, ada beberapa kasus di mana Anda ingin menggantikan perilaku tombol Kembali. Misalnya, jika layar Anda berisi browser web yang disematkan, tempat pengguna bisa berinteraksi dengan elemen laman untuk mengarahkan di antara laman web, Anda mungkin ingin memicu perilaku kembali default dari browser yang disematkan saat pengguna menekan tombol Kembali di perangkat.

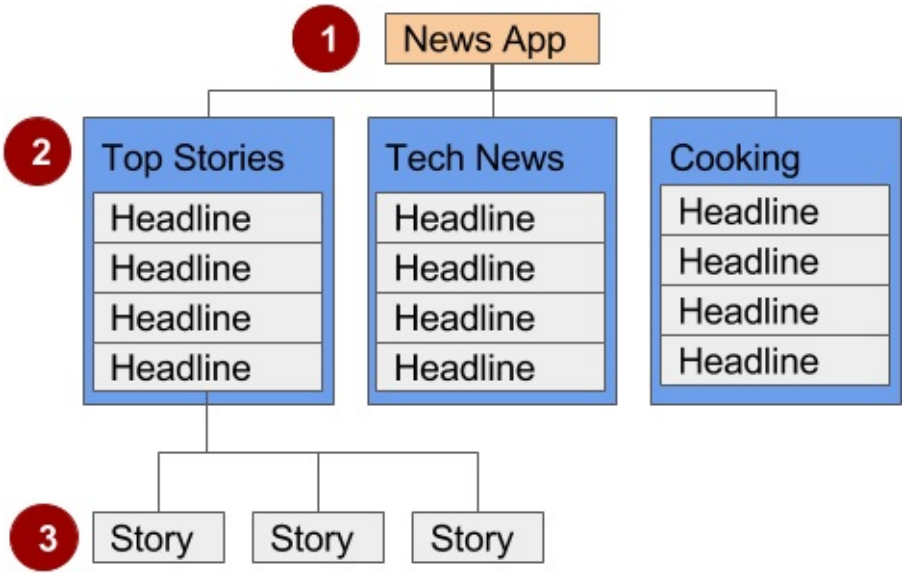
Metode onBackPressed() dari kelas Activity dipanggil bila aktivitas mendeteksi bahwa pengguna menekan tombol Kembali. Implementasi default hanya mengakhiri aktivitas saat ini, namun Anda bisa menggantinya agar melakukan sesuatu:

```
@Override
public void onBackPressed() {
    // Add the Back key handler here.
    return;
}
```

Jika kode Anda memicu browser yang disematkan dengan perilakunya sendiri ke tombol Kembali, Anda harus mengembalikan perilaku tombol Kembali ke perilaku default sistem, jika pengguna menggunakan tombol Kembali untuk melebihi awal riwayat internal browser.

Pola navigasi hierarkis

Untuk memberi jalur bagi pengguna ke beragam layar aplikasi, praktik terbaiknya adalah menggunakan beberapa bentuk navigasi hierarkis. Layar aplikasi umumnya disusun dalam hierarki induk-anak, seperti yang ditampilkan dalam gambar di bawah ini:



Dalam gambar di atas:

1. **Layar induk**. Layar induk (seperti layar utama aplikasi berita) memungkinkan navigasi turun ke layar *anak*.
   - Aktivitas utama suatu aplikasi biasanya adalah layar induk.
   - Implementasikan layar induk sebagai Activity dengan navigasi *turunan* ke satu atau beberapa layar anak.
2. **Saudara layar anak level pertama**. Saudara adalah beberapa layar dalam posisi yang sama dalam hierarki dengan layar induk yang sama (seperti saudara kandung).
   - Pada saudara level pertama, layar anak bisa berupa *kumpulan* layar yang mengumpulkan judul berita, seperti yang ditampilkan di atas.
   - Implementasikan setiap layar anak sebagai Activity atau Fragment.
   - Implementasikan navigasi *lateral* untuk mengarahkan dari satu saudara ke saudara lain pada level yang sama.
   - Jika ada layar level kedua, layar anak level pertama akan menjadi *induk* bagi saudara layar anak level kedua. Implementasikan navigasi *turunan* untuk layar anak level kedua.
3. **Saudara layar anak level kedua**. Dalam aplikasi berita dan aplikasi lainnya yang menawarkan beberapa level informasi, saudara layar anak level kedua mungkin menawarkan materi, misalnya cerita.
   - Implementasikan saudara layar anak level kedua sebagai Activity atau Fragment lain.
   - Cerita pada level ini bisa meliputi elemen cerita yang disematkan seperti video, peta, dan komentar, yang mungkin diimplementasikan sebagai fragmen.

Anda bisa memungkinkan pengguna untuk mengarahkan naik dan turun dari suatu induk, dan menyamping di antara layar saudara:

- Navigasi *turunan*: Mengarahkan turun dari suatu layar induk ke layar anak.
- Navigasi *leluhur*: Mengarahkan naik dari suatu layar anak ke layar induk.
- Navigasi *lateral*: Mengarahkan dari satu saudara ke saudara yang lain (pada level yang sama).

Anda bisa menggunakan aktivitas utama (sebagai layar induk) kemudian aktivitas atau fragmen lainnya untuk mengimplementasikan hierarki layar dalam aplikasi.

Aktivitas utama dengan aktivitas lainnya

Jika saudara layar anak level pertama memiliki layar anak level lain di bawahnya, Anda harus mengimplementasikan layar level pertama sebagai aktivitas, sehingga daur hidupnya dikelola dengan benar sebelum memanggil layar anak level kedua.

Misalnya, dalam gambar di atas, layar induk kemungkinan besar adalah aktivitas utama. Aktivitas utama aplikasi (biasanya MainActivity.java) umumnya adalah layar induk untuk semua layar lain dalam aplikasi, dan Anda akan mengimplementasikan pola navigasi dalam aktivitas utama untuk memungkinkan pengguna berpindah ke aktivitas atau fragmen lain. Misalnya, Anda bisa mengimplementasikan navigasi menggunakan suatu Maksud yang akan memulai aktivitas.

Tip: Penggunaan suatu Maksud dalam aktivitas saat ini untuk memulai aktivitas lain akan menambahkan aktivitas saat ini ke tumpukan panggilan, sehingga tombol **Kembali** di aktivitas lain (yang dijelaskan di bagian sebelumnya) akan mengembalikan pengguna ke aktivitas saat ini.

Sebagaimana yang Anda pelajari sebelumnya, sistem Android memulai kode dalam suatu instance Aktivitas dengan metode callback yang mengelola daur hidup aktivitas untuk Anda. (Pelajaran sebelumnya membahas daur hidup aktivitas; untuk informasi selengkapnya, lihat "Mengelola Daur Hidup Aktivitas" di bagian Pelatihan pada Panduan Developer Android.)

Hierarki aktivitas induk dan anak didefinisikan dalam file AndroidManifest.xml. Misalnya, yang berikut ini mendefinisikan `OrderActivity` sebagai anak dari `MainActivity` induk:

```
<activity android:name=".OrderActivity"
   android:label="@string/title_activity_order"
   android:parentActivityName=
                     "com.example.android.droidcafe.MainActivity">
   <meta-data
      android:name="android.support.PARENT_ACTIVITY"
      android:value=".MainActivity"/>
</activity>
```

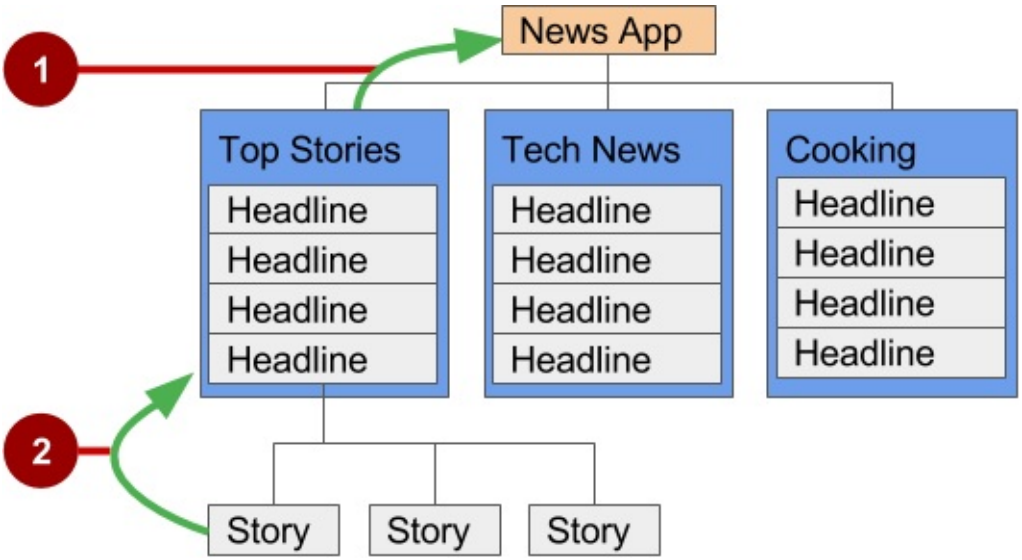
Aktivitas utama dengan fragmen

Jika saudara layar anak *tidak* memiliki layar anak level lain di bawahnya, Anda bisa mengimplementasikannya sebagai fragmen. Fragmen menyatakan perilaku atau porsi antarmuka pengguna dalam suatu aktivitas. Anda bisa menganggap fragmen sebagai bagian modular dari aktivitas, yang memiliki daur hidup sendiri, menerima kejadian masukan sendiri, dan yang bisa Anda tambahkan atau buang saat aktivitas berjalan.

Anda bisa mengombinasikan beberapa fragmen dalam satu aktivitas tunggal. Misalnya, dalam layar saudara bagian yang menampilkan cerita dan diimplementasikan sebagai suatu aktivitas, Anda mungkin memiliki layar anak untuk klip video yang diimplementasikan sebagai fragmen. Anda harus mengimplementasikan suatu cara bagi pengguna untuk mengarah ke fragmen klip video, kemudian kembali ke aktivitas yang menampilkan cerita.

Navigasi leluhur (tombol Naik)

Dengan navigasi leluhur dalam hierarki multitier, Anda memungkinkan pengguna untuk naik *ke atas* dari bagian saudara ke saudara kumpulan, kemudian *naik* ke layar induk.



Dalam gambar di atas:

1. Tombol **Naik** untuk navigasi leluhur dari saudara level pertama ke induk.
2. Tombol **Naik** untuk navigasi leluhur dari saudara level kedua ke layar anak level pertama yang bertindak sebagai layar induk.

Tombol **Naik** digunakan untuk berpindah dalam aplikasi berdasarkan hubungan hierarki antar layar. Misalnya (denganmerujuk pada gambar di atas):

- Jika layar anak level pertama menawarkan judul untuk berpindah ke layar anak level kedua, saudara layar anak level kedua harus menawarkan tombol **Naik** yang akan mengembalikan ke layar anak level pertama, yang merupakan *induk* mereka bersama.
- Jika layar induk menawarkan navigasi ke saudara anak level pertama, maka saudara anak level pertama harus menawarkan tombol **Naik** yang mengembalikan ke layar induk.
- Jika layar induk adalah layar paling atas dalam aplikasi (yaitu, layar utama aplikasi), layar tersebut tidak boleh menawarkan tombol **Naik**.

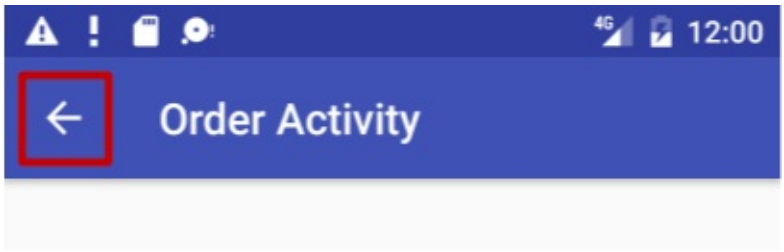
Tip: Tombol **Kembali** di bawah layar berbeda dari tombol **Naik**. Tombol **Kembali** menyediakan navigasi ke layar mana pun yang Anda tampilkan sebelumnya. Jika Anda memiliki sejumlah layar anak yang bisa disusuri oleh pengguna menggunakan pola navigasi lateral (seperti yang dijelaskan nanti di bab ini), tombol **Kembali** akan mengirim pengguna kembali ke layar anak sebelumnya, bukan ke layar induk. Gunakan tombol **Naik** jika Anda ingin menyediakan navigasi leluhur dari layar anak kembali ke layar induk. Untuk informasi selengkapnya tentang navigasi Naik, lihat Menyediakan Navigasi Naik.

Lihat bab konsep "Menu" untuk detail mengenai cara mengimplementasikan bilah aplikasi. Untuk menyediakan tombol **Naik** bagi aktivitas layar anak, deklarasikan induk aktivitas sebagai MainActivity dalam file AndroidManifest.xml. Anda juga bisa menyetel `android:label` ke judul layar aktivitas, seperti "Order Activity" (yang diekstrak ke dalam sumber daya string `title_activity_order` dalam kode di bawah). Ikuti langkah-langkah ini untuk mendeklarasikan induk dalam AndroidManifest.xml:

1. Buka **AndroidManifest.xml**
2. Ubah elemen aktivitas untuk aktivitas layar anak (dalam contoh ini, `OrderActivity`) ke yang berikut ini:

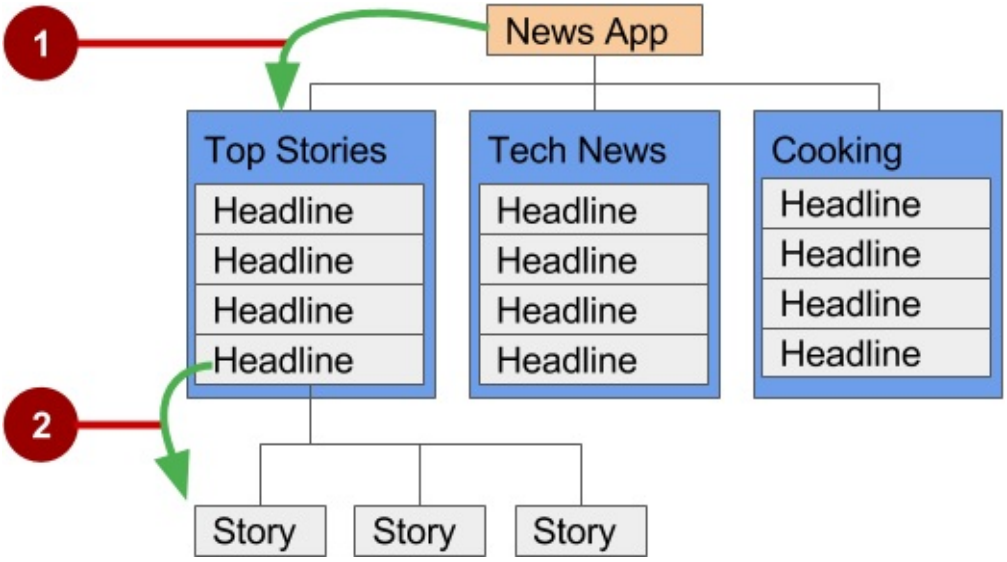
```
 <activity android:name=".OrderActivity"
   android:label="@string/title_activity_order"
   android:parentActivityName=
           "com.example.android.optionsmenuorderactivity.MainActivity">
   <meta-data
      android:name="android.support.PARENT_ACTIVITY"
      android:value=".MainActivity"/>
</activity>
```

Layar anak ("Order Activity") kini menyertakan tombol **Naik** dalam bilah aplikasi (disorot dalam gambar di bawah ini), yang bisa diketuk oleh pengguna untuk mengarahkan kembali ke layar induk.



Navigasi turunan

Dengan navigasi turunan, Anda memungkinkan pengguna untuk pergi dari layar induk ke layar anak level pertama, dan dari layar anak level pertama turun ke layar anak level kedua.



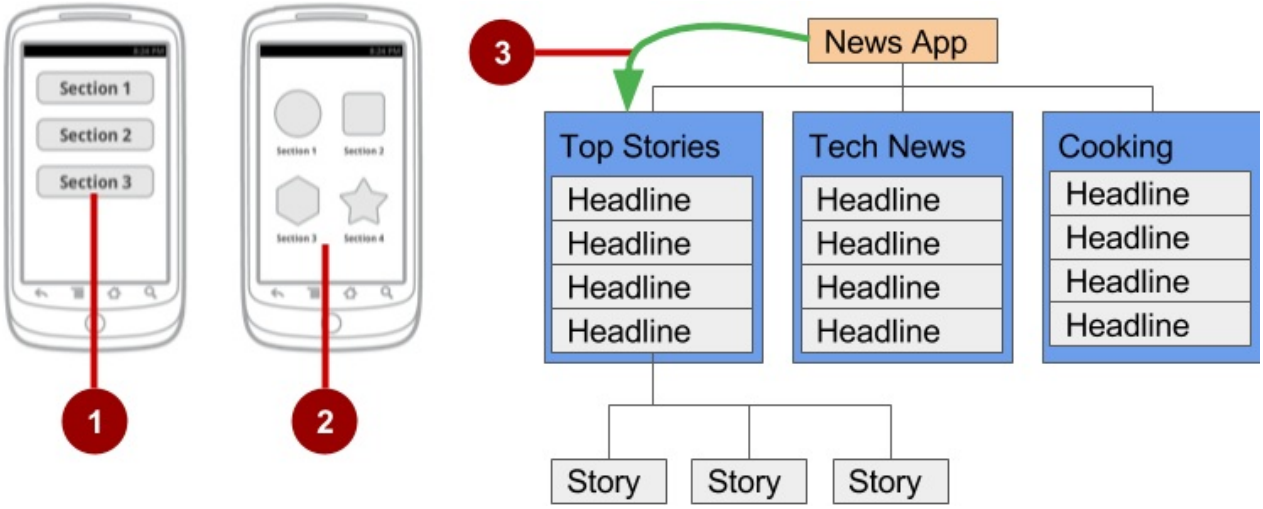
Dalam gambar di atas:

1. Navigasi turunan dari orang tua ke layar anak level pertama.
2. Navigasi turunan dari judul pada layar anak level pertama ke layar anak level kedua.

Tombol atau target

Praktik terbaik untuk navigasi turunan dari layar induk ke saudara kumpulan adalah menggunakan tombol atau *target* sederhana seperti susunan gambar atau tombol ikonis (disebut juga dengan *dasbor*). Bila pengguna menyentuh tombol, layar saudara kumpulan akan membuka, yang akan mengganti (layar) konteks saat ini seluruhnya.

Tip: Tombol dan target sederhana jarang sekali digunakan untuk berpindah ke saudara bagian *dalam* suatu kumpulan. Lihat daftar, menu korsel, dan kartu di bagian berikutnya.



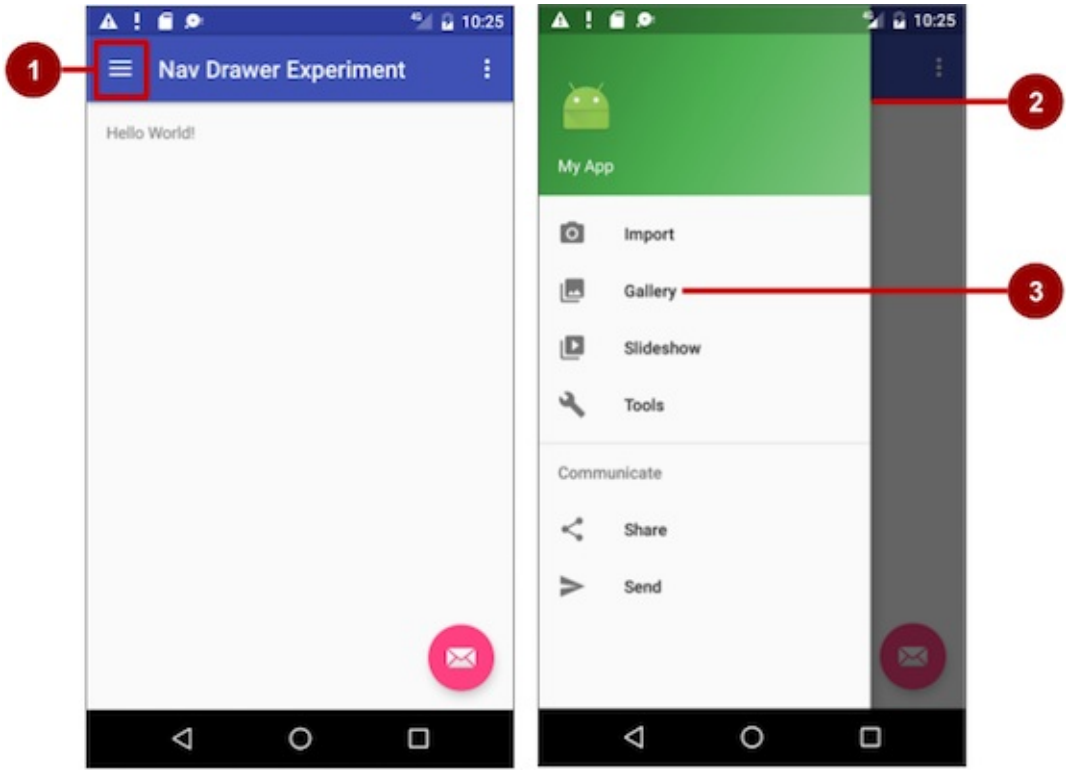
Dalam gambar di atas:

1. Tombol pada layar induk.
2. Target (Tombol gambar atau ikon) pada layar induk.
3. Pola navigasi turunan dari layar induk ke saudara anak level pertama.

Dasbor biasanya memiliki dua atau tiga baris dan kolom, dengan target sentuh luas untuk membuatnya mudah digunakan. Dasbor adalah yang paling tepat bila setiap saudara kumpulan sama pentingnya. Anda bisa menggunakan LinearLayout, RelativeLayout, atau GridLayout. Lihat Layout untuk ringkasan cara kerja layout.

Panel samping navigasi

Panel samping navigasi adalah panel yang biasanya menampilkan opsi navigasi pada tepi kiri layar, seperti yang ditampilkan pada sisi kanan gambar di bawah. Panel ini paling sering tersembunyi, namun ditampilkan bila pengguna menggesekkan jari dari tepi kiri layar atau menyentuh ikon navigasi dalam bilah aplikasi, seperti yang ditampilkan pada samping kiri gambar di bawah ini.



Dalam gambar di atas:

1. Ikon navigasi dalam bilah aplikasi
2. Panel samping navigasi
3. Item menu panel samping navigasi

Contoh bagus panel samping navigasi adalah aplikasi Gmail, yang menyediakan akses ke Inbox, folder email berlabel, dan setelan. Praktik terbaik untuk menggunakan panel samping navigasi adalah menyediakan navigasi turunan dari aktivitas induk ke semua aktivitas atau fragmen lain dalam aplikasi. Panel samping navigasi bisa menampilkan banyak target navigasi sekaligus—misalnya, bisa berisi tombol (seperti dasbor), tab, atau daftar item (panel samping Gmail).

Untuk membuat panel samping navigasi dalam aplikasi, Anda perlu melakukan yang berikut ini:

1. Buat layout berikut:
    - Panel samping navigasi sebagai tampilan akar layout aktivitas.
    - Tampilan navigasi untuk panel samping itu sendiri.
    - Layout bilah aplikasi yang akan menyertakan tombol ikon navigasi.
    - Layout materi untuk aktivitas yang menampilkan panel samping navigasi.
    - Layout untuk header panel samping navigasi.
2. Isilah menu panel samping navigasi dengan judul item dan ikon.
3. Persiapkan panel samping navigasi dan listener item dalam kode aktivitas.
4. Tangani pemilihan item menu navigasi.

Membuat layout panel samping navigasi

Untuk membuat layout panel samping navigasi, gunakan DrawerLayout API yang tersedia dalam Pustaka Dukungan. Untuk spesifikasi desain, ikuti prinsip desain panel samping navigasi dalam panduan desain Panel Samping Navigasi.

Untuk menambahkan panel samping navigasi, gunakan `DrawerLayout` sebagai tampilan akar layout aktivitas Anda. Di dalam `DrawerLayout`, tambahkan satu tampilan yang berisi materi utama untuk layar (layout utama Anda saat panel samping disembunyikan) dan tampilan lain, umumnya NavigationView, yang berisi materi panel samping navigasi.

Tip: Untuk membuat layout Anda lebih mudah dipahami, gunakan tag `include` untuk menyertakan layout XML dalam layout XML lainnya.

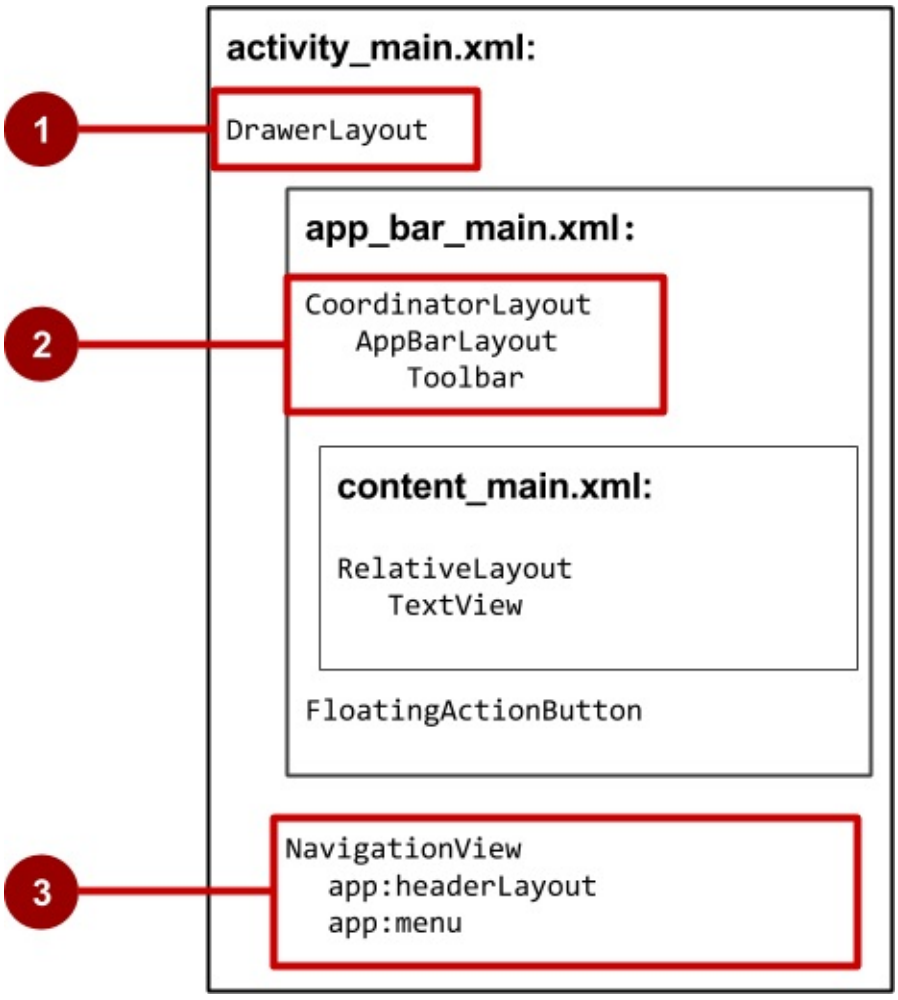
Misalnya, layout berikut menggunakan:

- Sebuah `DrawerLayout` sebagai akar layout aktivitas dalam **activity_main.xml**.
- Materi utama layar didefinisikan dalam file layout **app_bar_main.xml**.
- NavigationView yang menyatakan menu navigasi standar yang bisa diisi oleh file XML sumber daya menu.

Lihat gambar di bawah yang menyangkut layout ini:

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<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:openDrawer="start">

 <include
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

 <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:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />

</android.support.v4.widget.DrawerLayout>
```



Dalam gambar di atas:

1. DrawerLayout adalah tampilan akar layout aktivitas.
2. `app_bar_main` (**app_bar_main.xml**) yang disertakan menggunakan CoordinatorLayout sebagai akarnya, dan mendefinisikan layout bilah aplikasi dengan Bilah Alat yang akan menyertakan ikon navigasi untuk membuka panel samping.

3. NavigationView mendefinisikan layout panel samping navigasi dan header-nya, serta menambahkan item menu ke panel.

Perhatikan hal berikut dalam layout **activity_main.xml**:

- Identitas `android:id` untuk tampilan DrawerLayout adalah `drawer_layout`. Anda akan menggunakan ID ini untuk membuat instance objek `drawer` dalam kode Anda.
- Identitas `android:id` untuk `NavigationView` adalah `nav_view`. Anda akan menggunakan ID ini untuk membuat instance objek `navigationView` dalam kode Anda.
- Objek `NavigationView` harus menetapkan gravitasi horizontalnya dengan atribut `android:layout_gravity`. Gunakan nilai `"start"` untuk atribut ini (bukan `"left"`) sehingga jika aplikasi ini digunakan dengan bahasa kanan ke kiri (RTF), panel samping akan muncul di sisi kanan, buka di kiri.

  ```
      android:layout_gravity="start"
  ```

- Gunakan atribut `android:fitsSystemWindows="true"` untuk menyetel pengisi `DrawerLayout` dan `NavigationView` untuk memastikan materi tidak menghamparkan jendela sistem. `DrawerLayout` menggunakan `fitsSystemWindows` sebagai tanda bahwa anaknya (seperti tampilan materi utama) perlu disisipkan, namun tetap menggambar latar belakang bilah status atas di ruang itu. Akibatnya, panel samping navigasi tampak tumpang tindih, namun tidak mengaburkan, bilah status atas yang tembus pandang. Sisipan yang Anda dapatkan dari `fitsSystemWindows` akan dikoreksi pada semua versi platform untuk memastikan materi Anda tidak tumpang tindih dengan komponen UI yang disediakan sistem.

Header panel samping navigasi

Objek `NavigationView` menetapkan layout untuk *header* panel samping navigasi dengan atribut `app:headerLayout="@layout/nav_header_main"`. File **nav_header_main.xml** mendefinisikan layout header ini untuk menyertakan `ImageView` dan `TextView`, yang umum untuk panel samping navigasi, namun Anda juga bisa menyertakan Tampilan lain.

Tip: Tinggi header harus 160 dp, yang harus Anda ekstrak menjadi sumber daya dimensi (`nav_header_height`).

nav_header_main.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="@dimen/nav_header_height"
    android:background="@drawable/side_nav_bar"
    android:gravity="bottom"
    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"
    android:theme="@style/ThemeOverlay.AppCompat.Dark">

 <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="@dimen/nav_header_vertical_spacing"
        android:src="@android:drawable/sym_def_app_icon" />

 <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="@dimen/nav_header_vertical_spacing"
        android:text="@string/my_app_title"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1" />

</LinearLayout>
```

Layout bilah aplikasi

Tag `include` dalam layout `activity_main` menyertakan layout `app_bar_main` , yang menggunakan CoordinatorLayout sebagai akarnya. File layout **app_bar_main.xml** mendefinisikan layout bilah aplikasi dengan kelas Toolbar seperti yang ditampilkan sebelumnya dalam bab tentang menu. Layout ini juga mendefinisikan tombol aksi mengambang, dan menggunakan tag `include` untuk menyertakan `content_main` layout (**content_main.xml**):

app_bar_main.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"
    android:fitsSystemWindows="true"
    tools:context="com.example.android.navigationexperiments.MainActivity">

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

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

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

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

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

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

Perhatikan yang berikut ini:

- Layout `app_bar_main` menggunakan CoordinatorLayout sebagai akarnya, dan menyertakan layout `content_main` .
- Layout `app_bar_main` menggunakan atribut `android:fitsSystemWindows="true"` untuk menyetel pengisi bilah aplikasi guna memastikannya tidak menghamparkan jendela sistem seperti bilah status.

Layout materi untuk layar aktivitas utama

Layout di atas menggunakan tag `include` untuk menyertakan layout `content_main` , yang mendefinisikan layout layar aktivitas utama (**content_main.xml**). Dalam contoh layout di bawah ini, layar aktivitas utama menampilkan TextView yang menampilkan string "Hello World!":

content_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.example.android.navigationexperiments.MainActivity"
    tools:showIn="@layout/app_bar_main">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</RelativeLayout>
```

Perhatikan yang berikut ini:

- Layout `content_main` harus anak pertama dalam `DrawerLayout` karena panel samping harus berada di atas materi. Dalam layout kami di atas, layout `content_main` disertakan dalam layout `app_bar_main` , yang merupakan anak pertama.
- Layout `content_main` menggunakan sebuah grup tampilan `RelativeLayout` yang disetel untuk menyesuaikan dengan lebar dan panjang tampilan induk, karena layout ini menyatakan keseluruhan UI saat panel samping navigasi disembunyikan.
- *Perilaku* layout untuk `RelativeLayout` disetel ke sumber daya string `@string/appbar_scrolling_view_behavior` , yang mengontrol perilaku gulir layar sehubungan dengan bilah aplikasi di bagian atas. Perilaku ini didefinisikan oleh kelas AppBarLayout.ScrollingViewBehavior. Perilaku ini harus digunakan oleh Tampilan yang bisa menggulir secara vertikal—perilaku ini mendukung pengguliran tersarang untuk menggulir AppBarLayout yang seinduk secara otomatis.

Mengisi menu panel samping navigasi

Objek `NavigationView` di layout `activity_main` menetapkan item menu untuk panel samping navigasi dengan menggunakan pernyataan berikut:

```
app:menu="@menu/activity_main_drawer"
```

Item menu didefinisikan dalam file `activity_main_drawer.xml` , yang berlokasi di **app > res > menu**. Tag `<group></group>` mendefinisikan *grup menu*—kumpulan item yang memiliki ciri sama, seperti apakah mereka terlihat, aktif, atau bisa dicentang. Suatu grup harus berisi satu atau beberapa elemen `<item></>` dan merupakan anak dari elemen `<menu>` , seperti yang ditampilkan di bawah ini. Selain mendefinisikan setiap judul item menu dengan atribut `android:title` , file juga mendefinisikan setiap ikon item menu dengan atribut `android:icon` .

Grup didefinisikan dengan atribut `android:checkableBehavior` . Atribut ini memungkinkan Anda meletakkan elemen interaktif dalam panel samping navigasi, seperti switch beralih yang bisa diaktifkan atau dinonaktifkan, dan kotak centang serta tombol radio yang bisa dipilih. Pilihan untuk atribut ini adalah:

- `single` : Hanya satu item dari grup ini yang bisa dicentang. Gunakan untuk tombol radio.
- `all` : Semua item bisa dicentang. Gunakan untuk kotak centang.
- `none` : Tidak ada item yang bisa dicentang.

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

    <group android:checkableBehavior="none">
        <item
            android:id="@+id/nav_camera"
            android:icon="@drawable/ic_menu_camera"
            android:title="@string/import_camera" />
        <item
            android:id="@+id/nav_gallery"
            android:icon="@drawable/ic_menu_gallery"
            android:title="@string/gallery" />
        <item
            android:id="@+id/nav_slideshow"
            android:icon="@drawable/ic_menu_slideshow"
            android:title="@string/slideshow" />
        <item
            android:id="@+id/nav_manage"
            android:icon="@drawable/ic_menu_manage"
            android:title="@string/tools" />
    </group>

    <item android:title="@string/communicate">
        <menu>
            <item
                android:id="@+id/nav_share"
                android:icon="@drawable/ic_menu_share"
                android:title="@string/share" />
            <item
                android:id="@+id/nav_send"
                android:icon="@drawable/ic_menu_send"
                android:title="@string/send" />
        </menu>
    </item>

</menu>
```

Mempersiapkan panel samping navigasi dan listener item

Untuk menggunakan listener item menu panel samping navigasi, aktivitas yang menjadi host panel samping navigasi harus mengimplementasikan antarmuka OnNavigationItemSelectedListener:

1. Implementasikan `NavigationView.OnNavigationItemSelectedListener` dalam definisi kelas:

   ```
   public class MainActivity extends AppCompatActivity
                implements NavigationView.OnNavigationItemSelectedListener {
   ...
   }
   ```

   Antarmuka ini menawarkan metode onNavigationItemSelected(), yang dipanggil bila suatu item dalam item menu panel samping navigasi diketuk. Saat Anda memasukkan `OnNavigationItemSelectedListener`, bola lampu peringatan merah muncul pada margin kiri.

2. Klik bola lampu peringatan merah, pilih **Implement methods**, dan pilih metode **onNavigationItemSelected(item:MenuItem):boolean**.

   Android Studio akan menambahkan stub untuk metode ini:

   ```
   @Override
       public boolean onNavigationItemSelected(MenuItem item) {
           return false;
       }
   }
   ```

   Anda bisa mempelajari cara menggunakan stub ini di bagian berikutnya.

3. Sebelum mempersiapkan listener item navigasi, tambahkan kode ke metode `onCreate()` aktivitas untuk membuat instance objek DrawerLayout dan NavigationView (`drawer` dan `navigationView` dalam kode di bawah ini):

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   . . .
   DrawerLayout drawer = (DrawerLayout)
                              findViewById(R.id.drawer_layout);
   ActionBarDrawerToggle toggle =
               new ActionBarDrawerToggle(this, drawer, toolbar,
               R.string.navigation_drawer_open,
               R.string.navigation_drawer_close);
   if (drawer != null) {
      drawer.addDrawerListener(toggle);
   }
   toggle.syncState();

   NavigationView navigationView = (NavigationView)
               findViewById(R.id.nav_view);
   if (navigationView != null) {
      navigationView.setNavigationItemSelectedListener(this);
   }
}
```

   Kode di atas membuat instance ActionBarDrawerToggle, yang menggantikan sumber daya dapat digambar khusus untuk tombol **Naik** aktivitas dalam bilah aplikasi, dan tautkan aktivitas ke DrawerLayout. Sumber daya dapat digambar yang khusus akan muncul sebagai ikon navigasi "hamburger" jika panel samping ditutup, dan beranimasi menjadi tanda panah jika panel samping terbuka.

   Catatan: Pastikan menggunakan ActionBarDrawerToggle dalam support-library-v7.appcompact, bukan versi dalam support-library-v4.
   Tip: Anda bisa menyesuaikan peralihan beranimasi dengan mendefinisikan drawerArrowStyle dalam tema ActionBar (untuk informasi lebih detail tentang tema ActionBar, lihat Menambahkan Bilah Aplikasi dalam dokumentasi Developer Android.

   Kode di atas mengimplementasikan addDrawerListener() untuk mendengarkan kejadian panel samping terbuka dan tertutup, jadi saat pengguna mengetuk tombol dapat digambar yang khusus, panel samping navigasi akan bergeser keluar.

   Anda juga harus menggunakan metode syncState() dari ActionBarDrawerToggle untuk menyinkronkan keadaan indikator panel samping. Sinkronisasi harus terjadi setelah keadaan instance DrawerLayout dipulihkan, dan di waktu lain bila keadaan berubah sedemikian rupa sehingga ActionBarDrawerToggle tidak diberi tahu.

   Kode di atas diakhiri dengan menyetel listener, `setNavigationItemSelectedListener()` , untuk panel samping navigasi agar mendengar klik item.

4. ActionBarDrawerToggle juga memungkinkan Anda menetapkan string yang digunakan untuk menjelaskan tindakan buka/tutup panel samping untuk layanan aksesibilitas. Definisikan string berikut ini dalam file string.xml Anda:

```
<string name="navigation_drawer_open">Open navigation drawer</string>
<string name="navigation_drawer_close">Close navigation drawer</string>
```

Menangani pemilihan item menu navigasi

Tulis kode dalam stub metode onNavigationItemSelected() untuk menangani pemilihan item menu. Metode ini dipanggil bila item dalam menu panel samping navigasi diketuk.

Metode ini menggunakan pernyataan `if` untuk melakukan aksi yang sesuai berdasarkan `id` item menu, yang bisa Anda ambil dengan menggunakan metode getItemId():

```
@Override
public boolean onNavigationItemSelected(MenuItem item) {
   DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
   // Handle navigation view item clicks here.
   switch (item.getItemId()) {
      case R.id.nav_camera:
         // Handle the camera import action (for now display a toast).
         drawer.closeDrawer(GravityCompat.START);
         displayToast(getString(R.string.chose_camera));
         return true;
      case R.id.nav_gallery:
         // Handle the gallery action (for now display a toast).
         drawer.closeDrawer(GravityCompat.START);
         displayToast(getString(R.string.chose_gallery));
         return true;
      case R.id.nav_slideshow:
         // Handle the slideshow action (for now display a toast).
         drawer.closeDrawer(GravityCompat.START);
         displayToast(getString(R.string.chose_slideshow));
         return true;
      case R.id.nav_manage:
         // Handle the tools action (for now display a toast).
         drawer.closeDrawer(GravityCompat.START);
         displayToast(getString(R.string.chose_tools));
         return true;
      case R.id.nav_share:
         // Handle the share action (for now display a toast).
         drawer.closeDrawer(GravityCompat.START);
         displayToast(getString(R.string.chose_share));
         return true;
      case R.id.nav_send:
         // Handle the send action (for now display a toast).
         drawer.closeDrawer(GravityCompat.START);
         displayToast(getString(R.string.chose_send));
         return true;
      default:
         return false;
   }
}
```

Setelah pengguna mengetuk pilihan di panel samping navigasi atau mengetuk di luar panel samping, metode closeDrawer() DrawerLayout akan menutup panel samping.

Daftar dan korsel

Gunakan daftar gulir, seperti RecyclerView, untuk menyediakan target navigasi bagi navigasi turunan. Daftar yang bergulir secara vertikal sering kali digunakan untuk layar yang mencantumkan cerita, dengan setiap item daftar bertindak sebagai tombol bagi setiap cerita. Untuk item materi yang lebih bersifat visual atau kaya media seperti foto atau video, Anda bisa menggunakan daftar yang bergulir secara horizontal (disebut juga dengan *menu korsel*). Elemen UI ini bagus untuk menyajikan item dalam suatu kumpulan (misalnya, daftar berita).

Anda akan mempelajari semua tentang RecyclerView di bab berikutnya.

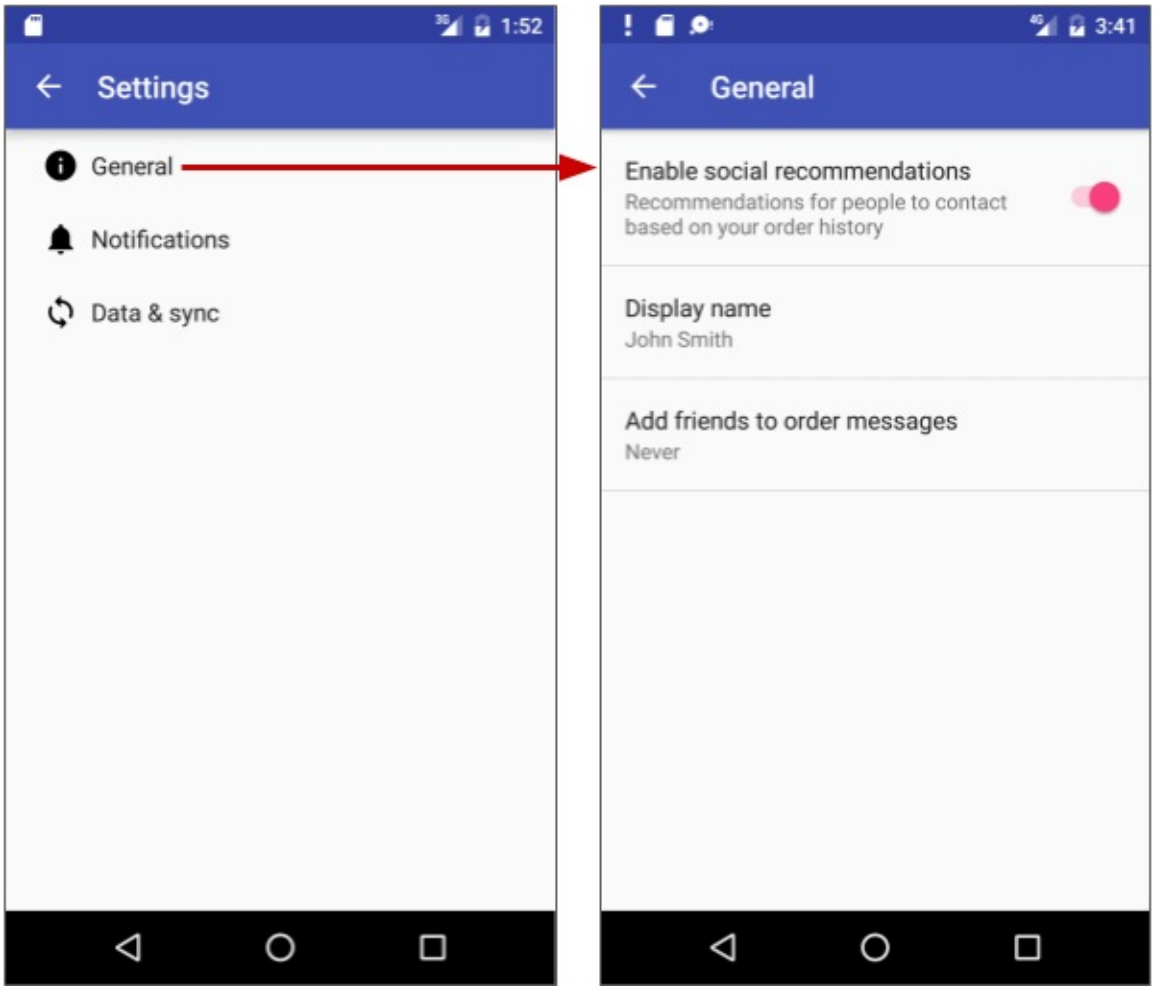
Alur navigasi master/detail

Dalam alur navigasi master/detail, layar master berisi daftar item, dan layar detail menampilkan informasi detail tentang item tertentu. Navigasi turunan biasanya diimplementasikan oleh salah satu hal berikut ini:

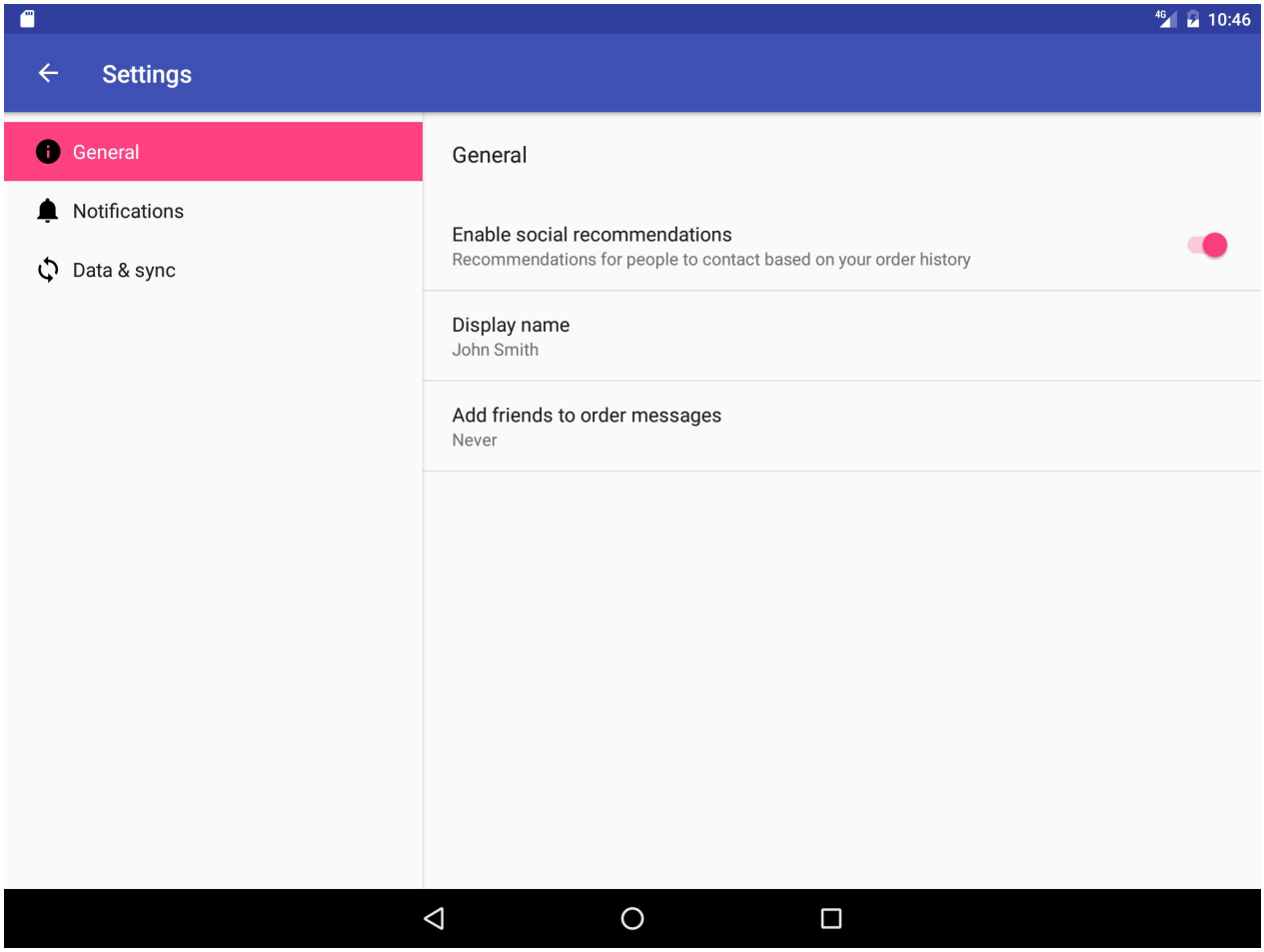
- Menggunakan Maksud yang memulai aktivitas yang menyatakan layar detail. Untuk informasi selengkapnya tentang Maksud, lihat Maksud dan Filter Maksud dalam Panduan Developer Android.
- Saat menambahkan Aktivitas Setelan, Anda bisa memperluas PreferenceActivity untuk membuat layout master/detail dua panel agar mendukung layar lebar, dan menyertakan fragmen dalam aktivitas untuk menggantikan materi aktivitas dengan fragmen setelan. Inilah pola yang berguna jika Anda memiliki beberapa grup setelan dan harus mendukung layar berukuran tablet serta ponsel cerdas. Anda akan mempelajari tentang Aktivitas Setelan dan PreferenceActivity

dalam bab berikutnya. Untuk informasi selengkapnya tentang menggunakan fragmen, lihat Fragmen dalam Panduan Developer Android.

Ponsel cerdas paling tepat untuk menampilkan satu layar setiap kalinya—seperti layar master (pada sisi kiri gambar di bawah) dan layar detail (pada sisi kanan gambar di bawah ini).



Sebaliknya, tampilan tablet, terutama saat dilihat pada orientasi lanskap, paling pas untuk menampilkan beberapa panel materi sekaligus: master di sebelah kiri, dan detail di sebelah kanan, seperti yang ditampilkan di bawah ini.

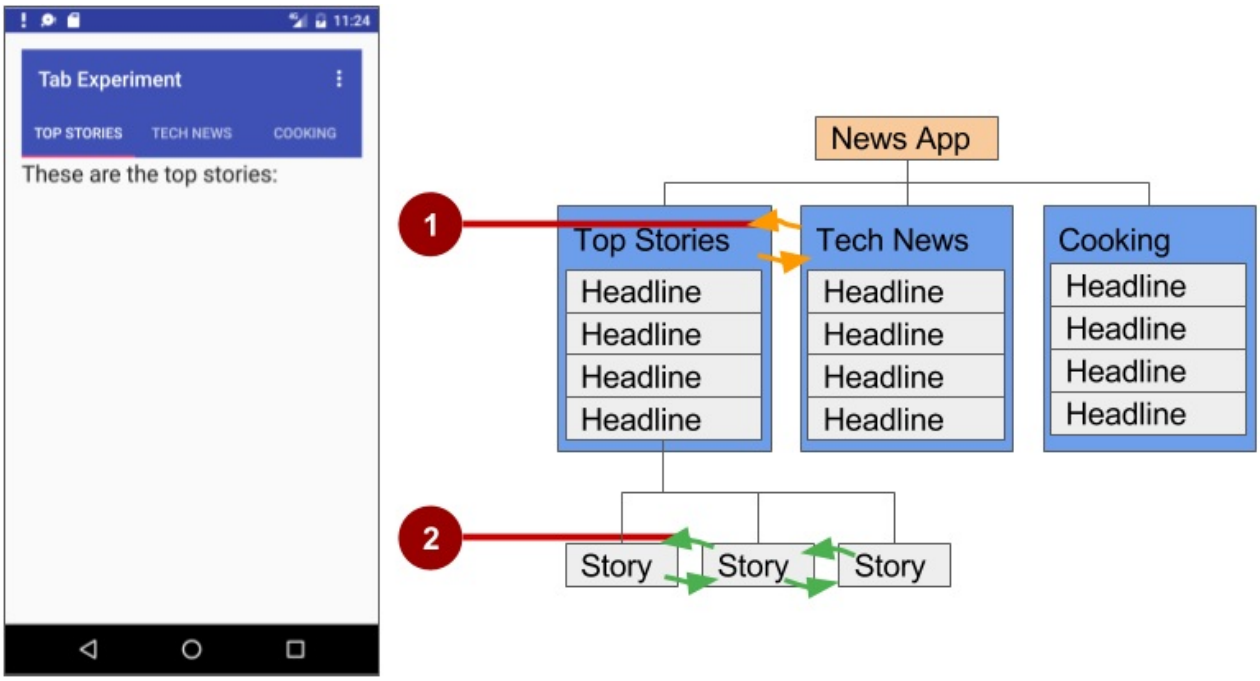


Menu opsi dalam bilah aplikasi

Bilah aplikasi umumnya berisi menu opsi, yang paling sering digunakan untuk pola navigasi bagi navigasi turunan. Bilah ini juga berisi ikon Naik dari navigasi leluhur, ikon navigasi untuk membuka panel samping navigasi, dan ikon filter untuk memfilter tampilan laman. Anda telah mempelajari cara mempersiapkan menu opsi dan bilah aplikasi dalam bab sebelumnya.

Navigasi lateral dengan tab dan gesek

Dengan navigasi lateral, Anda memungkinkan pengguna pergi dari satu saudara ke saudara yang lain (pada level yang sama dalam hierarki multitier). Misalnya, jika aplikasi Anda menyediakan sejumlah kategori cerita (seperti Top Stories, Tech News, dan Cooking, seperti yang ditampilkan dalam gambar di bawah ini), Anda mungkin ingin menyediakan bagi pengguna Anda kemampuan untuk beralih dari satu kategori ke kategori berikutnya, atau dari satu cerita terpopuler ke cerita berikutnya, tanpa harus mengarahkan kembali ke layar induk.



Dalam gambar di atas:

1. Navigasi lateral dari satu layar kategori ke layar kategori lainnya
2. Navigasi lateral dari satu layar cerita ke layar cerita lainnya

Contoh navigasi lateral lainnya adalah kemampuan untuk menggesek ke kiri atau kanan pada percakapan Gmail untuk menampilkan email baru atau lama dalam Inbox yang sama.

Anda bisa mengimplementasikan navigasi lateral dengan *tab* yang menyatakan setiap layar. Tab muncul sepanjang bagian atas layar, seperti yang ditampilkan pada sisi kiri gambar di atas, yang menyediakan navigasi ke layar lainnya. Navigasi tab adalah solusi umum untuk navigasi lateral dari satu layar anak ke layar anak lainnya yang *bersaudara*—di posisi yang sama dalam hierarki dan memiliki layar induk yang sama.

Tab paling tepat untuk rangkaian kecil (empat atau kurang) layar saudara. Anda bisa mengombinasikannya dengan tampilan gesek, sehingga pengguna bisa menggesek dari satu layar ke layar lainnya serta mengetuk tab.

Tab menawarkan dua manfaat:

- Karena ada tab tunggal yang lebih dulu dipilih, pengguna selalu memiliki akses ke materi tab tersebut dari layar induk tanpa navigasi lebih jauh.
- Pengguna bisa beralih dengan cepat di antara layar-layar terkait, tanpa perlu mengunjungi lagi layar induk terlebih dahulu.

Ingat praktik terbaik berikut saat menggunakan tab:

- Tab biasanya dihamparkan secara horizontal.
- Tab harus selalu memanjang di bagian atas layar, dan tidak boleh sejajar dengan bagian bawah layar.
- Tab harus menetap di semua layar yang terkait. Hanya region materi yang ditentukan yang akan berubah saat mengetuk tab, dan indikator harus tetap tersedia sepanjang waktu.
- Peralihan ke tab lain tidak boleh diperlakukan sebagai riwayat. Misalnya, jika seorang pengguna beralih dari tab A ke tab B, menekan tombol **Naik** pada bilah aplikasi tidak akan memilih lagi tab A, namun justru akan mengembalikan pengguna ke layar induk.

Langkah kunci untuk mengimplementasikan tab adalah:

1. Mendefinisikan layout tab. Kelas utama yang digunakan untuk menampilkan tab adalah TabLayout. Kelas ini menyediakan layout horizontal untuk menampilkan tab. Anda bisa menampilkan tab di bawah bilah aplikasi.
2. Mengimplementasikan Fragmen untuk setiap layar materi tab. *Fragmen* adalah perilaku atau bagian antarmuka pengguna dalam suatu aktivitas. Fragmen seperti aktivitas mini di dalam aktivitas utama, dengan daur hidupnya

sendiri. Salah satu manfaat menggunakan fragmen untuk materi tab adalah Anda bisa mengisolasi kode untuk mengelola materi tab dalam fragmen. Untuk mengetahui tentang fragmen, lihat Fragmen dalam Panduan API.

3. Menambahkan adapter pager. Gunakan kelas PagerAdapter untuk mengisi "laman" (layar) di dalam ViewPager, yaitu pengelola layout yang memungkinkan pengguna membalik ke kiri dan kanan pada layar data. Sediakan implementasi PagerAdapter untuk menghasilkan layar yang akan ditunjukkan oleh tampilan. ViewPager paling sering digunakan bersama Fragment, yang merupakan cara praktis untuk menyediakan dan mengelola daur hidup setiap layar.
4. Membuat instance layout tab, dan menyetel teks untuk setiap tab.
5. Menggunakan `PagerAdapter` untuk mengelola tampilan layar ("laman"). Setiap layar dinyatakan oleh fragmennya setiap.
6. Menyetel listener untuk menentukan tab mana yang diketuk.

Ada adapter standar untuk menggunakan fragmen bersama ViewPager:

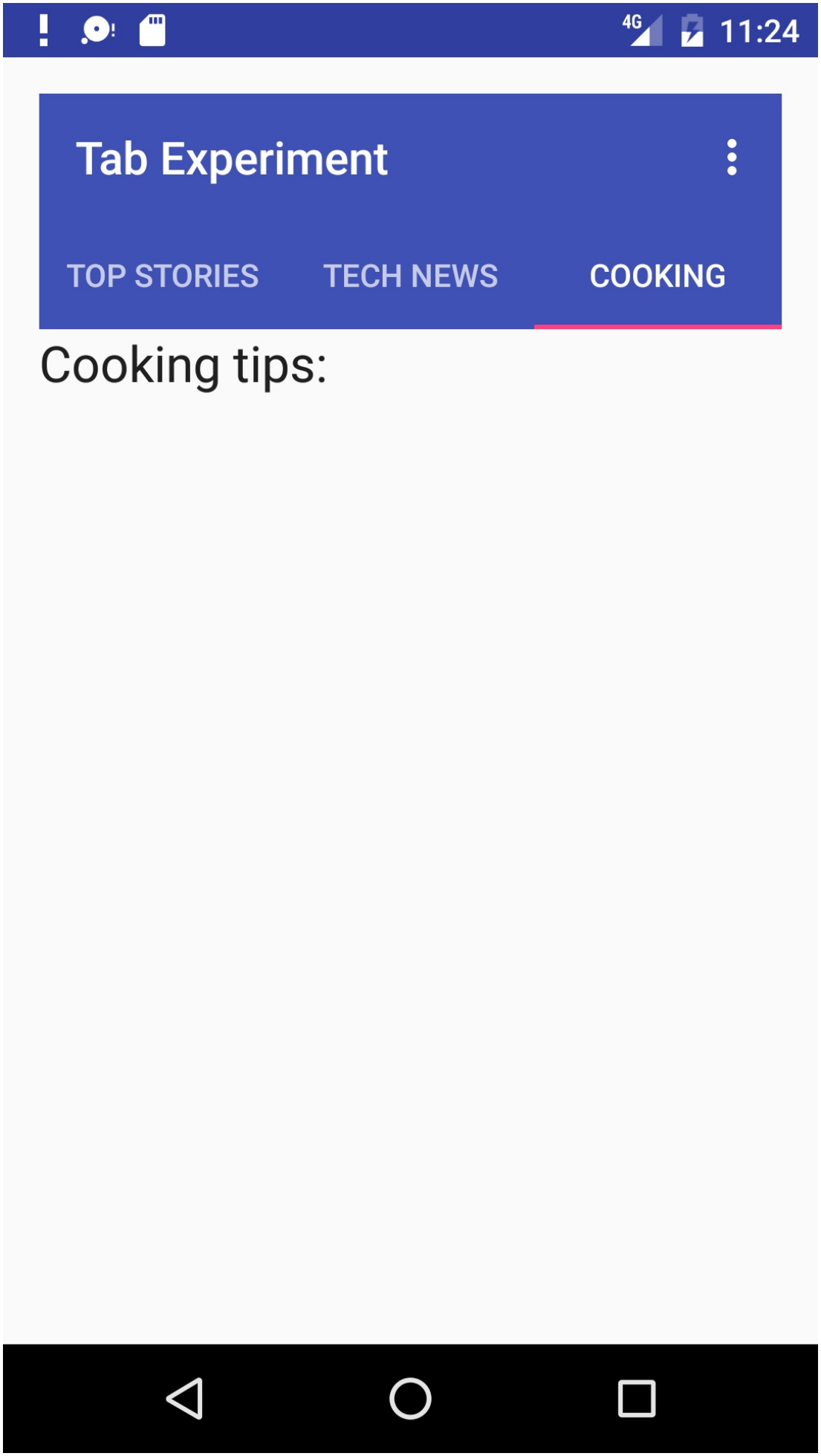
- FragmentPagerAdapter: Didesain untuk navigasi antar layar (laman) saudara yang menyatakan jumlah layar yang tetap dan sedikit.
- FragmentStatePagerAdapter: Didesain untuk paging ke semua kumpulan layar (laman) dengan jumlah layar tidak ditentukan. Hal ini akan memusnahkan fragmen saat pengguna beralih ke layar lain, sehingga meminimalkan penggunaan memori. Aplikasi untuk tantangan praktis ini menggunakan FragmentStatePagerAdapter.

Mendefinisikan layout tab

Untuk menggunakan TabLayout, Anda bisa mendesain layout aktivitas utama untuk menggunakan `Toolbar` sebagai bilah aplikasi, `TabLayout` untuk tab di bawah bilah aplikasi, dan `ViewPager` dalam layout akar untuk mengalihkan tampilan anak. Layout harus tampak sama dengan yang berikut ini, dengan anggapan setiap tampilan anak mengisi layar:

```
<android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:background="?attr/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>

    <android.support.design.widget.TabLayout
        android:id="@+id/tab_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/toolbar"
        android:background="?attr/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"/>

    <android.support.v4.view.ViewPager
        android:id="@+id/pager"
        android:layout_width="match_parent"
        android:layout_height="fill_parent"
        android:layout_below="@id/tab_layout"/>
```

Untuk setiap tampilan anak, buat file layout seperti tab_fragment1.xml, tab_fragment2.xml, tab_fragment3.xml, dan seterusnya.


11:24
Tab Experiment
TOP STORIES
TECH NEWS
COOKING
Cooking tips:

Mengimplementasikan setiap fragmen

Fragmen adalah perilaku atau bagian antarmuka pengguna dalam suatu aktivitas. Fragmen seperti aktivitas mini di dalam aktivitas utama, dengan daur hidupnya sendiri. Untuk mengetahui tentang fragmen, lihat Fragmen dalam Panduan API.

Tambahkan kelas untuk setiap fragmen (seperti TabFragment1.java, TabFragment2.java, dan TabFragment3.java) yang menyatakan layar yang bisa dikunjungi pengguna dengan mengeklik tab. Setiap kelas harus memperluas Fragment dan memekarkan layout yang terkait dengan layar (`tab_fragment1` , `tab_fragment2` , dan `tab_fragment3`). Misalnya, TabFragment1.java terlihat seperti ini:

```
public class TabFragment1 extends Fragment {

   @Override
   public View onCreateView(LayoutInflater inflater,
                  ViewGroup container, Bundle savedInstanceState) {
      return inflater.inflate(R.layout.tab_fragment1, container, false);
   }
}
```

Menambahkan adapter pager

Tambahkan sebuah `PagerAdapter` yang memperluas FragmentStatePagerAdapter dan:

1. Mendefinisikan jumlah tab.
2. Menggunakan metode getItem() kelas Adapter untuk menentukan tab mana yang diklik.
3. Menggunakan blok `switch case` untuk mengembalikan layar (laman) agar ditampilkan berdasarkan tab yang diklik

```
public class PagerAdapter extends FragmentStatePagerAdapter {
    int mNumOfTabs;

    public PagerAdapter(FragmentManager fm, int NumOfTabs) {
        super(fm);
        this.mNumOfTabs = NumOfTabs;
    }

    @Override
    public Fragment getItem(int position) {

        switch (position) {
            case 0:
                return new TabFragment1();
            case 1:
                return new TabFragment2();
            case 2:
                return new TabFragment3();
            default:
                return null;
        }
    }

    @Override
    public int getCount() {
        return mNumOfTabs;
    }
}
```

Membuat sebuah instance dari layout tab

Di metode `onCreate()` dari aktivitas utama, buat instance layout tab dari elemen `tab_layout` di layout, dan setel teks untuk setiap tab menggunakan addTab():

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   ...
   // Create an instance of the tab layout from the view.
   TabLayout tabLayout = (TabLayout) findViewById(R.id.tab_layout);
   // Set the text for each tab.
   tabLayout.addTab(tabLayout.newTab().setText("Top Stories"));
   tabLayout.addTab(tabLayout.newTab().setText("Tech News"));
   tabLayout.addTab(tabLayout.newTab().setText("Cooking"));
   // Set the tabs to fill the entire layout.
   tabLayout.setTabGravity(TabLayout.GRAVITY_FILL);
   // Use PagerAdapter to manage page views in fragments.
   ...
}
```

Ekstrak sumber daya string untuk teks tab yang disetel oleh `setText()`:

- `"Top Stories"` ke `tab_label1`
- `"Tech News"` ke `tab_label2`
- `"Cooking"` ke `tab_label3`

Mengelola tampilan layar dalam fragmen dan setel listener

Gunakan `PagerAdapter` dalam metode `onCreate()` aktivitas utama untuk mengelola tampilan layar ("laman") dalam fragmen. Setiap layar dinyatakan oleh fragmennya setiap. Anda juga perlu menyetel listener untuk menentukan tab mana yang diketuk. Kode berikut ini harus muncul setelah kode dari bagian sebelumnya dalam metode `onCreate()`:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   ...
   // Use PagerAdapter to manage page views in fragments.
   // Each page is represented by its own fragment.
   // This is another example of the adapter pattern.
   final ViewPager viewPager = (ViewPager) findViewById(R.id.pager);
   final PagerAdapter adapter = new PagerAdapter
               (getSupportFragmentManager(), tabLayout.getTabCount());
   viewPager.setAdapter(adapter);
   // Setting a listener for clicks.
   viewPager.addOnPageChangeListener(new
               TabLayout.TabLayoutOnPageChangeListener(tabLayout));
   tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
      @Override
      public void onTabSelected(TabLayout.Tab tab) {
         viewPager.setCurrentItem(tab.getPosition());
      }

      @Override
         public void onTabUnselected(TabLayout.Tab tab) {

      }

      @Override
         public void onTabReselected(TabLayout.Tab tab) {
      }
   });
}
```

Menggunakan ViewPager untuk tampilan gesek (paging horizontal)

ViewPager adalah pengelola layout yang memungkinkan pengguna membalik "laman" (layar) materi ke kiri dan ke kanan. ViewPager paling sering digunakan bersama Fragment, yang merupakan cara praktis untuk menyediakan dan mengelola daur hidup setiap "laman". ViewPager juga menyediakan kemampuan untuk menggesek "laman" secara horizontal.

Dalam contoh sebelumnya, Anda menggunakan `ViewPager` dalam layout akar untuk mengalihkan layar anak. Ini menyediakan kemampuan bagi pengguna untuk menggesek dari satu layar anak ke layar anak lainnya. Pengguna bisa beralih ke layar saudara dengan menyentuh dan menyeret layar secara horizontal dalam arah layar berdekatan yang diinginkan.

Tampilan gesek paling tepat jika ada kesamaan dalam tipe materi di antara laman yang bersaudara, dan jika jumlah saudara relatif kecil. Dalam kasus ini, pola bisa digunakan bersama tab di atas region materi untuk menunjukkan laman saat ini dan laman yang tersedia, guna membantu kemampuan untuk dapat ditemukan dan menyediakan lebih banyak konteks bagi pengguna.

Tip: Sebaiknya hindari paging secara horizontal bila layar anak berisi permukaan geser horizontal (seperti peta), karena interaksi yang saling bertentangan ini bisa menghalangi kegunaan layar Anda.

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Menggunakan Bilah Aplikasi dan Tab untuk Navigasi

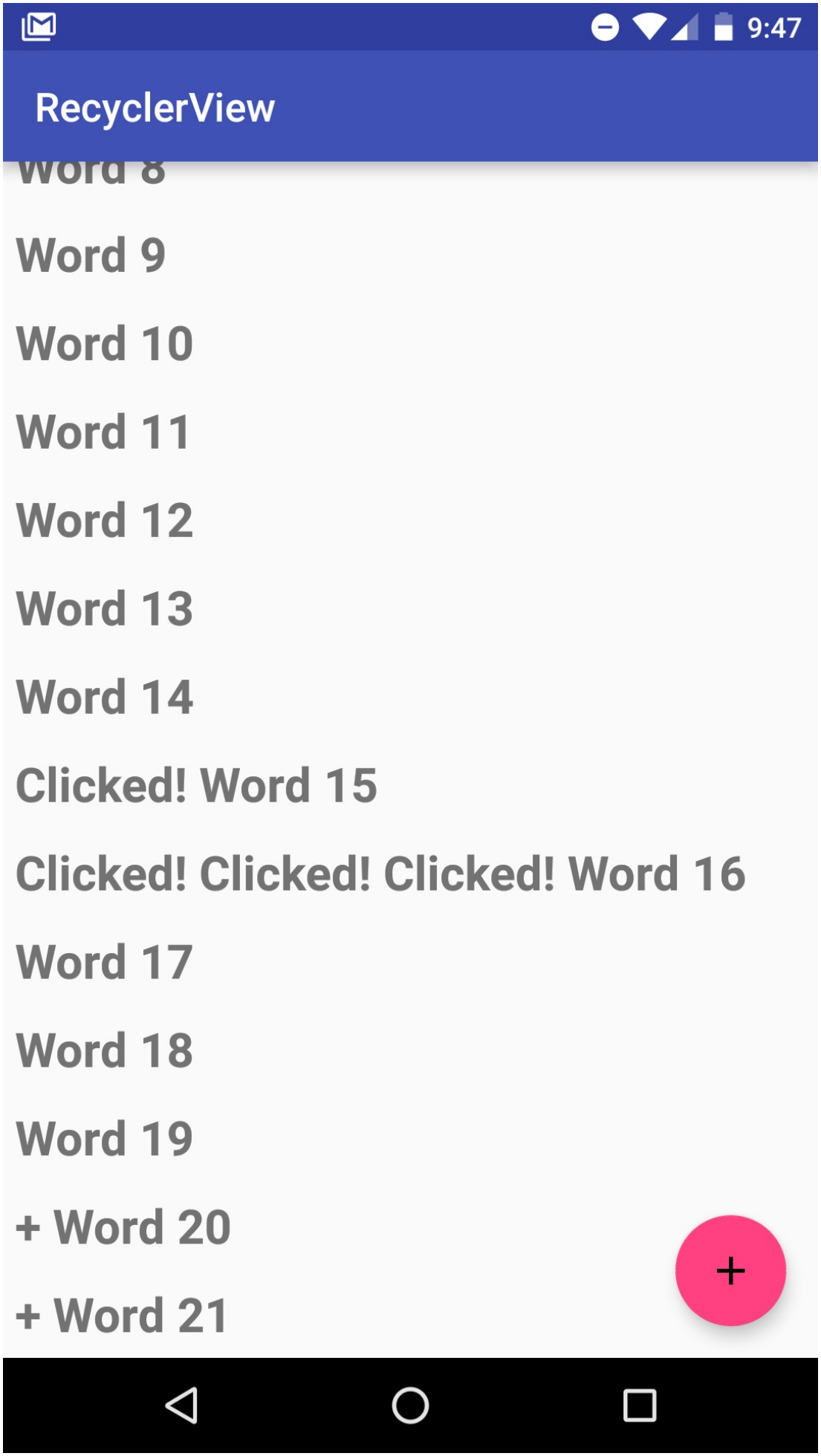
Ketahui selengkapnya

- Praktik Terbaik untuk Interaksi dan Keterlibatan
  - Mendesain Navigasi yang Efektif
  - Mengimplementasikan Navigasi yang Efektif
  - Membuat Tampilan Gesek dengan Tab
  - Membuat Panel Samping Navigasi
  - Menyediakan Navigasi Naik
  - Mengimplementasikan Navigasi Turunan
- Praktik Terbaik untuk Antarmuka Pengguna
- Desain - Pola - Navigasi
- Navigasi dengan Kembali dan Naik
- Bilah Aksi
- Menambahkan tutorial Bilah (Aplikasi) Aksi
- Menu

4.4: RecyclerView

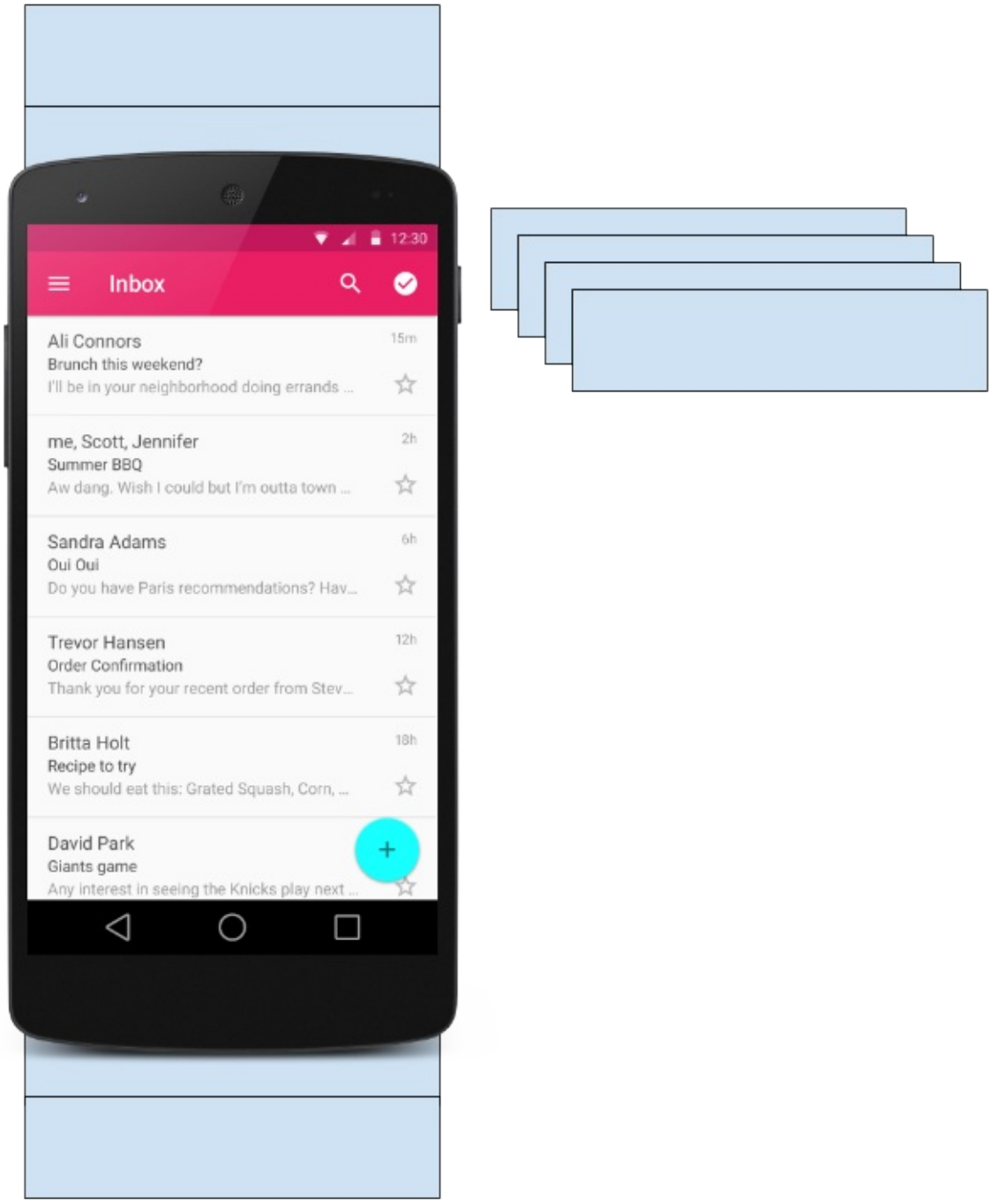
Materi:

Saat Anda menampilkan banyak item dalam daftar yang bisa digulir, sebagian besar item tidak terlihat. Misalnya, dalam daftar kata yang panjang atau banyak judul berita, pengguna hanya melihat sedikit item daftar untuk setiap kalinya.


9:47
RecyclerView
Word 8
Word 9
Word 10
Word 11
Word 12
Word 13
Word 14
Clicked! Word 15
Clicked! Clicked! Clicked! Word 16
Word 17
Word 18
Word 19
+ Word 20
+ Word 21

Atau, Anda bisa memiliki kumpulan data yang akan berubah saat pengguna berinteraksi dengannya. Jika Anda membuat tampilan baru setiap kali data berubah, itu juga membuat banyak tampilan, bahkan untuk kumpulan data yang kecil.

Dari perspektif kinerja, Anda bisa meminimalkan jumlah tampilan yang disimpan pada titik tertentu (Memori), dan jumlah tampilan yang harus Anda buat (Waktu). Kedua tujuan ini bisa dicapai dengan membuat agak lebih banyak tampilan daripada yang bisa dilihat pengguna pada layar, dan buat cache serta gunakan kembali tampilan yang dibuat sebelumnya dengan data berbeda saat pengguna menggulir ke dalam dan ke luar tampilan.



Kelas RecyclerView adalah versi ListView yang lebih canggih dan fleksibel. Widget ini adalah kontainer untuk menampilkan rangkaian data besar yang bisa digulir secara sangat efisien dengan mempertahankan tampilan dalam jumlah terbatas.

Gunakan widget RecyclerView bila Anda perlu menampilkan banyak data yang bisa digulir, atau kumpulan data dengan elemen yang berubah pada waktu proses berdasarkan aksi pengguna atau kejadian jaringan.

Komponen RecyclerView

Untuk menampilkan data dalam RecyclerView, Anda memerlukan bagian berikut:

- **Data.** Tidak penting dari mana asal data. Anda bisa membuat data secara lokal, seperti yang Anda lakukan dalam latihan, mendapatkannya dari database perangkat seperti yang akan Anda lakukan dalam praktik nanti, atau menariknya dari awan.
- **RecyclerView.** Daftar gulir yang berisi item daftar.

  Instance RecyclerView sebagaimana didefinisikan dalam file layout aktivitas Anda akan bertindak sebagai kontainer tampilan.

- **Layout untuk satu item data.** Semua item daftar tampak sama, sehingga Anda bisa menggunakan layout yang sama untuk semuanya. Layout item harus dibuat secara terpisah dari layout aktivitas, sehingga satu per satu tampilan item bisa dibuat dan diisi data.
- **Pengelola layout** Pengelola layout menangani penyusunan (layout) komponen antarmuka pengguna dalam suatu tampilan. Semua grup tampilan memiliki pengelola layout. Untuk LinearLayout, sistem Android menangani layout untuk Anda. RecyclerView memerlukan pengelola layout eksplisit untuk mengelola susunan item daftar yang terdapat di dalamnya. Layout ini bisa vertikal, horizontal, atau berupa petak.

  Pengelola layout adalah instance dari Recyclerview.LayoutManager untuk menyusun layout item dalam RecyclerView

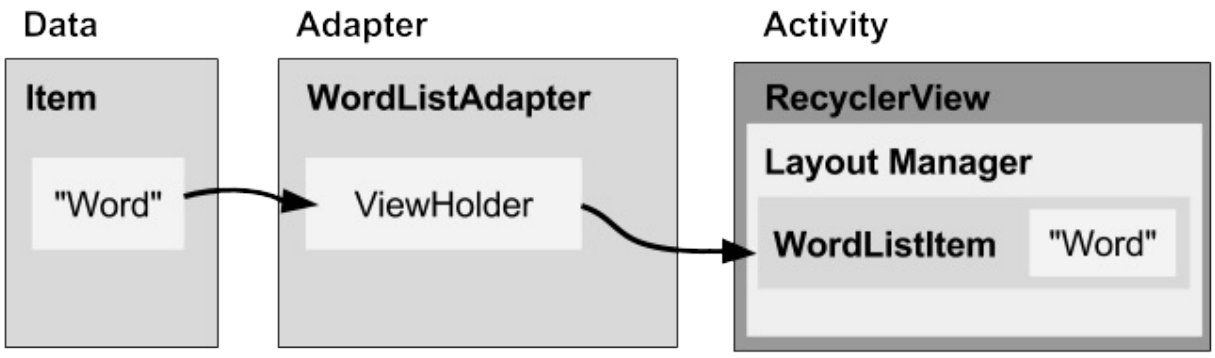
- **Adapter.** Adapter menghubungkan data Anda dengan RecyclerView. Adapter menyiapkan data dan cara menampilkan data dalam view holder. Bila data berubah, adapter akan memperbarui materi tampilan item daftar terkait dalam RecyclerView.

  Adapter juga merupakan ekstensi dari RecyclerView.Adapter. Adapter menggunakan ViewHolder untuk menampung tampilan yang menyusun setiap item dalam RecyclerView, dan mengikat data untuk ditampilkan dalam tampilan yang menampilkannya.

- **View holder.** View holder memperluas kelas ViewHolder. View holder berisi tampilan informasi untuk menampilkan satu item dari layout item.

  View holder digunakan oleh adapter untuk menyediakan data, yang merupakan ekstensi dari RecyclerView.ViewHolder

Diagram di bawah ini menampilkan hubungan antara komponen-komponen ini.



Data

Semua data yang bisa ditampilkan akan ditampilkan dalam RecyclerView.

- Teks
- Gambar
- Ikon

Data bisa berasal dari sumber mana pun.

- Dibuat oleh aplikasi. Misalnya, kata acak untuk permainan.

- Dari database lokal. Misalnya, daftar kontak.
- Dari storage awan atau internet. Misalnya judul berita.

RecyclerView

RecyclerView adalah:

- Suatu grup Tampilan untuk kontainer yang bisa digulir
- Ideal untuk daftar item serupa yang panjang
- Hanya menggunakan tampilan dalam jumlah terbatas yang digunakan kembali saat tampilan tersebut tidak tampak di layar. Hal ini menghemat memori dan mempercepat pembaruan item daftar saat pengguna menggulir data, karena tidak perlu membuat tampilan baru untuk setiap item yang muncul.
- Secara umum, RecyclerView menyimpan sebanyak mungkin tampilan item yang muat di layar, plus sedikit tambahan pada setiap akhir daftar untuk memastikan pengguliran berjalan cepat dan lancar.

Layout Item

Layout adalah sebuah item daftar yang disimpan dalam file terpisah sehingga adapter bisa membuat tampilan item dan mengedit materinya secara independen dari layout aktivitas.

Pengelola Layout

Pengelola layout memosisikan tampilan item di dalam grup tampilan, seperti RecyclerView dan menentukan kapan harus menggunakan kembali tampilan item yang tidak lagi terlihat oleh pengguna. Untuk menggunakan kembali (atau mendaur ulang) tampilan, pengelola layout bisa meminta adapter untuk mengganti materi tampilan dengan elemen lain dari kumpulan data. Mendaur ulang tampilan dengan cara ini akan meningkatkan kinerja karena menghindari pembuatan tampilan yang tidak diperlukan atau melakukan pencarian findViewById() yang mahal.

RecyclerView menyediakan semua pengelola layout bawaan ini:

- LinearLayoutManager menampilkan item dalam daftar gulir vertikal atau horizontal.
- GridLayoutManager menampilkan item dalam petak.
- StaggeredGridLayoutManager menampilkan item dalam petak zigzag.

Untuk membuat pengelola layout khusus, perluas kelas RecyclerView.LayoutManager.

Animasi

Animasi untuk menambahkan dan menghapus item diaktifkan secara default dalam RecyclerView. Untuk menyesuaikan animasi ini, perluas kelas RecyclerView.ItemAnimator dan gunakan metode RecyclerView.setItemAnimator().

Adapter

Adapter membantu dua antarmuka yang tidak kompatibel untuk bekerja bersama. Dalam RecyclerView, adapter menghubungkan data dengan tampilan. Adapter bertindak sebagai perantara antara data dan tampilan. Adapter menerima atau mengambil data, melakukan semua pekerjaan yang diperlukan agar bisa ditampilkan dalam suatu tampilan, dan menempatkan data dalam tampilan.

Misalnya, adapter bisa menerima data dari database sebagai objek Cursor, mengekstrak kata dan definisinya, mengonversinya menjadi string, dan menempatkan string dalam suatu tampilan item yang memiliki tampilan teks, satu untuk kata dan satu untuk definisi. Anda akan mengetahui selengkapnya tentang kursor dalam bab berikutnya.

RecyclerView.Adapter mengimplementasikan sebuah view holder, dan harus mengganti callback berikut:

- `onCreateViewHolder()` memekarkan tampilan item dan mengembalikan view holder baru yang memuatnya. Metode ini dipanggil bila RecyclerView memerlukan view holder baru untuk menyatakan suatu item.
- `onBindViewHolder()` menyetel materi item pada posisi yang ditentukan dalam RecyclerView. Proses ini dipanggil oleh RecyclerView, misalnya, bila sebuah item baru bergulir ke tampilan.

View holder

RecyclerView.ViewHolder menjelaskan tampilan data dan metadata tentnag tempatnya dalam RecyclerView. Setiap view holder menampung satu rangkaian data. Adapter menambahkan data ke view holder untuk ditampilkan oleh pengelola layout.

Definisikan layout view holder Anda dalam file sumber daya XML. Layout ini bisa berisi (hampir) semua tipe tampilan, termasuk elemen yang bisa diklik.

Mengimplementasikan RecyclerView

Mengimplementasikan RecyclerView memerlukan langkah-langkah berikut:

1. Tambahkan dependensi RecyclerView ke file app/build.gradle aplikasi.
2. Tambahkan RecyclerView ke layout aktivitas
3. Buat file XML layout untuk satu item
4. Perluas RecyclerView.Adapter dan implementasikan metode onCrateViewHolder serta onBindViewHolder.
5. Perluas RecyclerView.ViewHolder untuk membuat view holder bagi layout item Anda. Anda bisa menambahkan perilaku klik dengan mengganti metode onClick.
6. Dalam aktivitas Anda, dalam metode onCreate method, buat RecyclerView dan inisialisasi dengan adapter serta pengelola layout.

1. Tambahkan dependensi ke app/build.gradle

Tambahkan pustaka RecyclerView ke file app/build.gradle Anda sebagai dependensi. Lihat bab mengenai pustaka dukungan atau praktik RecyclerView, jika Anda memerlukan petunjuk detail.

```
dependencies {
    ...
    compile 'com.android.support:recyclerview-v7:24.1.1'
    ...
}
```

2. Tambahkan RecyclerView ke layout aktivitas Anda

Tambahkan RecyclerView dalam file layout aktivitas Anda.

```
    <android.support.v7.widget.RecyclerView
        android:id="@+id/recyclerview"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </android.support.v7.widget.RecyclerView>
```

Gunakan RecyclerView dari pustaka dukungan agar kompatibel dengan perangkat lama. Satu-satunya atribut yang diperlukan adalah id, beserta lebar dan tinggi. Sesuaikan item tersebut, bukan grup tampilan ini.

3. Buat layout untuk satu item

Buat file sumber daya XML dan tetapkan layout satu item. File ini akan digunakan oleh adapter untuk membuat view holder.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="6dp">

    <TextView
        android:id="@+id/word"
        style="@style/word_title" />

</LinearLayout>
```

Tampilan teks memiliki elemen `@style` . Gaya adalah kumpulan properti yang menetapkan sosok suatu tampilan. Anda bisa menggunakan gaya untuk berbagi atribut tampilan dengan beberapa tampilan. Sebuah cara mudah untuk membuat gaya adalah dengan mengekstrak gaya elemen UI yang sudah Anda buat. Misalnya, setelah menata gaya TextView, **Right-click > Refactor > Extract > Style** pada elemen tersebut dan ikuti konfirmasi dialog. Lebih detail mengenai gaya ada dalam praktik dan dalam bab berikutnya.

4. Buat adapter dengan view holder

Perluas RecyclerView.Adapter dan implementasikan metode onCrateViewHolder serta onBindViewHolder.

Buat kelas Java baru dengan tanda tangan berikut:

```
public class WordListAdapter extends RecyclerView.Adapter<WordListAdapter.WordViewHolder> {}
```

Dalam konstruktor, dapatkan inflater dari konteks saat ini, dan data Anda.

```
    public WordListAdapter(Context context, LinkedList<String> wordList) {
        mInflater = LayoutInflater.from(context);
        this.mWordList = wordList;
    }
```

Untuk adapter ini, Anda harus mengimplementasikan 3 metode.

- onCreateViewHolder() membuat suatu tampilan dan mengembalikannya.

  ```
  @Override
  public WordViewHolder onCreateViewHolder(ViewGroup parent, int viewType){
      // Inflate an item view.
      View mItemView = mInflater.inflate(R.layout.wordlist_item, parent, false);
      return new WordViewHolder(mItemView, this);
  }
  ```

- onBindViewHolder() menghubungkan data dengan view holder pada posisi yang ditentukan dalam RecyclerView.

  ```
  @Override
  public void onBindViewHolder(WordViewHolder holder, int position) {
      // Retrieve the data for that position
      String mCurrent = mWordList.get(position);
      // Add the data to the view
      holder.wordItemView.setText(mCurrent);
  }
  ```

- getItemCount() kembali ke jumlah item data yang tersedia untuk ditampilkan.

```
 @Override
public int getItemCount() {
    return mWordList.size();
}
```

5. Implementasikan kelas view holder

Perluas RecyclerView.ViewHolder untuk membuat view holder bagi layout item Anda. Anda bisa menambahkan perilaku klik dengan mengganti metode onClick.

Kelas ini biasanya didefinisikan sebagai kelas dalam untuk adapter dan memperluas RecyclerView.ViewHolder.

```
class WordViewHolder extends RecyclerView.ViewHolder {}
```

Jika Anda ingin menambahkan penanganan klik, Anda perlu mengimplementasikan listener klik. Salah satu cara untuk melakukannya adalah dengan memiliki view holder yang mengimplementasikan metode listener klik.

```
// Extend the signature of WordVewHolder to implement a click listener.
class WordViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {}
```

Dalam konstruktornya, view holder harus memekarkan layoutnya, menghubungkannya dengan adapter, dan jika berlaku, menyetel listener klik.

```
public WordViewHolder(View itemView, WordListAdapter adapter) {
    super(itemView);
    wordItemView = (TextView) itemView.findViewById(R.id.word);
    this.mAdapter = adapter;
    itemView.setOnClickListener(this);
}
```

Dan, jika Anda mengimplementasikan onClickListener, Anda juga harus mengimplementasikan onClick().

```
@Override
public void onClick(View v) {
    wordItemView.setText ("Clicked! "+ wordItemView.getText());
}
```

Perhatikan, untuk memasang listener klik ke elemen view holder lain, Anda harus melakukannya secara dinamis dalam onBindViewHolder. (Anda akan melakukan ini pada praktik nanti, saat Anda akan memperluas kode RecyclerView dari praktik.)

6. Buat RecyclerView

Terakhir, untuk mengikat semuanya, dalam metode onCreate() aktivitas Anda:

1. Dapatkan penanganan untuk RecyclerView.

   ```
   mRecyclerView = (RecyclerView) findViewById(R.id.recyclerview);
   ```

2. Buat adapter dan berikan data untuk ditampilkan.

   ```
   mAdapter = new WordListAdapter(this, mWordList);
   ```

3. Hubungkan adapter dengan RecyclerView.

   ```
   mRecyclerView.setAdapter(mAdapter);
   ```

4. Buat RecyclerView dengan pengelola layout default.

```
mRecyclerView.setLayoutManager(new LinearLayoutManager(this));
```

RecyclerView adalah cara efisien untuk menampilkan data daftar gulir. RecyclerView menggunakan pola adapter untuk menghubungkan data dengan tampilan item daftar. Untuk mengimplementasikan RecyclerView, Anda perlu membuat adapter dan view holder, dan metode yang mengambil data serta menambahkannya ke item daftar.

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Buat RecyclerView

Ketahui selengkapnya

- RecyclerView
- Kelas RecyclerView
- Kelas RecyclerView.Adapter
- Kelas RecyclerView.ViewHolder
- Kelas RecyclerView.LayoutManager

5.1: Sumber Daya Dapat Digambar, Gaya, dan Tema

Daftar Isi:

Dalam bab ini Anda akan mengetahui cara menggunakan *sumber daya dapat digambar*, yaitu kompilasi gambar yang bisa digunakan dalam aplikasi. Android menyediakan kelas dan sumber daya untuk membantu Anda menyertakan gambar sempurna dalam aplikasi dengan dampak minimal pada kinerjanya.

Anda juga akan mengetahui cara menggunakan gaya serta tema agar menghasilkan penampilan yang konsisten untuk semua elemen dalam aplikasi dengan mengurangi jumlah kode.

Sumber Daya Dapat Digambar

Sumber daya dapat digambar adalah grafik yang bisa digambar ke layar. Anda mengambil sumber daya dapat digambar menggunakan API seperti `getDrawable(int)`, dan menerapkan sumber daya dapat digambar ke sumber daya XML dengan menggunakan atribut seperti `android:drawable` dan `android:icon`.

Android menyertakan sejumlah tipe sumber daya dapat digambar, sebagian besar di dibahas dalam bab ini.

Yang dibahas dalam bab ini:	Yang tidak dibahas dalam bab ini:
• File gambar • File nine-patch • Daftar layer • Sumber daya dapat digambar untuk bentuk • Daftar keadaan • Daftar level • Sumber daya dapat digambar untuk transisi • Sumber daya dapat digambar untuk vektor	• Sumber daya dapat digambar untuk skala • Sumber daya dapat digambar untuk sisipan • Sumber daya dapat digambar untuk klip

Menggunakan sumber daya dapat digambar

Untuk menampilkan sumber daya dapat digambar, gunakan kelas `ImageView` untuk membuat Tampilan. Di elemen `<ImageView>` dalam file XML Anda, definisikan cara menampilkan sumber daya dapat digambar dan tempat menggambarnya. Misalnya, `ImageView` ini menampilkan gambar yang disebut "birthdaycake.png":

```
<ImageView
      android:id="@+id/tiles"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:src="@drawable/birthdaycake" />
```

Tentang atribut `<ImageView>`:

- Atribut `android:id` menyetel nama pintasan yang Anda gunakan nanti untuk memanggil gambar.
- Atribut `android:layout_width` dan `android:layout_height` menetapkan ukuran Tampilan. Dalam contoh ini, tinggi dan lebar disetel ke `wrap_content`, yang berarti Tampilan hanya cukup besar untuk memasukkan gambar di dalamnya,

plus pengisi.

- Atribut `android:src` memberikan lokasi menyimpan gambar. Jika Anda memiliki beberapa versi gambar yang cocok untuk beberapa resolusi layar berbeda, simpan gambar dalam folder bernama res/drawable-[density]/. Misalnya, simpan versi birthdaycake.png yang cocok untuk layar hdpi dalam res/drawable-hdpi/birthdaycake.png. Untuk informasi selengkapnya, lihat panduan multilayar.
- `<ImageView>` juga memiliki atribut yang bisa Anda gunakan untuk memotong gambar jika gambar terlalu besar atau memiliki rasio aspek berbeda dari layout atau Tampilan. Untuk detail lengkap, lihat dokumentasi kelas `ImageView`.

Untuk menyatakan sumber daya dapat digambar dalam aplikasi Anda, gunakan kelas `Drawable` atau salah satu subkelasnya. Misalnya, kode ini mengambil gambar birthdaycake.png sebagai `Drawable`:

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

File gambar

File gambar merupakan file bitmap generik. Android mendukung file gambar dalam sejumlah format: WebP (diutamakan), PNG (diutamakan), dan JPG (diterima). Format GIF dan BMP didukung, namun tidak disarankan.

Format WebP sepenuhnya didukung dari Android 4.2. WebP memadatkan lebih baik daripada format lain dalam hal kompresi lossless dan lossy, yang berpotensi menghasilkan gambar lebih dari 25% lebih kecil daripada format JPEG. Anda bisa mengonversi gambar PNG dan JPEG yang ada menjadi format WebP sebelum diunggah. Untuk informasi selengkapnya tentang WebP, lihat dokumentasi WebP.

Simpan file gambar dalam folder res/drawable. Gunakan gambar tersebut bersama atribut `android:src` untuk `ImageView` dan turunannya, atau untuk membuat kelas BitmapDrawable dalam kode Java.

Ketahuilah bahwa gambar terlihat berbeda pada layar yang memiliki kepadatan piksel serta rasio aspek yang berbeda. Untuk informasi tentang mendukung ukuran layar yang berbeda, lihat Mempercepat aplikasi Anda, di bawah ini, dan panduan ukuran layar.

Catatan: Selalu gunakan gambar dengan ukuran yang sesuai, karena gambar bisa menggunakan banyak ruang disk dan memengaruhi kinerja aplikasi Anda.

File nine-patch

9-patch adalah gambar PNG yang Anda gunakan untuk mendefinisikan region yang dapat direntang. Gunakan 9-patch sebagai gambar latar belakang bagi Tampilan untuk memastikan Tampilan terlihat pas untuk orientasi dan ukuran layar yang berbeda.

Misalnya, dalam Tampilan yang memiliki `layout_width` disetel ke `"wrap_content"`, Tampilan akan tetap cukup besar untuk menampung isinya (plus pengisi). Jika Anda menggunakan gambar PNG biasa sebagai gambar latar belakang bagi Tampilan, gambar mungkin akan terlalu kecil bagi Tampilan pada beberapa perangkat karena Tampilan direntang untuk mengakomodasi isinya. Jika Anda menggunakan gambar 9-patch, gambar tersebut juga akan direntang saat Tampilan direntang.

Widget Button standar Android merupakan contoh Tampilan yang menggunakan 9-patch sebagai gambar latar belakangnya. 9-patch direntang untuk mengakomodasi teks atau gambar di dalam tombol.

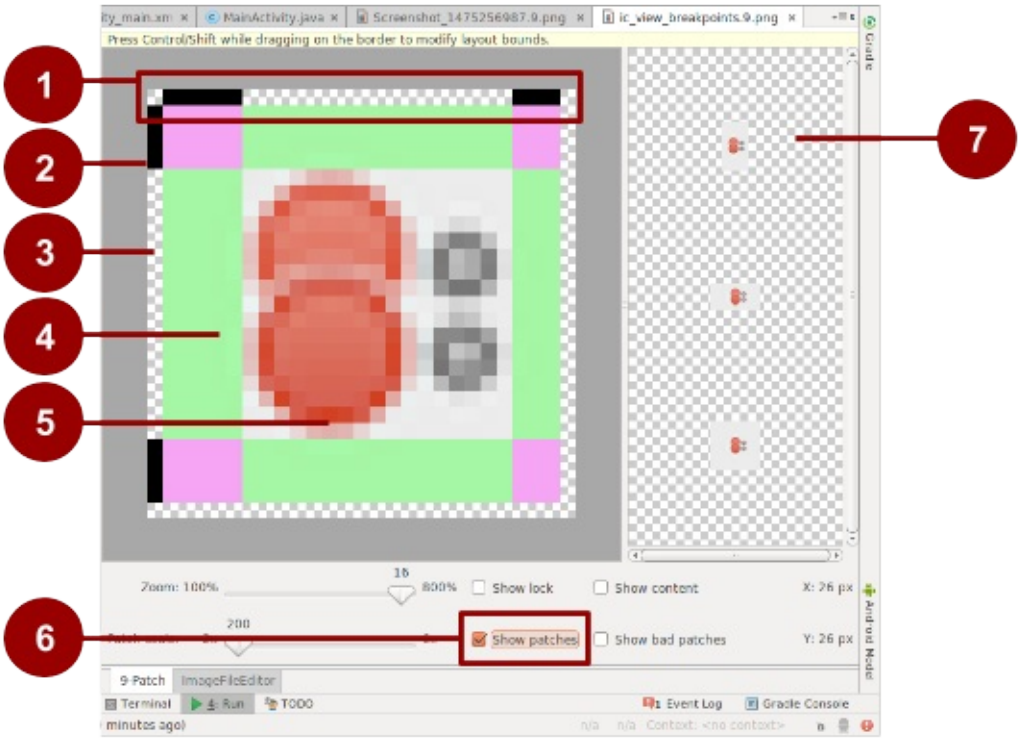
Simpan file 9-patch dengan ekstensi 9.png dalam folder res/drawable. Gunakan gambar tersebut bersama atribut `android:src` untuk `ImageView` dan turunannya, atau untuk membuat kelas `NinePatchDrawable` dalam kode Java.

Untuk membuat 9-patch, gunakan alat (bantu) Draw 9-Patch di Android Studio. Alat (bantu) tersebut memungkinkan Anda memulai dengan PNG biasa dan mendefinisikan border 1 piksel di sekeliling gambar di tempat yang memungkinkan sistem Android merentang gambar jika diperlukan. Untuk menggunakan alat (bantu):

1. Masukkan file PNG dalam folder res/drawable. (Caranya, salin file gambar dalam folder app/src/main/res/drawable proyek Anda.)

2. Dalam Android Studio, klik-kanan pada file dan pilih **Create 9-Patch file**. Android Studio menyimpan file dengan ekstensi .9.png.
3. Di Android Studio, klik dua kali file .9.png untuk membuka editor.
4. Tetapkan region gambar yang bisa direntang.

#



1. Border untuk menunjukkan region yang siap direntang melebar (secara horizontal).

   Misalnya, dalam Tampilan yang lebih lebar daripada gambar, setrip hijau di sisi kiri dan kanan 9-patch ini bisa direntang untuk mengisi Tampilan. Tempat yang bisa direntang ditandai dengan warna hitam. Klik untuk menghitamkan piksel.

2. Border untuk menunjukkan region yang siap direntang memanjang (secara vertikal). Misalnya, dalam Tampilan yang lebih tinggi daripada gambar, setrip hijau di bagian atas dan bawah 9-patch ini bisa direntang untuk mengisi Tampilan.
3. Nonaktifkan piksel dengan mengeklik tombol shift (klik ctrl pada Mac).
4. Area yang dapat direntang.
5. Tidak dapat direntang.
6. Centang **Show patches** untuk pratinjau jalur yang dapat direntang dalam area menggambar.
7. Pratinjau gambar yang direntang.
   Tip: Pastikan region yang dapat direntang setidaknya berukuran 2x2 piksel. Jika tidak, region bisa menghilang bila skala gambar diturunkan.

Untuk pembahasan lebih detail tentang cara membuat file 9-patch dengan region yang dapat direntang, lihat panduan 9-patch.

Sumber daya dapat digambar untuk daftar layer

Di Android, Anda bisa membangun gambar dengan melapiskan gambar lain, seperti yang bisa dilakukan di GIMP dan program manipulasi gambar lainnya. Setiap layer dinyatakan oleh sumber daya dapat digambar individual. Sumber daya dapat digambar yang membentuk gambar tunggal diatur dan dikelola dalam suatu elemen `<layer-list>` dalam XML. Dalam `<layer-list>`, setiap sumber daya dapat digambar dinyatakan melalui elemen `<item>`.

Layer digambar di atas setiap dengan urutan yang didefinisikan dalam file XML, berarti sumber daya dapat digambar yang terakhir dalam daftar akan digambar paling atas. Misalnya, sumber daya dapat digambar untuk daftar layer dibuat dari tiga sumber daya dapat digambar yang saling melapis: 

Dalam XML berikut, yang mendefinisikan daftar layer ini, gambar `android_blue` didefinisikan terakhir, sehingga digambar terakhir dan ditampilkan paling atas:

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
      <bitmap android:src="@drawable/android_red"
        android:gravity="center" />
    </item>
    <item android:top="10dp" android:left="10dp">
      <bitmap android:src="@drawable/android_green"
        android:gravity="center" />
    </item>
    <item android:top="20dp" android:left="20dp">
      <bitmap android:src="@drawable/android_blue"
        android:gravity="center" />
    </item>
</layer-list>
```

`LayerDrawable` adalah objek sumber daya dapat digambar yang mengelola larik sumber daya dapat digambar lainnya. Untuk informasi selengkapnya tentang cara menggunakan sumber daya dapat digambar untuk daftar layer, lihat panduan daftar layer.

Sumber daya dapat digambar untuk bentuk

Sumber daya dapat digambar untuk bentuk adalah persegi panjang, oval, garis, atau cincin yang Anda definisikan dalam XML. Anda bisa menetapkan ukuran dan gaya bentuk dengan menggunakan atribut XML.

Misalnya, file XML ini membuat persegi panjang dengan sudut tumpul dan gradien warna. Warna pengisi persegi berubah dari putih (`#000000`) di sudut kiri bawah menjadi biru (`#0000dd)` di sudut kanan atas. Atribut `angle` menentukan cara gradien dimiringkan:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <corners android:radius="8dp" />
    <gradient
        android:startColor="#000000"
        android:endColor="#0000dd"
        android:angle="45"/>
    <padding android:left="7dp"
        android:top="7dp"
        android:right="7dp"
        android:bottom="7dp" />
</shape>
```

Dengan anggapan bahwa file XML sumber daya dapat digambar untuk bentuk disimpan di res/drawable/gradient_box.xml, XML layout berikut menerapkan sumber daya dapat digambar untuk bentuk sebagai latar belakang untuk Tampilan:

```
<TextView
    android:background="@drawable/gradient_box"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content" />
```

Kode berikut menampilkan cara mendapatkan sumber daya dapat digambar untuk bentuk lewat program dan menggunakannya sebagai latar belakang Tampilan, sebagai alternatif untuk mendefinisikan atribut latar belakang dalam XML:

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

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

Anda bisa menyetel atribut lain bagi sumber daya dapat digambar untuk bentuk. Sintaks lengkapnya adalah seperti berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape=["rectangle" | "oval" | "line" | "ring"] >
    <!-- If it's a line, the stroke element is required. -->
    <corners
        android:radius="integer"
        android:topLeftRadius="integer"
        android:topRightRadius="integer"
        android:bottomLeftRadius="integer"
        android:bottomRightRadius="integer" />
    <gradient
        android:angle="integer"
        <!-- The angle must be 0 or a multiple of 45 -->
        android:centerX="float"
        android:centerY="float"
        android:centerColor="integer"
        android:endColor="color"
        android:gradientRadius="integer"
        android:startColor="color"
        android:type=["linear" | "radial" | "sweep"]
        android:useLevel=["true" | "false"] />
    <padding
        android:left="integer"
        android:top="integer"
        android:right="integer"
        android:bottom="integer" />
    <size
        android:width="integer"
        android:height="integer" />
    <solid
        android:color="color" />
    <stroke
        android:width="integer"
        android:color="color"
        android:dashWidth="integer"
        android:dashGap="integer" />
</shape>
```

Untuk detail tentang atribut ini, lihat referensi sumber daya dapat digambar untuk bentuk.

Sumber daya dapat digambar untuk daftar keadaan

`StateListDrawable` adalah objek sumber daya dapat digambar yang menggunakan gambar berbeda untuk menyatakan objek yang sama, bergantung pada keadaan objek tersebut berada. Misalnya, sebuah widget Button bisa ada dalam salah satu dari sejumlah keadaan (ditekan, difokuskan, diarahkan ke atas, atau tidak satu pun dari keadaan ini). Dengan menggunakan sumber daya dapat digambar untuk daftar keadaan, Anda bisa menyediakan beragam gambar latar untuk setiap keadaan.

Anda bisa menjelaskan daftar keadaan dalam file XML. Setiap grafik dinyatakan melalui elemen `<item>` di dalam sebuah elemen `<selector>` . Setiap `<item>` menggunakan sebuah atribut `state_` untuk menunjukkan situasi tempat menggunakan grafik.

Selama setiap perubahan keadaan, Android akan menyusuri daftar keadaan dari atas ke bawah. Item pertama yang cocok dengan keadaan saat ini akan digunakan, yang berarti pemilihan ini tidak berdasarkan pada "kecocokan terbaik," melainkan cukup dengan item pertama yang memenuhi kriteria minimum keadaan tersebut.

Daftar keadaan dalam contoh berikut mendefinisikan gambar yang akan ditampilkan bila tombol dalam keadaan berbeda. Bila tombol ditekan—yaitu, bila `state_pressed="true"` —aplikasi akan menampilkan gambar bernama button_pressed. Bila tombol sedang difokus (`state_focused="true"`), atau bila tombol diarahkan ke atas (`state_hovered="true"`), aplikasi akan menampilkan tombol berbeda.

```
<?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_focused" /> <!-- focused -->
    <item android:state_hovered="true"
          android:drawable="@drawable/button_focused" /> <!-- hovered -->
    <item android:drawable="@drawable/button_normal" /> <!-- default -->
</selector>
```

Keadaan lain yang tersedia antara lain `android:state_selected,` `android:state_checkable, android:state_checked` , dan lainnya. Untuk detail tentang semua opsi, lihat panduan daftar keadaan.

Sumber daya dapat digambar untuk daftar level

Sumber daya dapat digambar untuk daftar level mendefinisikan alternatif sumber daya dapat digambar, setiap mendapat nilai numerik maksimum. Untuk memilih sumber daya dapat digambar yang akan digunakan, panggil metode `setLevel()` , yang meneruskan integer yang cocok dengan integer tingkat maksimum yang didefinisikan dalam XML. Sumber daya dengan tingkat maksimum terendah lebih besar dari atau sama dengan integer yang diteruskan ke dalam `setLevel()` akan dipilih.

Misalnya, XML berikut mendefinisikan daftar level yang menyertakan dua alternatif sumber daya dapat digambar, `status_off` dan `status_on` :

```
<?xml version="1.0" encoding="utf-8"?>
<level-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:drawable="@drawable/status_off"
        android:maxLevel="0" />
    <item
        android:drawable="@drawable/status_on"
        android:maxLevel="1" />
</level-list>
```

Untuk memilih sumber daya dapat digambar `status_off` , panggil `setLevel(0)` . Untuk memilih sumber daya dapat digambar `status_on` , panggil `setLevel(1)` .

Contoh penggunaan `LevelListDrawable` adalah ikon indikator tingkat daya baterai yang menggunakan gambar berbeda untuk menunjukkan beragam tingkat daya baterai saat ini.

Sumber daya dapat digambar untuk transisi

`TransitionDrawable` adalah sumber daya dapat digambar yang memudar bersilang di antara dua sumber daya dapat digambar. Untuk mendefinisikan sumber daya dapat digambar untuk transisi dalam XML, gunakan elemen `<transition>` . Setiap sumber daya dapat digambar dinyatakan melalui elemen `<item>` di dalam elemen `<transition>` . Tidak lebih dari dua elemen `<item>` yang didukung.

Misalnya, sumber daya dapat digambar ini memudar bersilang di antara sumber daya dapat digambar untuk keadaan "hidup" dan "mati":

```
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/on" />
    <item android:drawable="@drawable/off" />
</transition>
```

Untuk transisi maju, artinya berubah dari sumber daya dapat digambar yang pertama ke yang kedua, panggil `startTransition()` . Untuk transisi ke arah lain, panggil `reverseTransition()` . Setiap metode ini memerlukan argumen bertipe `int` , yang menyatakan jumlah milidetik untuk transisi.

Sumber daya dapat digambar untuk vektor

Di Android 5.0 (API level 21) dan di atasnya, Anda bisa mendefinisikan *sumber daya dapat digambar untuk vektor*, yakni gambar yang didefinisikan oleh suatu jalur. Sumber daya dapat digambar untuk vektor menskalakan tanpa kehilangan definisi. Sebagian besar sumber daya dapat digambar untuk vektor menggunakan file SVG, yaitu file teks biasa atau file biner kompresi yang menyertakan koordinat dua dimensi sebagai cara menggambar pada layar.

Karena file SVG adalah teks, file tersebut lebih hemat ruang daripada file gambar lainnya. Selain itu, Anda juga hanya memerlukan satu file untuk gambar vektor sebagai ganti satu file untuk setiap kepadatan layar, seperti kasus untuk gambar bitmap.

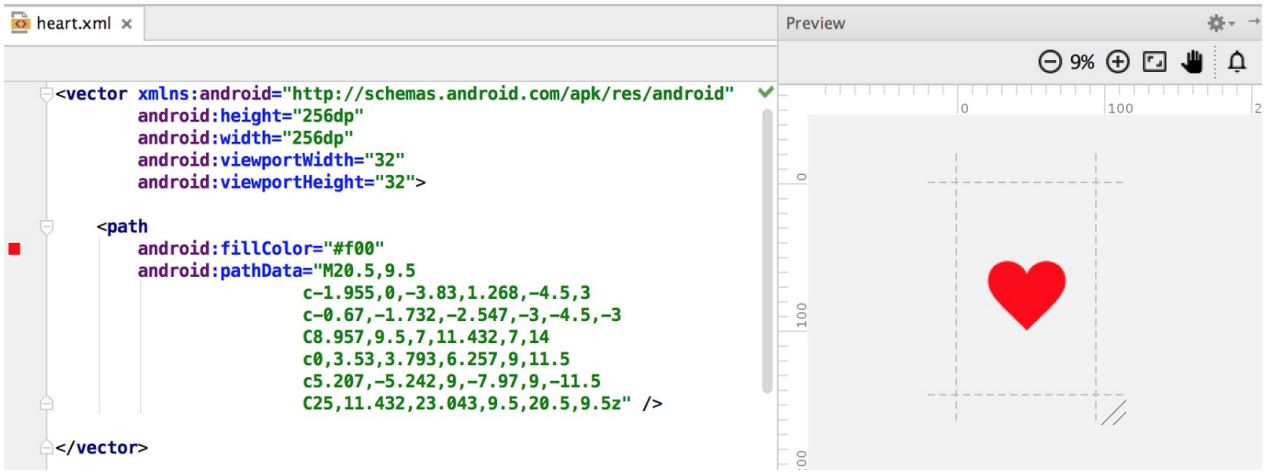
Untuk memasukkan gambar vektor atau ikon Desain Material yang ada ke dalam proyek Android Studio Anda sebagai sumber daya dapat digambar untuk vektor:

1. Klik-kanan pada folder res/drawable.
2. Pilih **New > Vector Asset**. Vector Asset Studio akan terbuka dan memandu Anda melalui proses tersebut.

Untuk membuat gambar vektor, definisikan detail bentuk di dalam elemen XML `<vector>` . Misalnya, kode berikut mendefinisikan bentuk hati dan mengisinya dengan warna merah (`#f00`):

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    <!-- intrinsic size of the drawable -->
    android:height="256dp"
    android:width="256dp"
    <!-- size of the virtual canvas -->
    android:viewportWidth="32"
    android:viewportHeight="32">

  <!-- draw a path -->
  <path android:fillColor="#f00"
      android:pathData="M20.5,9.5
                        c-1.955,0,-3.83,1.268,-4.5,3
                        c-0.67,-1.732,-2.547,-3,-4.5,-3
                        C8.957,9.5,7,11.432,7,14
                        c0,3.53,3.793,6.257,9,11.5
                        c5.207,-5.242,9,-7.97,9,-11.5
                        C25,11.432,23.043,9.5,20.5,9.5z" />
</vector>
```

Android Studio akan menampilkan pratinjau sumber daya dapat digambar untuk vektor, misalnya, inilah hasil pembuatan file XML yang dijelaskan di atas:



Jika Anda telah memiliki gambar dalam format SVG, ada sejumlah cara untuk mendapatkan informasi `pathData` gambar:

- Di Android Studio, klik-kanan pada folder sumber daya dapat digambar dan pilih **New > Vector Asset** untuk membuka alat (bantu) Vector Asset Studio. Gunakan alat (bantu) ini untuk mengimpor file SVG lokal.
- Gunakan alat (bantu) konversi file seperti svg2android.
- Buka gambar dalam editor teks, atau jika Anda menampilkan gambar di browser, tampilkan sumber laman. Cari informasi `d=` , yang setara dengan `pathData` dalam XML Anda.

Gambar vektor dinyatakan dalam Android sebagai objek `VectorDrawable` . Untuk detail tentang sintaks `pathData` , lihat referensi Jalur SVG. Untuk mempelajari cara menganimasikan properti sumber daya dapat digambar untuk vektor, lihat Animasikan Sumber Daya Dapat Digambar untuk Vektor.

Gambar

Gambar, dari ikon peluncur ke gambar spanduk, digunakan dalam banyak cara di Android. Setiap kasus penggunaan memiliki syarat berbeda untuk resolusi, skalabilitas, dan kesederhanaan gambar. Di bagian ini, Anda akan mempelajari beberapa macam cara untuk menghasilkan gambar dan menyertakannya dalam aplikasi.

Membuat ikon

Setiap aplikasi memerlukan setidaknya sebuah ikon peluncur, dan seringkali aplikasi menyertakan ikon untuk tindakan bilah aksi, notifikasi, dan kasus penggunaan lainnya.

Ada dua pendekatan untuk membuat ikon:

- Buat serangkaian gambar dari ikon yang sama dalam resolusi dan ukuran berbeda, sehingga ikon tampak sama di berbagai perangkat yang memiliki kepadatan layar berbeda. Anda bisa menggunakan Image Asset Studio untuk melakukannya.
- Gunakan sumber daya dapat digambar untuk vektor, yang menskalakan secara otomatis tanpa membuat gambar menjadi pecah atau kabur. Anda bisa menggunakan Vector Asset Studio untuk melakukannya.

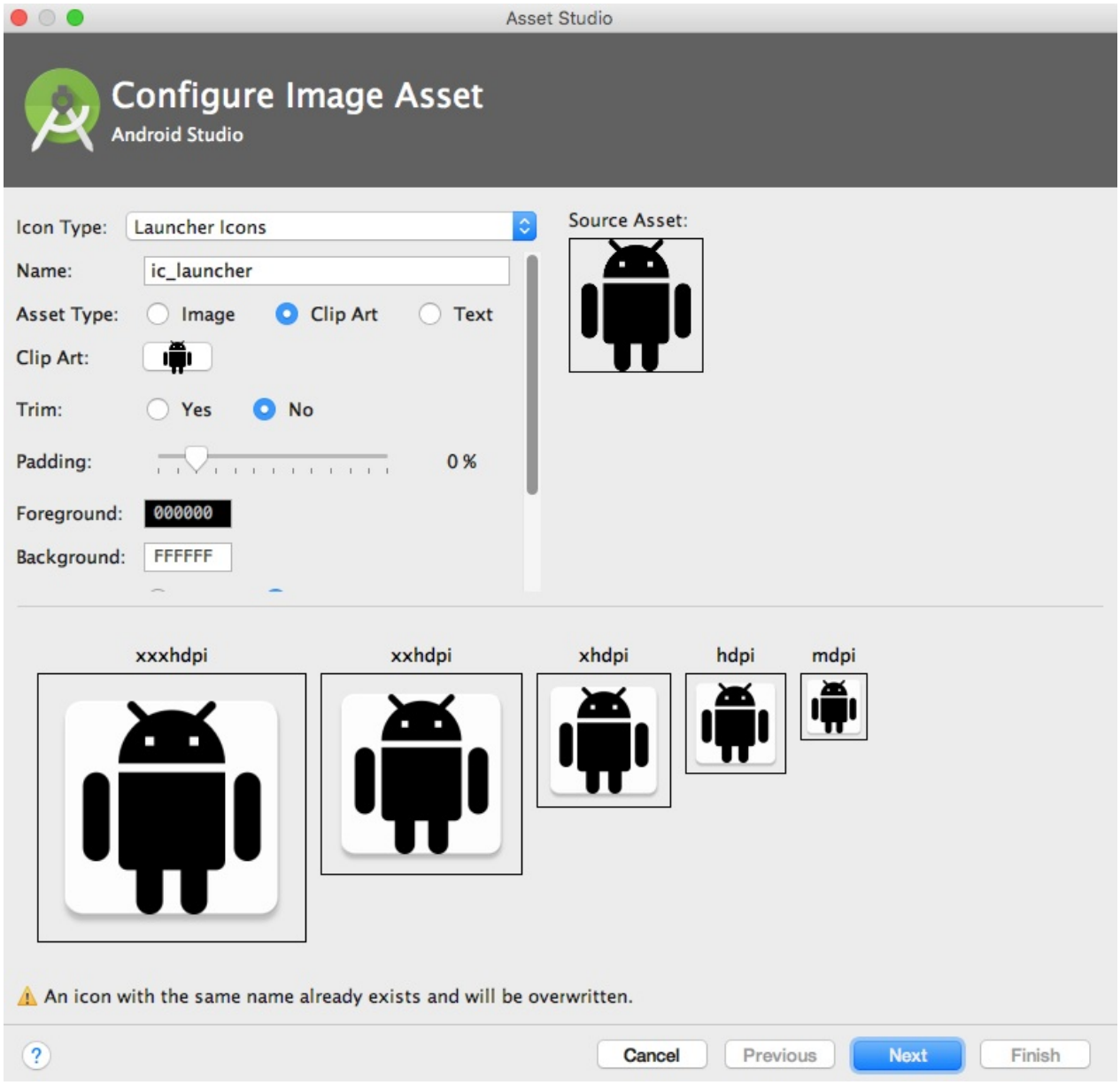
Image Asset Studio

Android Studio menyertakan alat (bantu) bernama Image Asset Studio yang membantu Anda membuat ikon aplikasi sendiri dari ikon Desain Material, gambar khusus, dan string teks. Alat ini menghasilkan serangkaian ikon dengan resolusi yang sesuai untuk setiap kepadatan layar umum yang didukung aplikasi Anda. Image Asset Studio menempatkan ikon yang baru dihasilkan dalam folder kepadatan-khusus pada folder res/ dalam proyek Anda. Pada waktu proses, Android menggunakan sumber daya yang sesuai berdasarkan kepadatan layar tempat menjalankan aplikasi Anda.

Image Asset Studio membantu Anda menghasilkan tipe ikon berikut:

- Ikon peluncur
- Bilah aksi dan ikon tab
- Ikon notifikasi

Untuk menggunakan Image Asset Studio, klik-kanan pada folder res/ di Android Studio dan pilih **New > Image Asset**. Wizard Configure Asset Studio akan terbuka dan memandu Anda melalui proses ini.



Untuk informasi selengkapnya tentang Image Asset Studio, lihat panduan Image Asset Studio.

Vector Asset Studio

Mulai dengan API 21, Anda bisa menggunakan sumber daya dapat digambar untuk vektor sebagai ganti file gambar untuk ikon.

Kelebihan menggunakan sumber daya dapat digambar untuk vektor sebagai ikon:

- Sumber daya dapat digambar untuk vektor bisa mengurangi ukuran file APK secara dramatis, karena Anda tidak perlu menyertakan banyak versi untuk setiap gambar ikon. Anda bisa menggunakan satu gambar vektor untuk menskalakan ke semua resolusi dengan mulus.
- Pengguna mungkin lebih suka mengunduh aplikasi yang memiliki file lebih kecil dan ukuran paket lebih kecil.

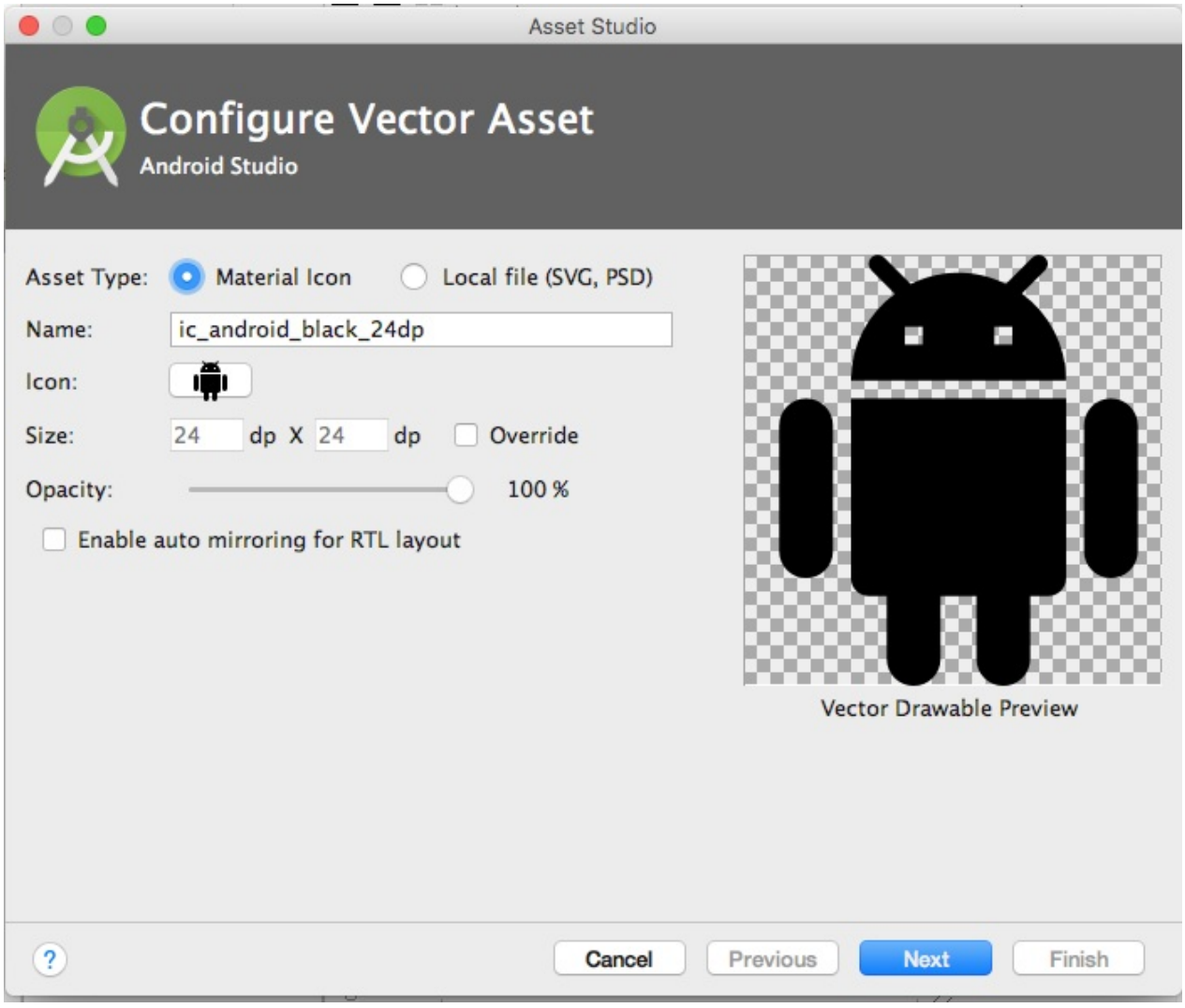
Kelemahan penggunaan sumber daya dapat digambar untuk vektor sebagai ikon:

- Sumber daya dapat digambar untuk vektor hanya bisa menyertakan detail dengan jumlah terbatas. Sumber daya dapat digambar untuk vektor sebagian besar digunakan untuk ikon yang tidak begitu detail seperti ikon Desain Material. Ikon dengan detail lebih banyak biasanya memerlukan beberapa file gambar.
- Sumber daya dapat digambar untuk vektor tidak didukung pada perangkat yang menjalankan API level 20 ke bawah.

Untuk menggunakan sumber daya dapat digambar untuk vektor pada perangkat yang menjalankan API level 20 ke bawah, Anda harus memutuskan antara dua metode kompatibilitas mundur:

- Secara default, pada waktu pembangunan, sistem membangun versi bitmap sumber daya dapat digambar untuk vektor Anda dalam resolusi yang berbeda. Hal ini memungkinkan ikon berjalan pada perangkat yang tidak bisa menggambar sumber daya dapat digambar untuk vektor.
- Kelas `VectorDrawableCompat` dalam Pustaka Dukungan Android memungkinkan Anda mendukung sumber daya dapat digambar untuk vektor dalam Android 2.1 (API level 7) dan yang lebih tinggi.

Vector Asset Studio adalah alat (bantu) yang membantu Anda menambahkan ikon Desain Material dan sumber daya dapat digambar untuk vektor ke proyek Android. Untuk menggunakannya, klik-kanan pada folder res/ di Android Studio dan pilih **New > Vector Asset**. Wizard Configure Asset Studio akan terbuka dan memandu Anda melalui proses ini.



Untuk informasi selengkapnya tentang menggunakan Vector Asset Studio dan dukungan kompatibilitas mundur, lihat panduan Vector Asset Studio.

Membuat gambar lainnya

Gambar spanduk, gambar profil pengguna, dan gambar lainnya dalam semua bentuk dan ukuran. Dalam banyak kasus, gambar tersebut lebih besar daripada seharusnya untuk antarmuka pengguna (UI) aplikasi pada umumnya. Misalnya, aplikasi Galeri sistem menampilkan foto yang diambil menggunakan kamera perangkat Android, dan foto ini umumnya

memiliki resolusi yang jauh lebih tinggi daripada kepadatan layar perangkat. Perangkat Android memiliki memori terbatas, sehingga idealnya Anda hanya perlu memuat foto versi resolusi rendah dalam memori. Versi resolusi rendah harus cocok dengan ukuran komponen UI yang menampilkannya. Gambar dengan resolusi lebih tinggi tidak memberikan manfaat jelas, namun tetap memerlukan memori yang berharga dan menambah overhead kinerja tambahan karena tambahan penskalaan sambil-jalan.

Anda bisa memuat gambar dengan ukuran yang diubah secara manual, namun sejumlah pustaka pihak ketiga telah dibuat untuk membantu memuat, menskalakan, dan meng-cache gambar.

Menggunakan pustaka pemuatan gambar

Pustaka pemuatan gambar seperti Glide dan Picasso bisa menangani pengaturan ukuran, penyimpanan ke cache, dan menampilkan gambar. Pustaka pihak ketiga ini dioptimalkan untuk seluler, dan didokumentasikan dengan baik.

Glide mendukung pengambilan, decoding, dan penampilan video diam, gambar, dan GIF animasi. Anda bisa menggunakan Glide untuk memuat gambar dari Web API, serta gambar yang ada dalam file sumber daya. Glide menyertakan fitur seperti pemuatan gambar placeholder (untuk memuat gambar yang lebih detail), animasi memudar bersilang, dan penyimpanan ke cache secara otomatis.

Untuk menggunakan Glide:

1. Unduh pustaka.
2. Sertakan dependensi di aplikasi Anda dalam file app-level build.gradle, dengan mengganti `<em>n.n.n</em>` dengan Glide versi terbaru: `compile 'com.github.bumptech.glide:glide:<em>n.n.n</em>'`

Anda bisa menggunakan Glide untuk memuat gambar ke dalam elemen UI. Contoh berikut memuat gambar dari URL ke dalam sebuah `ImageView` :

```
ImageView imageView = (ImageView) findViewById(R.id.my_image_view);
Glide.with(this).load("URL").into(imageView);
```

Dalam cuplikan kode, `this` lihat konteks aplikasi. Ganti "URL" dengan URL lokasi gambar. Secara default, gambar disimpan dalam cache lokal dan diakses dari sana bila nanti dipanggil.

- Untuk contoh selengkapnya, lihat wiki Glide.
- Untuk informasi selengkapnya tentang Glide, lihat dokumentasi Glide dan tag [glide] pada stack overflow.
- Untuk informasi selengkapnya tentang Picasso, lihat dokumentasi Picasso dan tag [picasso] pada stack overflow.
- Untuk membandingkan fitur pustaka yang berbeda, telusuri di stack overflow, misalnya Glide vs. Picasso.

Menguji rendering gambar

Gambar dirender berbeda pada perangkat yang berbeda. Agar tidak terkejut, gunakan Android Virtual Device (AVD) Manager untuk membuat perangkat virtual yang menyimulasikan layar dengan ukuran dan kepadatan berbeda. Gunakan AVD untuk menguji semua gambar Anda.

Mempercepat aplikasi Anda

Mengambil dan meng-cache gambar

Saat mengambil gambar, aplikasi bisa menggunakan banyak data. Untuk menghemat data, pastikan permintaan Anda mulai dari sekecil mungkin. Definisikan dan simpan gambar yang belum disesuaikan ukurannya pada sisi server, kemudian minta gambar yang sudah disesuaikan ukurannya dengan Tampilan.

Simpan gambar Anda ke cache sehingga setiap gambar hanya perlu melewati jaringan sekali. Bila gambar diminta, terlebih dahulu periksa cache Anda. Mintalah gambar melalui jaringan saja jika gambar tidak ada dalam cache. Gunakan pustaka pemuatan gambar seperti Glide atau Picasso untuk menangani caching. Pustaka ini juga mengelola ukuran cache,

membuang gambar lama atau yang tidak digunakan. Untuk informasi selengkapnya tentang pustaka, lihat bagian pustaka pada bab ini.

Untuk memaksimalkan kinerja dalam konteks berbeda, setel aturan bersyarat untuk cara aplikasi menangani gambar, dengan bergantung pada tipe koneksi dan stabilitas. Gunakan `ConnectivityManager` untuk menentukan tipe koneksi dan status, kemudian setel aturan bersyarat yang sesuai. Misalnya, bila pengguna sedang menggunakan koneksi data (bukan WiFi), turunkan resolusi gambar yang diminta di bawah resolusi layar. Tingkatkan lagi resolusi layar yang diminta saat pengguna menggunakan WiFi.

Saat aplikasi Anda mengambil gambar melalui jaringan, koneksi lambat akan membuat pengguna menunggu. Inilah cara membuat aplikasi Anda terasa cepat, walaupun gambar dimuat dengan lambat:

- Prioritaskan gambar yang lebih penting sehingga dimuat lebih dulu. Pustaka seperti Glide dan Picasso memungkinkan Anda meminta berdasarkan prioritas gambar.
- Prioritaskan permintaan teks sebelum permintaan gambar. Jika aplikasi Anda dapat digunakan tanpa gambar, misalnya jika berupa aplikasi umpan berita, membiarkan pengguna menggulir gambar bisa membuat aplikasi fungsional dan bahkan mungkin merender permintaan gambar usang.
- Tampilkan warna placeholder sambil mengambil gambar.

Jika menampilkan warna placeholder, Anda mungkin ingin penampilan aplikasi tetap konsisten di saat aplikasi memuat gambar. Gunakan pustaka Palette untuk memilih warna placeholder berdasarkan keseimbangan warna gambar yang diminta. Pertama, sertakan pustaka Palette dalam file build.gradle Anda.

```
dependencies: {
    compile 'com.android.support:palette-v7:24.2.1'
}
```

Tarik warna dominan untuk gambar yang Anda inginkan dan setel sebagai warna latar belakang dalam `ImageView`. Jika Anda mengambil gambar menggunakan pustaka, letakkan kode berikut setelah mendefinisikan URL yang akan dimuat ke dalam `ImageView`:

```
Palette palette = Palette.from(tiles).generate(new PaletteAsyncListener(){
   Public void onGenerated(Pallet pallette) {
      Palette.Swatch background = palette.getDominantSwatch();
         if (background != null) {
            ImageView.setBackgroundColor(background.getRgb());
          }
     }
}
```

Menyajikan gambar melalui jaringan

Untuk menghemat bandwidth dan agar aplikasi tetap berjalan cepat, gunakan format WebP untuk menyajikan dan mengirim gambar.

Cara lain untuk menghemat bandwith adalah dengan menyajikan dan meng-cache gambar berukuran khusus. Caranya, izinkan klien menetapkan resolusi dan ukuran yang diperlukan untuk perangkat dan Tampilan mereka, kemudian buat dan cache gambar yang diperlukan pada sisi server sebelum Anda mengirimkannya.

Misalnya, laman landas umpan berita mungkin hanya meminta gambar kecil. Sebagai ganti mengirim gambar berukuran penuh, kirim saja gambar kecil yang ditetapkan oleh `ImageView`. Anda bisa mengurangi ukuran gambar kecil lebih jauh dengan memproduksi gambar pada resolusi yang berbeda.

Tip: Gunakan metode `Activity.isLowRamDevice()` untuk mengetahui apakah suatu perangkat mendefinisikan dirinya sebagai "minim RAM". Jika metode mengembalikan `true`, kirim gambar resolusi rendah sehingga aplikasi Anda menggunakan memori perangkat lebih sedikit.

Gaya

Di Android, *gaya* adalah kumpulan atribut yang mendefinisikan tampilan dan format Tampilan. Anda bisa menerapkan gaya yang sama ke sejumlah Tampilan dalam aplikasi; misalnya, sejumlah TextView mungkin memiliki ukuran teks dan layout yang sama. Penggunaan gaya memungkinkan Anda menyimpan atribut umum ini di satu lokasi dan menerapkannya ke setiap TextView dengan menggunakan satu baris kode dalam XML.

Anda bisa mendefinisikan gaya sendiri atau menggunakan salah satu dari gaya platform yang disediakan Android.

Mendefinisikan dan menerapkan gaya

Untuk membuat gaya, tambahkan elemen `<style>` di dalam elemen `<resources>` dalam file XML yang terletak di folder res/values/. Bila membuat proyek di Android Studio, file res/values/styles.xml akan dibuatkan untuk Anda.

Elemen `<style>` menyertakan yang berikut ini:

- Atribut `name` . Gunakan nama gaya bila Anda menerapkan gaya ke Tampilan.
- Atribut `parent` opsional. Anda akan mempelajari cara menggunakan atribut `parent` di bagian Pewarisan di bawah ini.
- Sejumlah elemen `<item>` sebagai elemen anak `<style>` . Setiap `<item>` menyertakan satu atribut gaya.

Contoh ini membuat gaya yang memformat teks untuk menggunakan jenis huruf spasi tunggal abu-abu muda, sehingga tampak seperti kode:

```
<resources>
    <style name="CodeFont">
        <item name="android:typeface">monospace</item>
        <item name="android:textColor">#D7D6D7</item>
    </style>
</resources>
```

XML berikut menerapkan gaya `CodeFont` baru ke Tampilan:

```
<TextView
    style="@style/CodeFont"
    android:text="@string/code_string" />
```

Pewarisan

Gaya baru bisa mewarisi properti gaya yang ada. Bil Anda membuat gaya yang mewarisi properti, cukup definisikan properti yang ingin diubah atau ditambahkan. Anda bisa mewarisi properti dari gaya platform dan dari gaya yang dibuat sendiri. **Untuk mewarisi gaya platform,** gunakan atribut `parent` untuk menetapkan ID sumber daya gaya yang ingin Anda warisi. Misalnya, inilah cara mewarisi penampilan teks default platform Android (gaya `TextAppearance`) dan mengubah warnanya:

```
<style name="GreenText" parent="@android:style/TextAppearance">
    <item name="android:textColor">#00FF00</item>
</style>
```

Untuk menerapkan gaya ini, gunakan `@style/GreenText` . **Untuk mewarisi gaya yang dibuat sendiri,** gunakan nama gaya yang ingin Anda warisi sebagai bagian pertama pada nama gaya baru, dan pisahkan bagian tersebut dengan titik:

```
 name="StyleToInherit.Qualifier"
```

Misalnya, untuk membuat gaya yang mewarisi gaya `CodeFont` yang didefinisikan di atas, gunakan `CodeFont` sebagai bagian pertama pada nama gaya baru:

```
<style name="CodeFont.RedLarge">
    <item name="android:textColor">#FF0000</item>
    <item name="android:textSize">34sp</item>
</style>
```

Contoh ini menyertakan atribut `typeface` dari gaya `CodeFont` asli, menggantikan atribut `textColor` asal dengan merah, dan menambahkan atribut baru, `textSize`. Untuk menerapkan gaya ini, gunakan `@style/CodeFont.RedLarge`.

Tema

Anda bisa membuat tema dengan cara yang sama dengan membuat gaya, yaitu dengan menambahkan elemen `<style>` di dalam elemen `<resources>` dalam file XML yang terletak di folder res/values/.

Apakah perbedaan antara gaya dan tema?

- *Gaya* diterapkan pada Tampilan. Di XML, Anda menerapkan gaya menggunakan atribut `style`.
- *Tema* diterapkan ke semua Aktivitas atau aplikasi, bukan ke Tampilan individual. Di XML, Anda bisa menerapkan sebuah tema mengggunakan atribut `android:theme`.

Gaya apa pun bisa digunakan sebagai tema. Misalnya, Anda bisa menerapkan gaya `CodeFont` sebagai tema Aktivitas, dan semua teks di dalam Aktivitas akan menggunakan font spasi tunggal abu-abu.

Menerapkan tema

Untuk menerapkan tema ke aplikasi Anda, deklarasikan tema dalam elemen `<application>` di dalam file AndroidManifest.xml. Contoh ini menerapkan tema `AppTheme` ke seluruh aplikasi:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.exampledomain.myapp">
    <application
        ...
        android:theme="@style/AppTheme">
    </application>
    ...
```

Untuk menerapkan tema ke Aktivitas, deklarasikan tema dalam elemen `<activity>` dalam file AndroidManifest.xml. Dalam contoh ini, atribut `android:theme` menerapkan tema platform Theme_Dialog ke Aktivitas:

```
<activity android:theme="@android:style/Theme.Dialog">
```

Tema default

Bila membuat proyek baru di Android Studio, tema default akan didefinisikan untuk Anda dalam file style.xml. Misalnya, kode ini mungkin ada dalam file styles.xml Anda:

```
<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>
```

Dalam contoh ini, `AppTheme` mewarisi dari `Theme.AppCompat.Light.DarkActionBar`, yang merupakan salah satu dari banyak tema platform Android yang tersedia untuk Anda. (Anda akan mempelajari tentang atribut warna dalam unit di Desain Material.)

Gaya dan tema platform

Platform Android menyediakan kumpulan gaya dan tema yang bisa Anda gunakan dalam aplikasi. Untuk menemukan daftar semua tema, Anda perlu melihat di dua tempat:

- Kelas R.style mencantumkan sebagian besar gaya dan tema platform yang tersedia.
- Kelas support.v7.appcompat.R.style mencantumkan lebih banyak lagi. Gaya dan tema ini memiliki " `AppCompat` " dalam namanya, dan didukung oleh pustaka v7 appcompat.

Nama gaya dan tema menyertakan setrip bawah. Untuk menggunakannya dalam kode Anda, ganti setrip bawah dengan titik. Misalnya, inilah cara menerapkan tema Theme_NoTitleBar ke aktivitas:

```
<activity android:theme="@android:style/Theme.NoTitleBar"
```

Dan inilah cara menerapkan gaya AlertDialog_AppCompat ke Tampilan:

```
<TextView
    style="@style/AlertDialog.AppCompat"
    android:text="@string/code_string" />
```

Dokumentasi tidak menjelaskan semua gaya dan tema secara detail, namun Anda bisa menyimpulkan dari namanya. Misalnya, dalam `Theme.AppCompat.Light.DarkActionBar`

- "Theme" menunjukkan bahwa gaya ini dimaksudkan untuk digunakan sebagai tema.
- "AppCompat" menunjukkan bahwa tema ini didukung oleh pustaka v7 appcompat.
- "Light" menunjukkan bahwa tema terdiri dari latar belakang terang, yaitu putih secara default. Semua warna teks dalam tema ini adalah gelap, agar kontras dengan latar belakang yang terang. (Jika menginginkan latar belakang gelap dan teks terang, tema Anda bisa mewarisi dari tema seperti `Theme.AppCompat` tanpa "Light" dalam namanya.)
- "DarkActionBar" menunjukkan bahwa warna gelap digunakan untuk bilah aksi, sehingga semua teks atau ikon dalam bilah aksi berwarna terang.

Tema berguna lainnya adalah `Theme.AppCompat.DayNight` , yang memungkinkan pengguna menjelajahi dalam "mode malam" dengan kontras rendah pada malam hari. Tema ini secara otomatis mengubah tema dari `Theme.AppCompat.Light` menjadi `Theme.AppCompat` , berdasarkan waktu. Untuk mengetahui selengkapnya tentang tema `DayNight` , baca entri blog Chris Banes.

Untuk mengetahui selengkapnya tentang penggunaan gaya dan tema platform, kunjungi panduan gaya dan tema.

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Sumber Daya Dapat Digambar, Gaya, dan Tema

Ketahui selengkapnya

- Panduan sumber daya sumber daya dapat digambar
- Panduan Image Asset Studio
- Panduan Vector Asset Studio
- Android Asset Studio oleh Roman Nurik
- Panduan gaya dan tema

5.2: Desain Material

Daftar Isi:

Desain Material adalah filosofi desain visual yang dibuat Google tahun 2014. Tujuan Desain Material adalah pengalaman pengguna terpadu lintas platform dan ukuran perangkat. Desain Material menyertakan serangkaian panduan untuk gaya, layout, gerakan, dan aspek desain aplikasi lainnya. Panduan lengkap tersedia di Spesifikasi Desain Material.

Desain Material adalah untuk aplikasi web desktop serta aplikasi seluler. Bab ini memfokuskan pada Desain Material untuk aplikasi seluler di Android.

Prinsip Desain Material

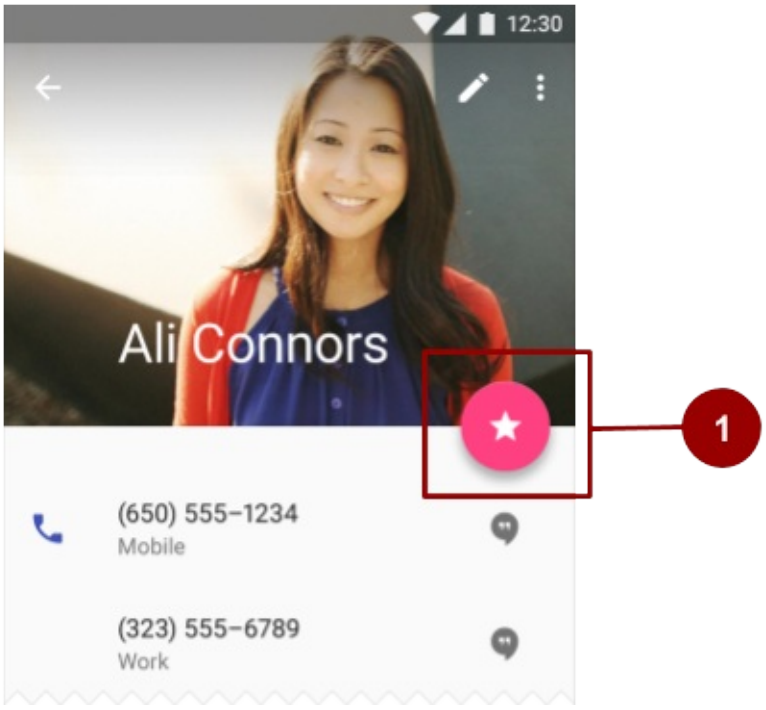
Metafora "material"

Dalam Desain Material, elemen dalam aplikasi Android Anda berperilaku seperti material sungguhan: mentransmisikan bayangan, menempati ruang, dan saling berinteraksi.

Menyolok, grafik, intensional

Desain Material melibatkan pilihan warna yang disengaja, citra detail, tipografi skala besar, dan ruang putih intensional yang menghasilkan antarmuka yang menyolok dan grafik.

Beri penekanan pada tindakan pengguna dalam aplikasi Anda, sehingga pengguna langsung tahu apa yang harus dilakukan, dan cara melakukannya. Misalnya, sorot hal-hal yang bisa berinteraksi dengan pengguna, seperti tombol, bidang EditText, dan switch.



1. Dalam layout ini, tombol aksi mengambang disorot dengan warna aksen merah muda.

Gerakan bermakna

Buat animasi dan gerakan lain dalam aplikasi Anda menjadi bermakna, sehingga gerakan tidak terjadi secara acak. Gunakan gerakan untuk memperkuat ide bahwa pengguna adalah penggerak utama aplikasi. Misalnya, desain aplikasi Anda sehingga sebagian besar gerakan diinisiasi oleh tindakan pengguna, bukan oleh kejadian di luar kontrol pengguna. Anda juga bisa menggunakan gerakan untuk memfokuskan perhatian pengguna, memberi masukan yang halus bagi pengguna, atau menyoroti elemen aplikasi Anda.

Bila aplikasi Anda menghadirkan suatu objek kepada pengguna, pastikan gerakannya tidak memotong keberlangsungan pengalaman pengguna. Misalnya, pengguna seharusnya tidak perlu menunggu animasi atau transisi selesai.

Bagian Gerakan dalam bab ini menjelaskan detail selengkapnya tentang cara menggunakan gerakan dalam aplikasi Anda.

Warna

Palet warna Desain Material

Prinsip Desain Material menyertakan penggunaan warna menyolok. Palet warna Desain Material berisi warna-warna yang bisa dipilih, setiap dengan warna primer dan bayangan yang diberi label dari 50 hingga 900:

- Pilih warna dengan label "500" sebagai warna primer merek Anda. Gunakan warna tersebut dan bayangan warna itu dalam aplikasi Anda.
- Pilih warna kontras sebagai warna aksen dan gunakan untuk membuat sorotan dalam aplikasi Anda. Pilih warna apa pun yang dimulai dengan "A".

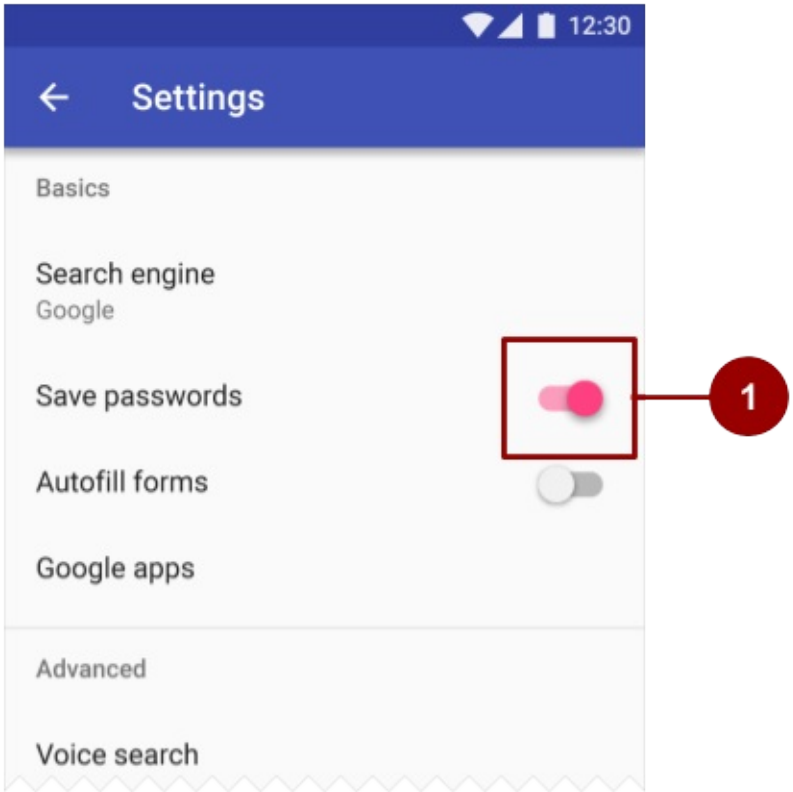
Bila membuat proyek Android dalam Android Studio, contoh skema warna Desain Material akan dipilihkan untuk Anda dan diterapkan ke tema. Dalam values/colors.xml, tiga elemen `<color>` didefinisikan, `colorPrimary`, `colorPrimaryDark`, dan `colorAccent`:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <!-- Indigo. -->
    <color name="colorPrimaryDark">#303F9F</color>
    <!-- A darker shade of indigo. -->
    <color name="colorAccent">#FF4081</color>
    <!-- A shade of pink. -->
</resources>
```

Dalam values/styles.xml, tiga warna yang telah didefinisikan diterapkan ke tema default, yang menerapkan warna ke beberapa elemen aplikasi secara default:

- `colorPrimary` digunakan oleh sejumlah Tampilan secara default. Misalnya, dalam tema `AppTheme` , `colorPrimary` digunakan sebagai warna latar belakang untuk bilah aksi. Ubah nilai ini ke "500" yang dipilih sebagai warna primer merek Anda.
- `colorPrimaryDark` digunakan dalam area yang memerlukan sedikit kontras dengan warna primer, misalnya bilah status. Setel nilai ini ke versi yang sedikit lebih gelap dari warna primer Anda.
- `colorAccent` digunakan sebagai warna sorotan untuk sejumlah Tampilan. Ini juga digunakan untuk switch dalam posisi "aktif", tombol aksi mengambang, dan lainnya.

Dalam tangkapan layar di bawah ini, latar belakang bilah aksi menggunakan `colorPrimary` (indigo), bilah status menggunakan `colorPrimaryDark` (bayangan indigo yang lebih gelap), dan switch dalam posisi "aktif" menggunakan



`colorAccent` (warna merah muda).

1. Dalam layout ini, switch pada posisi "hidup" disorot dengan warna aksen merah muda.

Sebagai rangkuman, inilah cara menggunakan palet warna Desain Material dalam aplikasi Android Anda:

1. Pilih warna primer untuk aplikasi dari palet warna Desain Material dan salin nilai heksanya ke dalam item `colorPrimary` dalam colors.xml.
2. Pilih tingkatan warna yang lebih gelap dari warna ini dan salin nilai heksanya ke dalam item `colorPrimaryDark` .
3. Pilih warna aksen dari tingkatan warna yang dimulai dengan "A" dan salin nilai heksanya ke dalam item colorAccent.

4. Jika Anda memerlukan lebih banyak warna, buat elemen `<color>` tambahan dalam file colors.xml. Misalnya, Anda bisa memilih versi indigo yang lebih terang dan buat elemen `<color>` tambahan bernama `colorPrimaryLight`. (Namanya terserah Anda.)

   ```
   <color name="colorPrimaryLight">#9FA8DA</color>
   <!-- A lighter shade of indigo. -->
   ```

   Untuk menggunakan warna ini, referensikan sebagai `@color/colorPrimaryLight`.

Perubahan nilai dalam colors.xml secara otomatis mengubah warna Tampilan dalam aplikasi Anda, karena warna diterapkan ke tema dalam styles.xml.

Kontras

Pastikan semua teks dalam UI aplikasi Anda kontras dengan latar belakangnya. Di tempat yang berlatar belakang gelap, berikan warna terang untuk teks di atasnya, dan sebaliknya. Kontras semacam ini penting untuk keterbacaan dan aksesibilitas, karena tidak semua orang melihat warna dengan cara yang sama.

Jika menggunakan tema platform seperti `Theme.AppCompat`, Anda yang akan menangani kontras antara teks dan latar belakangnya. Misalnya:

- Jika tema Anda mewarisi dari `Theme.AppCompat`, sistem menganggap Anda menggunakan latar belakang gelap. Karena itu, semua teks mendekati putih secara default.
- Jika tema Anda mewarisi dari `Theme.AppCompat.Light`, teks mendekati hitam, karena tema memiliki latar belakang terang.
- Jika Anda menggunakan tema `Theme.AppCompat.Light.DarkActionBar`, teks dalam bilah aksi mendekati putih, agar kontras dengan latar belakang gelap bilah aksi. Teks selebihnya dalam aplikasi mendekati hitam, agar kontras dengan latar belakang terang.

Gunakan kontras warna untuk menciptakan pemisahan visual di antara elemen aplikasi Anda. Gunakan warna `colorAccent` Anda untuk menarik perhatian pada elemen UI utama seperti tombol aksi mengambang dan switch dalam posisi "aktif".

Opasitas

Aplikasi Anda bisa menampilkan teks dengan derajat opasitas berbeda untuk menyatakan relativitas kepentingan informasi. Misalnya, teks yang kurang penting mungkin hampir transparan (opasitas rendah).

Setel atribut `android:textColor` menggunakan salah satu format ini: `"#rgb"`, `"#rrggbb"`, `"#argb"`, atau `"#aarrggbb"`. Untuk menyetel opasitas teks, gunakan format `"#argb"` atau `"#aarrggbb"` dan sertakan nilai untuk *saluran alfa*. Saluran alfa adalah `a` atau `aa` pada awal nilai `textColor`.

Nilai opasitas maksimum, `FF` dalam heksa, akan membuat warna buram sepenuhnya. Nilai minimum, `00` dalam heksa, akan membuat warna transparan sepenuhnya.

Untuk menentukan jumlah heksa yang digunakan dalam saluran alfa:

1. Putuskan tingkat opasitas yang ingin Anda gunakan, dalam persentase. Tingkat opasitas yang digunakan untuk teks bergantung pada apakah latar belakang Anda terang atau gelap. Untuk mengetahui tingkat opasitas yang digunakan dalam situasi berbeda, lihat Porsi warna teks pada panduan Desain Material.
2. Lipat gandakan persentase tersebut, sebagai nilai desimal, hingga 255. Misalnya, jika Anda memerlukan teks utama yang 87% buram, lipat gandakan menjadi 0,87 x 255. Hasilnya adalah 221,85.
3. Bulatkan hasil ke bilangan bulat terdekat: 222.
4. Gunakan konverter heksa untuk mengonversikan hasil menjadi heksa: `DE`. Jika hasilnya nilai tunggal, awali dengan `0`.

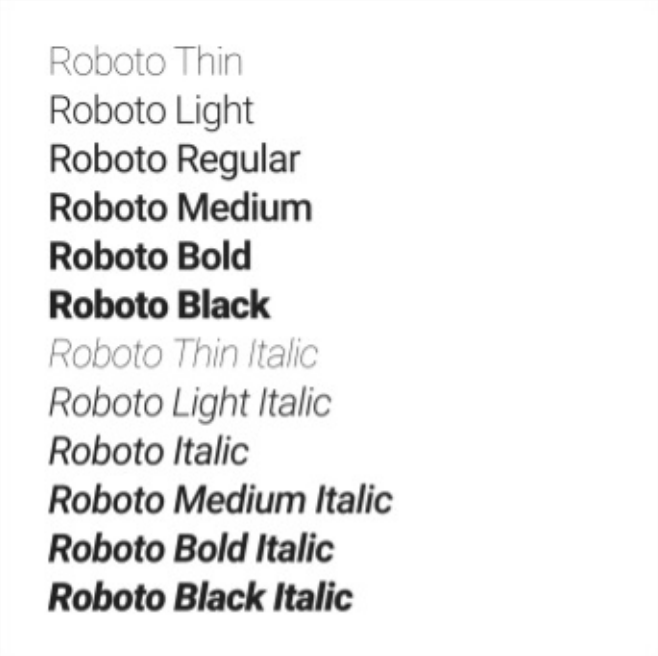
Dalam kode XML berikut, latar belakang teks adalah gelap, dan warna teks utama adalah 87% putih (`deffffff`). Dua angka pertama kode warna (`de`) menunjukkan opasitas.

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Hello World!"
   android:textSize="45dp"
   android:background="@color/colorPrimaryDark"
   android:textColor="#defffff"/>
```

Tipografi

Jenis huruf

Roboto adalah jenis huruf Desain Material standar di Android. Roboto memiliki enam bobot: Thin, Light, Regular, Medium,



Bold, dan Black.

Gaya font

Platform Android menyediakan gaya dan ukuran font yang telah didefinisikan sebelumnya, yang bisa Anda gunakan dalam aplikasi. Gaya dan ukuran ini dikembangkan untuk mengimbangi kepadatan materi dan kenyamanan membaca dalam ketentuan umum. Ukuran ditetapkan dengan sp (piksel yang bisa diskalakan) agar memungkinkan mode ketik besar untuk aksesibilitas.

Berhati-hatilah agar tidak menggunakan terlalu banyak ukuran sekaligus gaya berbeda dalam layout Anda.

Untuk menggunakan salah satu gaya yang telah didefinisikan sebelumnya ini dalam Tampilan, setel atribut `android:textAppearance` . Atribut ini mendefinisikan penampilan default teks: warna, jenis huruf, ukuran, dan gaya. Gunakan gaya kompatibel-mundur `TextAppearance.AppCompat` .

Misalnya, untuk membuat TextView muncul dalam gaya Display 3, tambahkan atribut berikut ke TextView dalam XML:

```
android:textAppearance="@style/TextAppearance.AppCompat.Display3"
```

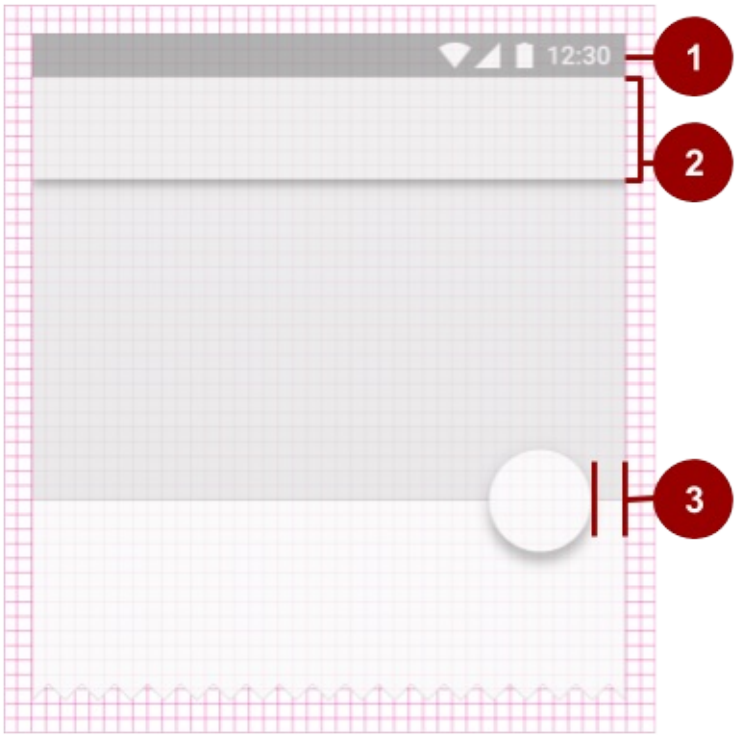
Untuk informasi selengkapnya mengenai penataan gaya teks, lihat Tipografi panduan Desain Material.

Layout

Metrik dan keyline

Komponen dalam template Desain Material yang ditujukan untuk perangkat seluler, tablet, dan desktop sejajar dengan petak kotak 8 dp. *Dp* adalah piksel yang tidak tergantung kepadatan, unit abstrak berdasarkan kepadatan layar. Dp serupa dengan sp, namun sp juga diskalakan oleh pilihan ukuran font pengguna. Itu sebabnya sp lebih disukai untuk aksesibilitas. Untuk informasi selengkapnya tentang unit pengukuran, lihat unit Layout, Tampilan, dan Sumber Daya.

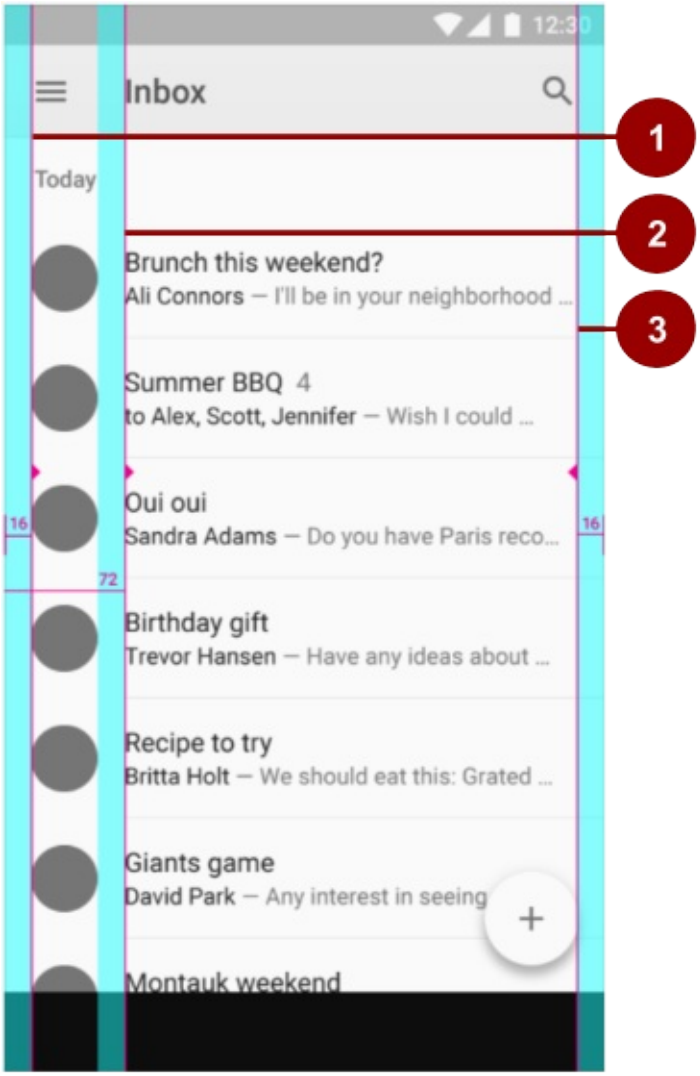
Petak kotak 8 dp memandu penempatan elemen dalam layout Anda. Setap kotak dalam petak berukuran 8 dp x 8 dp, sehingga tinggi dan lebar tiap elemen dalam layout adalah kelipatan 8 dp.



1. Bilah status dalam layout adalah setinggi 24dp, setara ketinggian kotak tiga petak.
2. Bilah alat setinggi 56 dp, setara ketinggian kotak tujuh petak.
3. Salah satu batas materi sebelah kanan adalah 16 dp dari tepi layar, setara lebar kotak dua persegi.

Ikonografi dalam bilah alat sejajar dengan kotak petak 4 dp sebagai ganti kotak persegi 8 dp, sehingga dimensi ikon dalam bilah alat adalah kelipatan 4 dp.

Keyline adalah outline dalam petak layout yang menentukan penempatan teks dan ikon. Misalnya, keyline menandai tepi margin dalam layout.



1. Keyline yang menampilkan margin kiri tepi layar, dalam hal ini adalah 16 dp.
2. Keyline yang menampilkan margin kiri materi terkait dengan ikon atau avatar, 72 dp.
3. Keyline yang menampilkan margin kanan tepi layar, 16 dp.

Tipografi Desain Material selaras dengan *petak patokan* 4 dp, yaitu petak yang terbuat dari garis-garis horizontal saja.

Panduan Desain Material menyediakan template yang bisa diunduh untuk layar UI yang umum digunakan. Untuk mengetahui selengkapnya tentang metrik dan keyline dalam Desain Material, kunjungi panduan metrik dan keyline.

Komponen dan pola

Tombol dan banyak Tampilan lainnya yang digunakan di Android secara default mematuhi prinsip Desain Material. Panduan Desain Material menyertakan *komponen dan pola* yang bisa Anda bangun untuk membantu pengguna memahami cara kerja elemen dalam UI, bahkan jika pengguna tersebut baru menggunakan aplikasi.

Gunakan Desain Material komponen untuk panduan spesifikasi dan perilaku tombol, chip, kartu, dan banyak lagi elemen UI lainnya. Gunakan pola Desain Material untuk panduan cara memformat tanggal dan waktu, isyarat, panel samping navigasi, dan banyak lagi aspek UI lainnya.

Bagian ini mengajarkan tentang Pustaka Dukungan Desain dan beberapa komponen serta pola yang tersedia untuk Anda. Untuk dokumentasi lengkap tentang komponen dan pola yang bisa Anda gunakan, lihat panduan Desain Material

Pustaka Dukungan Desain

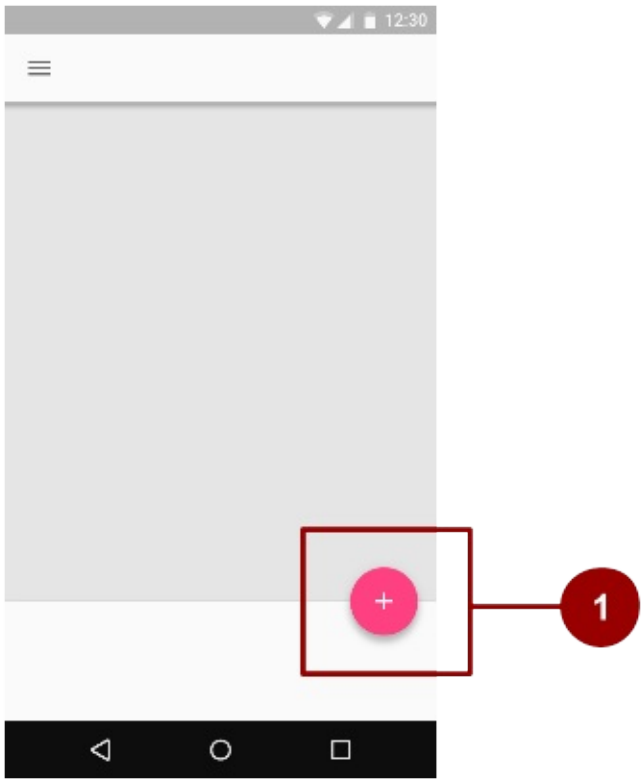
Paket Desain menyediakan API untuk mendukung penambahan komponen Desain Material dan pola ke aplikasi Anda. Pustaka Dukungan Desain menambahkan dukungan untuk beragam komponen dan pola Desain Material yang akan Anda bangun. Untuk menggunakan pustaka, sertakan dependensi berikut dalam file build.gradle Anda:

```
compile 'com.android.support:design:25.0.1'
```

Untuk memastikan Anda memiliki nomor versi terbaru Pustaka Dukungan Desain, periksa laman Pustaka Dukungan.

Tombol aksi mengambang (FAB)

Gunakan *tombol aksi mengambang (FAB)* bagi tindakan yang ingin Anda sarankan untuk dipakai pengguna. FAB adalah ikon lingkaran yang mengambang "di atas" UI. Saat difokus, warnanya sedikit berubah, dan tampak terangkat bila dipilih.



Saat diketuk, FAB bisa berisi tindakan terkait.

1. FAB berukuran normal

Untuk mengimplementasikan FAB, gunakan widget `FloatingActionButton` dan setel atribut FAB dalam XML layout Anda. Misalnya:

```
<android.support.design.widget.FloatingActionButton
    android:id="@+id/addNewItemFAB"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_plus_sign"
    app:fabSize="normal"
    app:elevation="10%" />
```

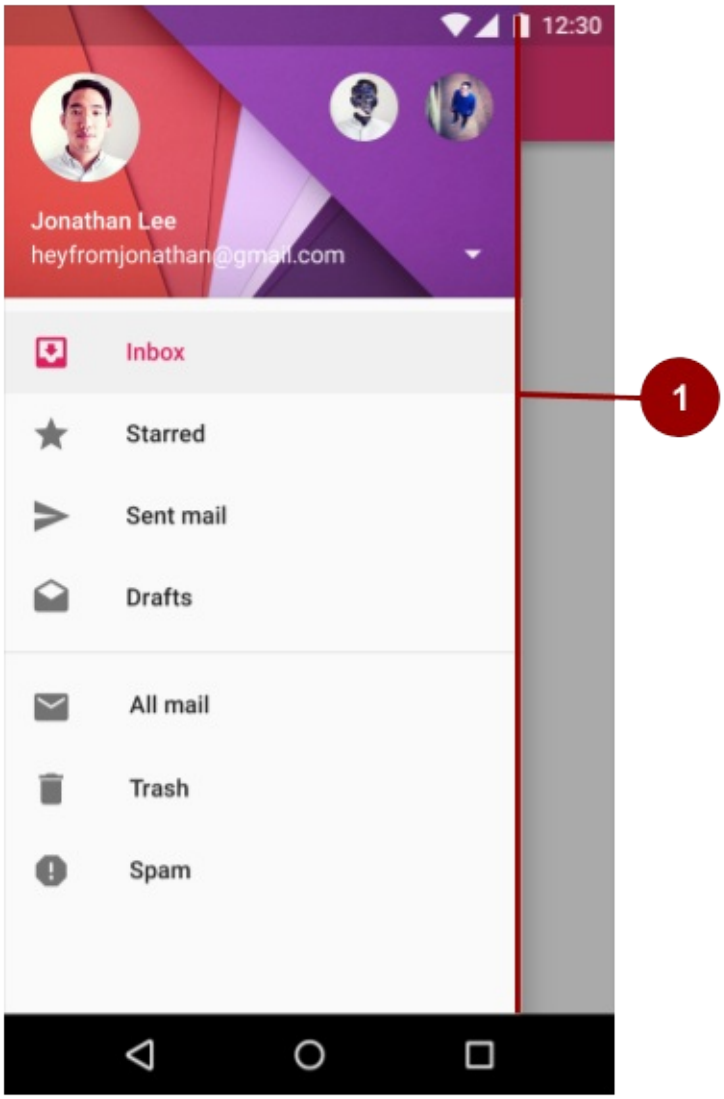
Atribut `fabSize` menyetel ukuran FAB. Ukurannya bisa `"normal"` (56 dp), `"mini"` (40 dp), atau `"auto"`, dengan perubahan yang berdasarkan ukuran jendela.

Elevasi FAB adalah jarak antara permukaan dan kedalaman bayangannya. Anda bisa menyetel atribut `elevation` sebagai referensi ke sumber daya, string, Boolean, atau sejumlah cara lainnya.

Untuk mengetahui tentang semua atribut yang bisa Anda setel untuk FAB termasuk `clickable`, `rippleColor`, dan `backgroundTint`, lihat FloatingActionButton. Untuk memastikan Anda menggunakan FAB sebagaimana dimaksud, periksa informasi penggunaan dalam panduan Desain Material yang ekstensif.

Panel samping navigasi

Panel samping navigasi adalah panel yang bergeser masuk dari kiri dan berisi tujuan navigasi untuk aplikasi Anda. Panel samping navigasi membentang setinggi layar, dan semua di belakangnya terlihat, namun gelap.



1. Panel samping navigasi yang "terbuka"

Untuk mengimplementasikan panel samping navigasi, gunakan `DrawerLayout` API yang tersedia dalam Pustaka Dukungan.

Dalam XML Anda, gunakan objek `DrawerLayout` sebagai tampilan akar layout. Di dalamnya, tambahkan dua tampilan, satu untuk layout utama saat panel samping disembunyikan, dan satu untuk materi panel samping.

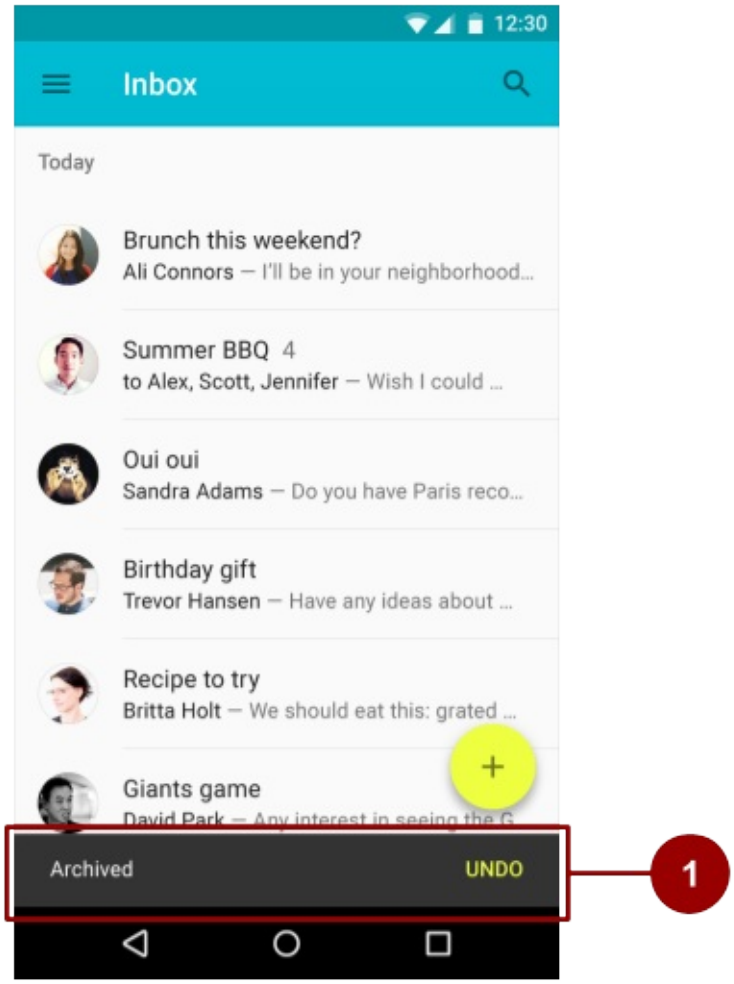
Misalnya, layout berikut memiliki dua tampilan anak: `FrameLayout` yang berisi materi utama (diisi oleh `Fragmen` pada waktu proses), dan `ListView` untuk panel samping navigasi.

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <!-- The main content view -->
    <FrameLayout
        android:id="@+id/content_frame"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <!-- The navigation drawer -->
    <ListView android:id="@+id/left_drawer"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:choiceMode="singleChoice"
        android:divider="@android:color/transparent"
        android:dividerHeight="0dp"
        android:background="#111"/>
</android.support.v4.widget.DrawerLayout>
```

Untuk informasi selengkapnya, lihat Membuat Panel Samping Navigasi dan informasi penggunaan dalam panduan Desain Material.

Snackbar

Snackbar menyediakan masukan singkat tentang suatu operasi melalui pesan dalam bilah horizontal di layar. Snackbar berisi baris teks tunggal yang secara langsung terkait dengan operasi yang dilaksanakan. Snackbar bisa berisi aksi teks,



namun tidak ada ikon.

1. Snackbar

Snackbar secara otomatis hilang setelah waktu tunggu atau setelah interaksi pengguna di tempat lain pada layar. Anda bisa mengaitkan snackbar dengan berbagai tampilan (setiap objek yang diturunkan dari `kelas View`). Akan tetapi, jika Anda mengaitkan snackbar dengan `CoordinatorLayout`, snackbar akan mendapatkan fitur tambahan:

- Pengguna bisa menutup snackbar dengan menggeseknya menjauh.
- Layout akan menggerakkan beberapa elemen UI bila snackbar muncul. Misalnya, jika memiliki FAB, layout akan memindahkan FAB ke atas saat menampilkan snackbar, sebagai ganti menarik snackbar ke atas FAB.

Untuk membuat objek `Snackbar` gunakan metode `Snackbar.make()`. Tetapkan ID tampilan `CoordinatorLayout` untuk digunakan snackbar, pesan yang ditampilkan snackbar, dan lama waktu menampilkan pesan. Misalnya, pernyataan Java ini membuat snackbar dan memanggil `show()` untuk menampilkan snackbar kepada pengguna:

```
Snackbar.make(findViewById(R.id.myCoordinatorLayout), R.string.email_sent,
                        Snackbar.LENGTH_SHORT).show;
```

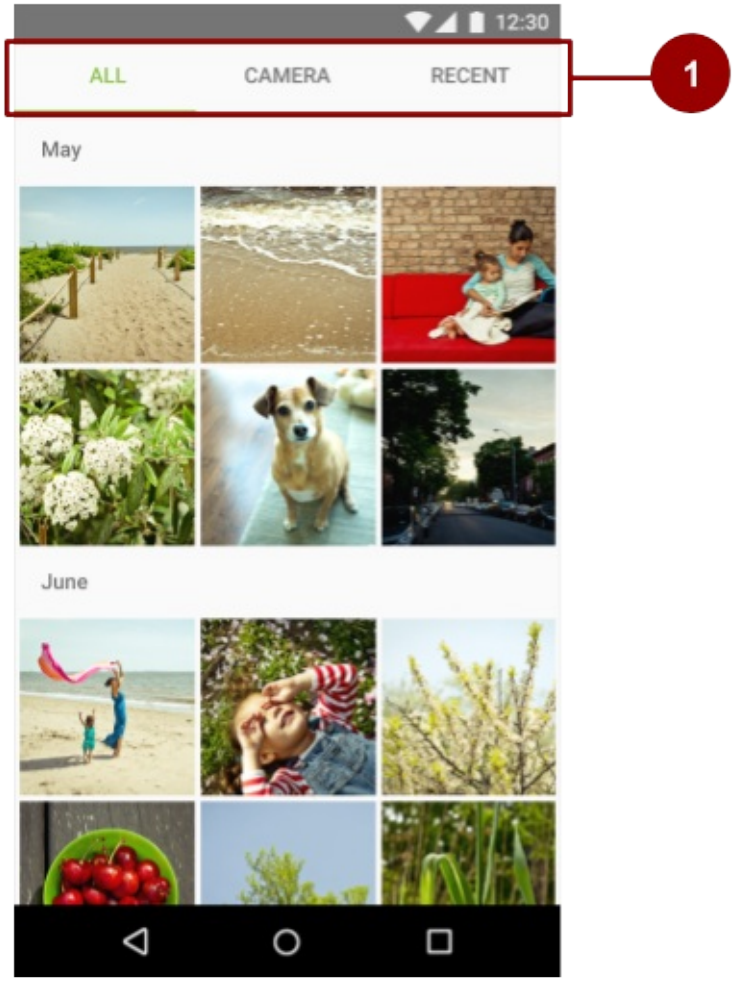
Untuk informasi selengkapnya, lihat referensi Membangun dan Menampilkan Pesan Munculan dan Snackbar. Untuk memastikan Anda menggunakan snackbar sebagaimana dimaksud, lihat informasi penggunaan snackbar dalam panduan Desain Material.

Tip: *Toast* serupa dengan snackbar, namun toast biasanya digunakan untuk perpesanan sistem, dan toast tidak bisa digesek keluar dari layar.

Tab

Gunakan *tab* untuk mengatur materi tingkat tinggi. Misalnya, pengguna dapat menggunakan tab untuk beralih antara Tampilan, rangkatan data, atau aspek fungsional aplikasi. Sajikan tab sebagai baris tunggal di atas materi terkait. Buat label tab yang pendek dan informatif.

Anda bisa menggunakan tab dengan *tampilan gesek* yang bisa digunakan pengguna untuk beralih di antara tab dengan isyarat jari horizontal (paging horizontal). Jika tab Anda menggunakan tampilan gesek, jangan sandingkan tab dengan



materi yang juga mendukung gesekan.

1. Tiga tab, dengan tab **ALL** dipilih

Untuk informasi mengenai implementasi tab, lihat Membuat Tampilan Gesek dengan Tab. Untuk memastikan Anda menggunakan tab sebagaimana dimaksud, lihat informasi penggunaan tab dalam panduan Desain Material.

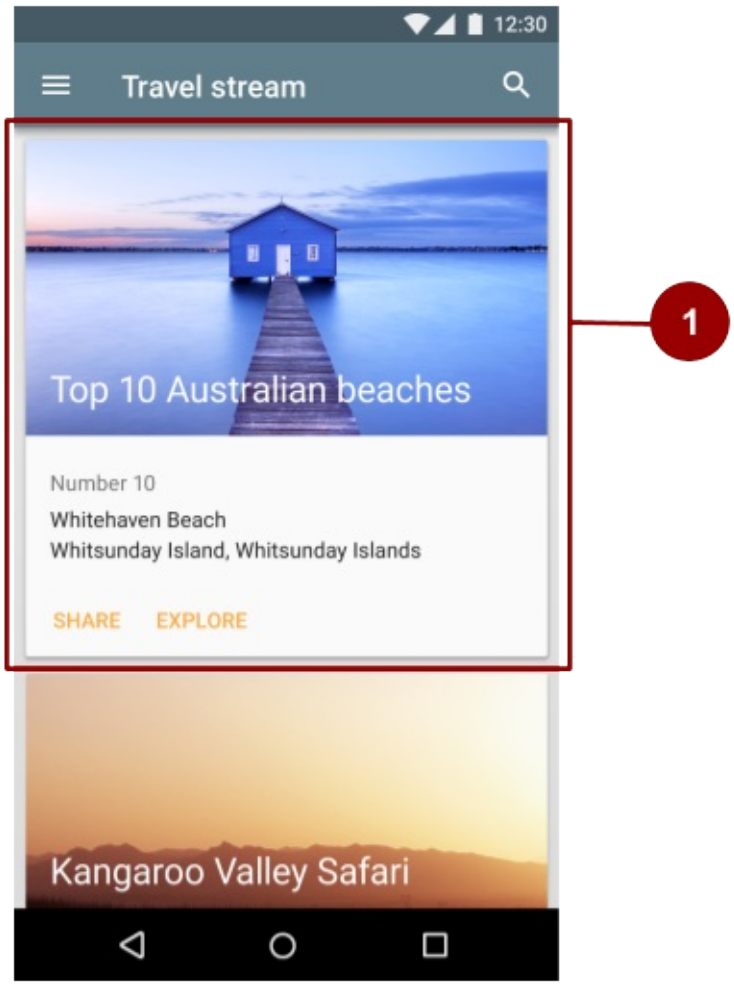
Kartu

Kartu adalah sheet material yang berfungsi sebagai titik masuk informasi yang lebih detail. Setiap kartu mencakup satu subjek saja. Sebuah kartu bisa berisi foto, teks, dan tautan. Kartu bisa menampilkan materi berisi elemen dengan ukuran bervariasi, seperti foto dengan panjang teks yang berbeda.

Kumpulan kartu adalah layout kartu pada bidang yang sama.

Widget `CardView` disertakan sebagai bagian dari pustaka dukungan v7. Untuk menggunakannya, tambahkan dependensi berikut ke file build.gradle:

```
compile 'com.android.support:cardview-v7:24.2.1'
```

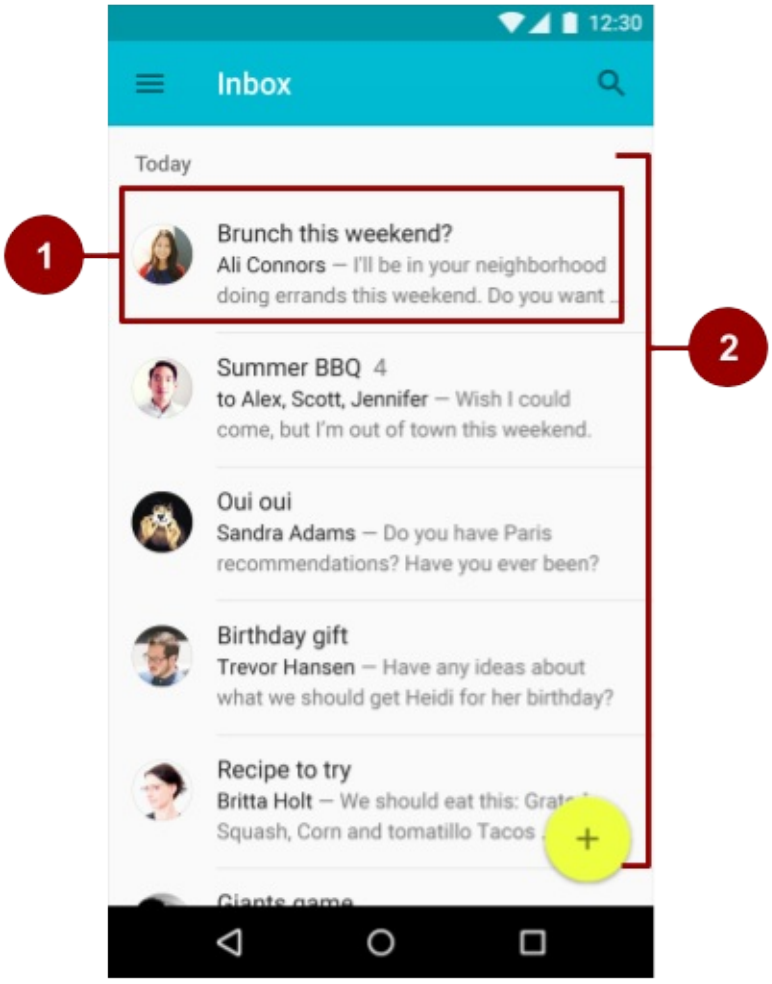


1. Satu kartu dalam kumpulan kartu

Untuk informasi selengkapnya mengenai penggunaan widget `CardView` , kunjungi panduan kartu.

Daftar

Daftar adalah kolom baris bersambung tunggal yang sama lebarnya. Setiap baris berfungsi sebagai kontainer petak. *Petak* berisi materi, dan tingginya bisa bervariasi dalam daftar.



1. Petak dalam daftar
2. Daftar berisi baris dengan lebar hampir sama, setiap berisi sebuah petak

Untuk membuat daftar, gunakan widget `RecyclerView.` Sertakan dependensi berikut dalam file build.gradle.

```
compile 'com.android.support:recyclerview-v7:24.2.1'
```

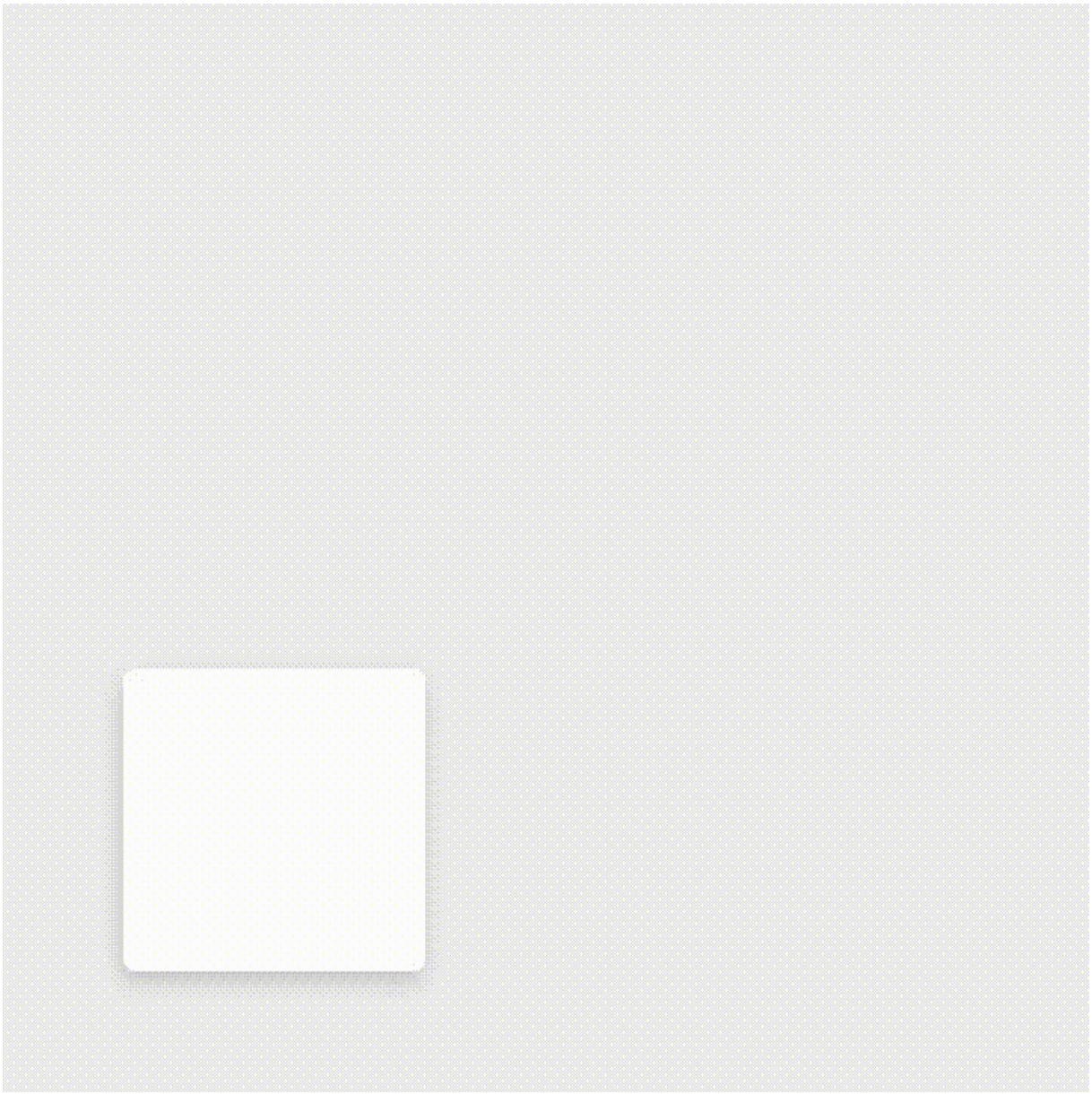
Untuk informasi selengkapnya mengenai pembuatan daftar di Android, lihat panduan daftar kreatif.

Gerakan

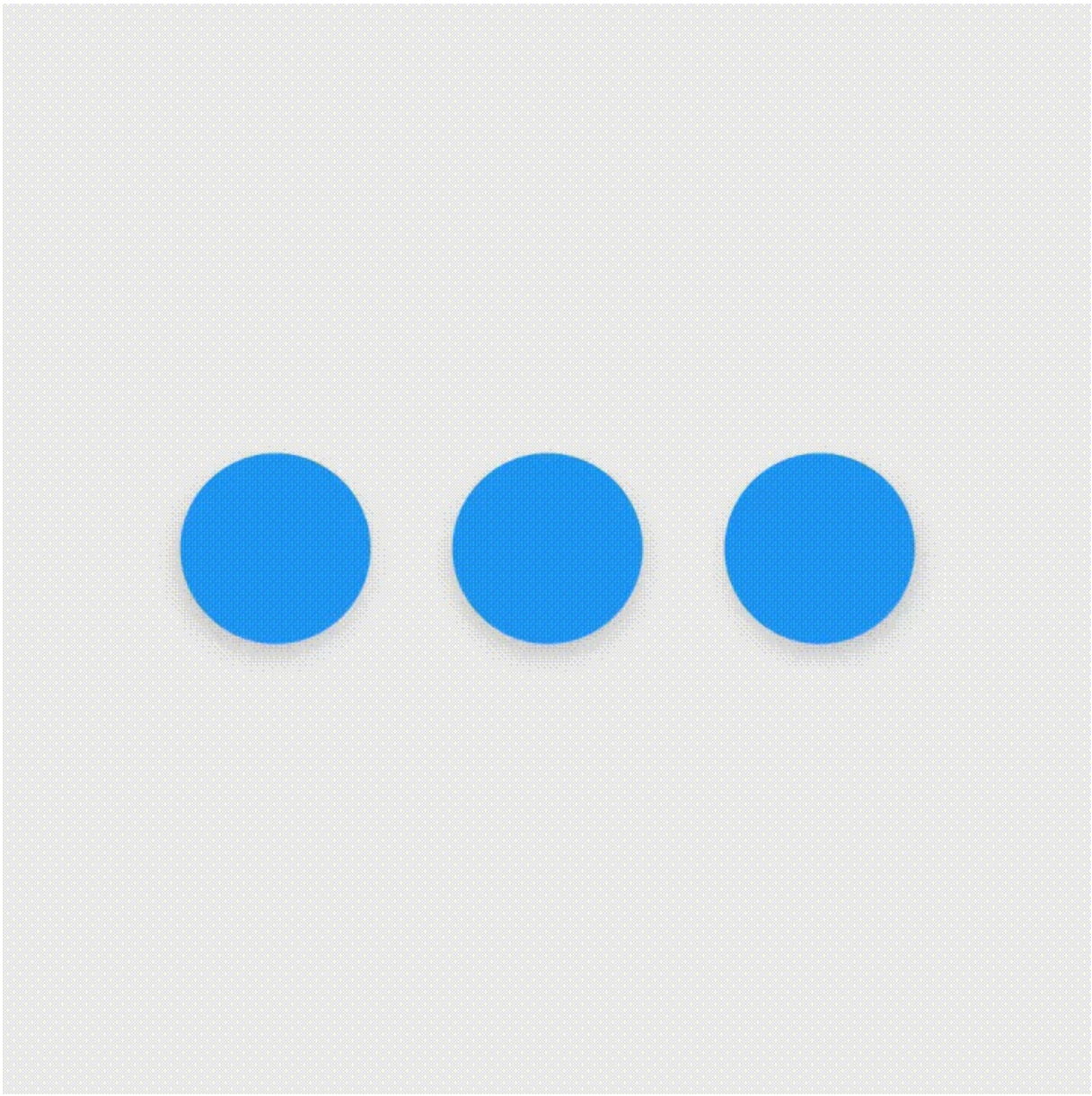
Gerakan dalam dunia Desain Material digunakan untuk menjelaskan hubungan spasial, fungsionalitas, dan intensi dengan keindahan dan fluiditas. Gerakan menampilkan cara aplikasi Anda diatur dan apa yang bisa dilakukannya.

Gerakan dalam Desain Material harus:

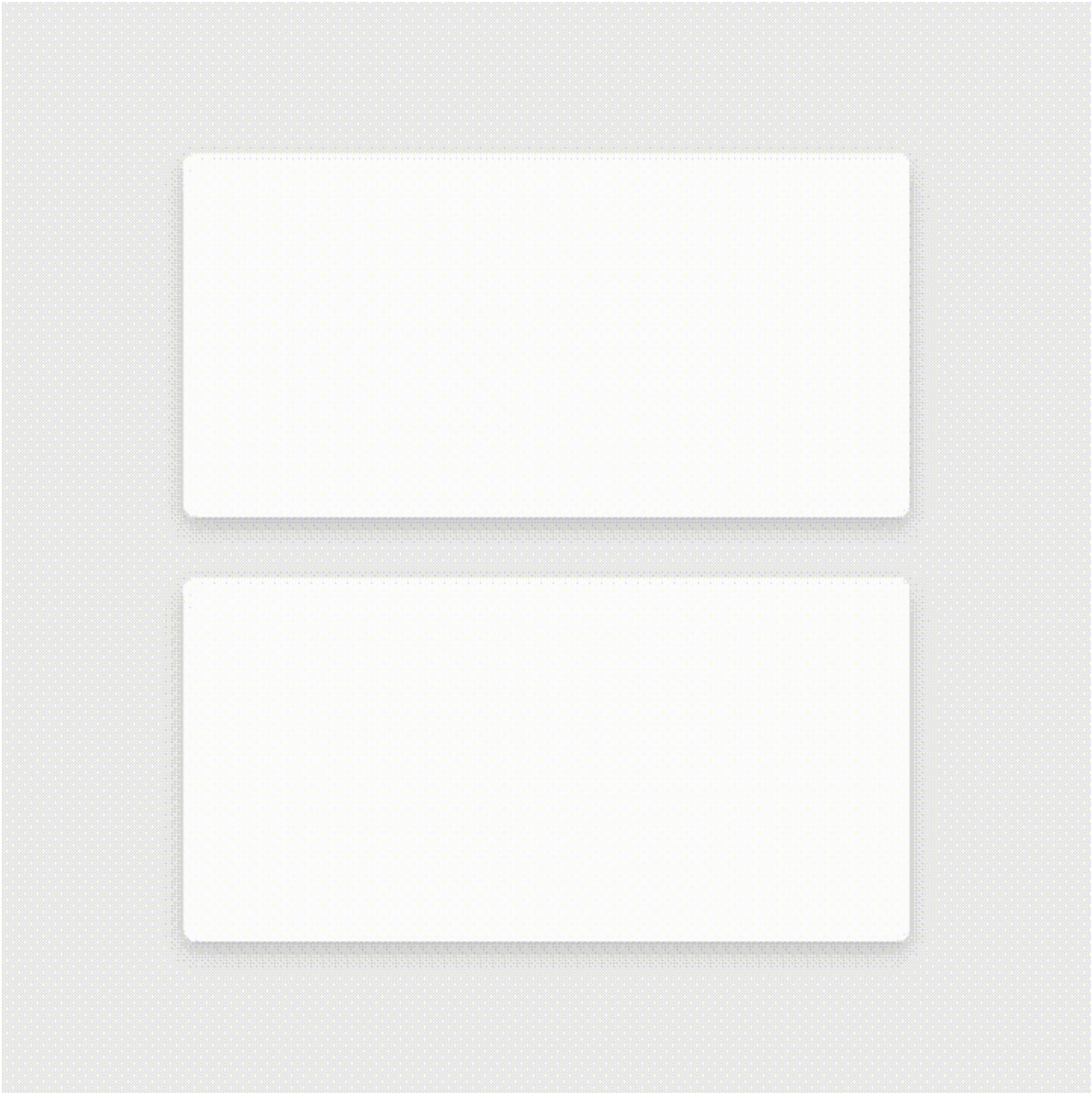
1. **Responsif**. Gerakan dengan cepat merespons masukan pengguna secara tepat saat pengguna memicunya.
2. **Alami**. Gerakan diilhami oleh kekuatan dalam dunia nyata. Misalnya, gaya sungguhan seperti gravitasi mengilhami gerakan elemen sepanjang busur, bukan dalam garis lurus.



3. **Sadar**. Material menyadari keadaan sekitarnya, termasuk pengguna dan material lain di sekitarnya. Objek bisa tertarik pada objek lain dalam UI, dan merespons sesuai dengan maksud pengguna. Saat elemen bertransisi ke dalam tampilan, gerakannya dikoreografikan sedemikian rupa untuk mendefinisikan hubungannya.



4. **Intensional**. Gerakan memandu fokus pengguna ke tempat yang tepat pada saat yang tepat. Gerakan bisa mengomunikasikan sinyal berbeda, misalnya apakah suatu aksi tidak tersedia.



Untuk mempraktikkan prinsip-prinsip ini di Android, gunakan animasi dan transisi.

Animasi

Ada tiga cara untuk membuat animasi dalam aplikasi Anda:

- Animasi properti mengubah properti objek dalam periode waktu yang ditetapkan. Sistem animasi properti dikenalkan dalam Android 3.0 (API level 11). Animasi properti lebih fleksibel daripada animasi tampilan, dan menawarkan lebih banyak fitur.
- Animasi tampilan memperhitungkan titik mulai penggunaan animasi, titik akhir, rotasi, dan aspek animasi lainnya. Sistem animasi tampilan Android lebih tua dari pada sistem animasi properti dan hanya bisa digunakan untuk Tampilan. Sistem ini relatif mudah dipersiapkan dan menawarkan cukup kemampuan untuk banyak kasus penggunaan.
- Animasi dapat digambar memungkinkan Anda memuat serangkaian sumber daya dapat digambar berturut-turut untuk membuat animasi. Animasi dapat digambar berguna jika Anda ingin menganimasikan sesuatu yang lebih mudah dinyatakan dengan sumber daya dapat digambar, seperti kemajuan gambar bitmap.

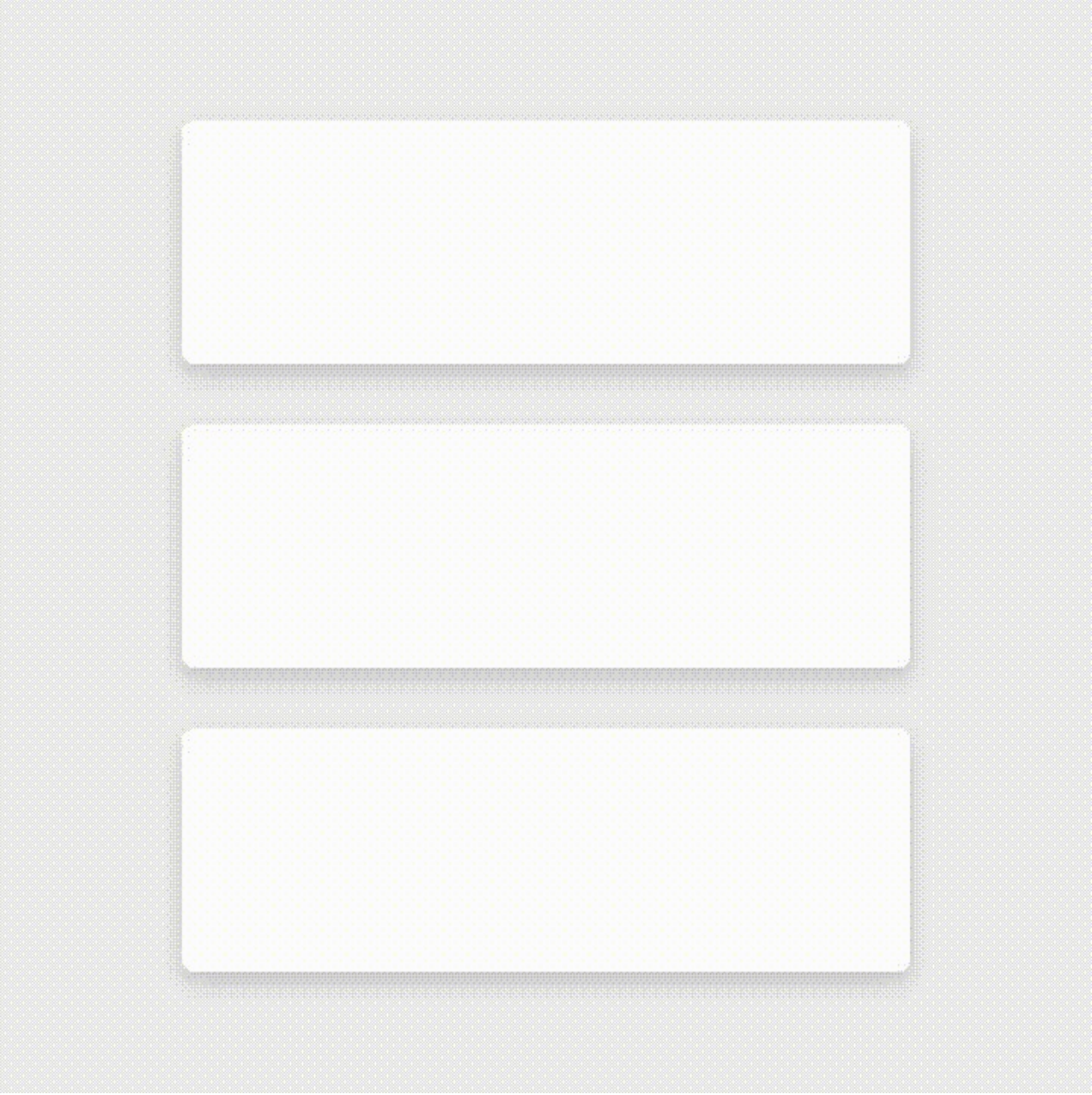
Untuk detail lengkap tentang tiga tipe animasi ini, lihat Ringkasan Animasi dan Grafik.

Tema Desain Material menyediakan beberapa animasi default untuk masukan sentuh dan aktivitas. API animasi memungkinkan Anda membuat animasi khusus untuk masukan sentuh dalam kontrol UI, perubahan keadaan tampilan, dan transisi aktivitas.

Masukan sentuh

Masukan sentuh menyediakan konfirmasi instan pada titik kontak saat pengguna berinteraksi dengan elemen UI. Animasi masukan sentuh default untuk tombol menggunakan kelas RippleDrawable, yang bertransisi di antara berbagai keadaan dengan efek riak.

Dalam contoh ini, riak tinta meluas dari titik sentuhan untuk mengonfirmasi masukan pengguna. Kartu "mengangkat" dan mentransmisikan bayangan untuk menunjukkan keadaan aktif:



Di sebagian besar kasus, Anda harus menerapkan fungsionalitas riak dalam XML tampilan dengan menetapkan latar belakang tampilan seperti berikut:

- `?android:attr/selectableItemBackground` untuk riak berbatas.
- `?android:attr/selectableItemBackgroundBorderless` untuk riak yang meluas melampaui tampilan. Latar belakang ini digambar di atas, dan dibatasi oleh, induk tampilan terdekat dengan latar belakang bukan nol.
  Catatan: Atribut `selectableItemBackgroundBorderless` ini diperkenalkan dalam API Level 21.

Atau, Anda bisa mendefinisikan RippleDrawable sebagai sumber daya XML dengan menggunakan elemen `<ripple>` .

Anda bisa menetapkan warna ke objek-objek RippleDrawable. Untuk mengubah warna default masukan sentuh, gunakan atribut tema `android:colorControlHighlight` .

Untuk informasi selengkapnya, lihat referensi API untuk kelas RippleDrawable.

Singkap melingkar

Animasi singkap menampilkan atau menyembunyikan sekelompok elemen UI dengan menganimasikan batas pemotongan. Dalam *singkap melingkar*, Anda menampilkan atau menyembunyikan tampilan dengan menggerakkan lingkaran pemotongan. (*Lingkar pemotongan* adalah lingkaran yang memotong atau menyembunyikan bagian gambar yang ada di luar lingkaran.)

Untuk menganimasikan lingkaran pemotongan, gunakan metode `ViewAnimationUtils.createCircularReveal()` . Misalnya, inilah cara menyingkap tampilan yang sebelumnya tidak terlihat dengan menggunakan singkap melingkar:

```
// previously invisible view
View myView = findViewById(R.id.my_view);

// get the center for the clipping circle
int cx = myView.getWidth() / 2;
int cy = myView.getHeight() / 2;

// get the final radius for the clipping circle
float finalRadius = (float) Math.hypot(cx, cy);

// create the animator for this view (the start radius is zero)
Animator anim =
    ViewAnimationUtils.createCircularReveal(myView, cx, cy, 0, finalRadius);

// make the view visible and start the animation
myView.setVisibility(View.VISIBLE);
anim.start();
```

Begini caranya menyembunyikan tampilan yang sebelumnya terlihat menggunakan singkap melingkar:

```
// previously visible view
final View myView = findViewById(R.id.my_view);

// get the center for the clipping circle
int cx = myView.getWidth() / 2;
int cy = myView.getHeight() / 2;

// get the initial radius for the clipping circle
float initialRadius = (float) Math.hypot(cx, cy);

// create the animation (the final radius is zero)
Animator anim =
    ViewAnimationUtils.createCircularReveal(myView, cx, cy, initialRadius, 0);

// make the view invisible when the animation is done
anim.addListener(new AnimatorListenerAdapter() {
    @Override
    public void onAnimationEnd(Animator animation) {
        super.onAnimationEnd(animation);
        myView.setVisibility(View.INVISIBLE);
    }
});

// start the animation
anim.start();
```

Transisi aktivitas

Transisi aktivitas adalah animasi yang menyediakan koneksi visual di antara berbagai keadaan UI Anda. Anda bisa menetapkan animasi khusus untuk *masuk* dan *keluar* transisi, dan untuk transisi elemen bersama di antara aktivitas.

- *Transisi masuk* menentukan cara tampilan dalam aktivitas masuk suatu adegan. Misalnya dalam *transisi letupan masuk*, tampilan memasuki adegan dari luar dan melayang ke arah pusat layar.
- *Transisi keluar* menentukan cara tampilan dalam aktivitas keluar dari adegan tersebut. Misalnya dalam *transisi letupan keluar*, tampilan keluar dari adegan dengan bergerak menjauhi pusat layar.
- *Transisi elemen bersama* menentukan penggunaan bersama suatu tampilan oleh dua transisi aktivitas di antara aktivitas-aktivitas ini. Misalnya, jika dua aktivitas memiliki gambar yang sama dalam berbagai posisi dan ukuran, transisi elemen `changeImageTransform` bersama akan menerjemahkan dan menskalakan gambar dengan halus di antara aktivitas ini.

Untuk menggunakan transisi ini, setel atribut transisi dalam elemen `<style>` di XML Anda. Contoh berikut membuat tema bernama `BaseAppTheme` yang mewarisi salah satu tema Desain Material. Tema `BaseAppTheme` menggunakan ketiga tipe transisi aktivitas:

```
<style name="BaseAppTheme" parent="android:Theme.Material">
  <!-- enable window content transitions -->
  <item name="android:windowActivityTransitions">true</item>

  <!-- specify enter and exit transitions -->
  <item name="android:windowEnterTransition">@transition/explode</item>
  <item name="android:windowExitTransition">@transition/explode</item>

  <!-- specify shared element transitions -->
  <item name="android:windowSharedElementEnterTransition">
    @transition/change_image_transform</item>
  <item name="android:windowSharedElementExitTransition">
    @transition/change_image_transform</item>
</style>
```

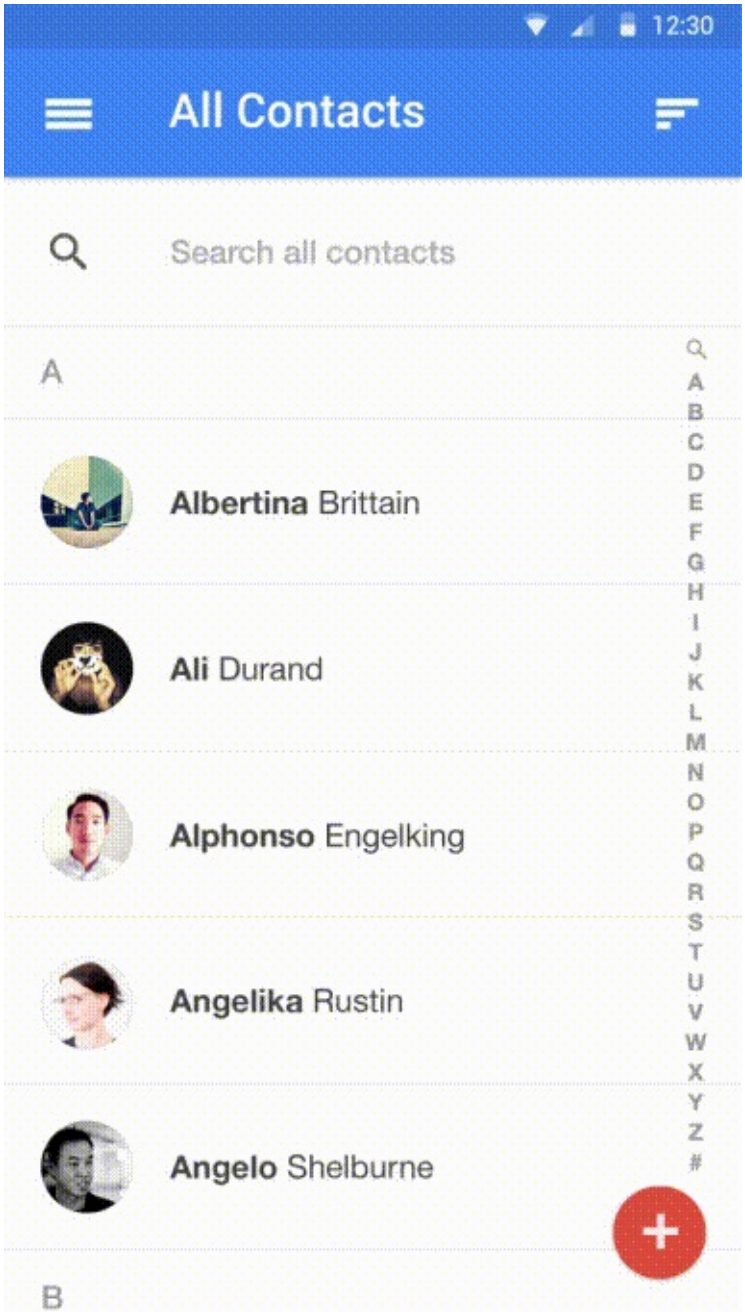
Tema `change_image_transform` dalam contoh ini didefinisikan seperti berikut:

```
<!-- res/transition/change_image_transform.xml -->
<!-- (see also Shared Transitions below) -->
<transitionSet xmlns:android="http://schemas.android.com/apk/res/android">
  <changeImageTransform/>
</transitionSet>
```

Elemen `changeImageTransform` berhubungan dengan kelas `ChangeImageTransform.` Untuk informasi selengkapnya, lihat referensi API untuk `Transisi` .

Untuk mengaktifkan transisi materi jendela dalam kode Java Anda, panggil metode `Window.requestFeature()` :

```
// inside your activity (if you did not enable transitions in your theme)
getWindow().requestFeature(Window.FEATURE_CONTENT_TRANSITIONS);

// set an exit transition
getWindow().setExitTransition(new Explode());
```



Untuk menetapkan transisi dalam kode Anda, panggil metode berikut bersama objek `Transisi`:

- `Window.setEnterTransition()`
- `Window.setExitTransition()`
- `Window.setSharedElementEnterTransition()`
- `Window.setSharedElementExitTransition()`

Untuk detail tentang metode ini, lihat `dokumentasi referensi Jendela`.

Untuk memulai aktivitas yang menggunakan transisi, gunakan metode `ActivityOptions.makeSceneTransitionAnimation()`.

Untuk informasi selengkapnya tentang implementasi transisi dalam aplikasi Anda, lihat panduan transisi aktivitas.

Gerakan melengkung

Dalam Android 5.0 (API level 21) ke atas, Anda bisa mendefinisikan kurva pengaturan waktu khusus dan pola gerakan melengkung untuk animasi. Caranya, gunakan kelas `PathInterpolator`, yang menginterpolasikan jalur objek berdasarkan kurva Bézier atau objek `Path`. Interpolator menetapkan kurva gerakan dalam bujur sangkar 1x1, dengan titik-titik jangkar

di (0,0) dan (1,1) dan titik-titik kontrol yang Anda tetapkan menggunakan argumen konstruktor. Anda juga bisa mendefinisikan interpolator jalur sebagai sumber daya XML:

```
<pathInterpolator xmlns:android="http://schemas.android.com/apk/res/android"
    android:controlX1="0.4"
    android:controlY1="0"
    android:controlX2="1"
    android:controlY2="1"/>
```

Sistem menyediakan sumber daya XML untuk tiga kurva dasar dalam spesifikasi desain material:

- `@interpolator/fast_out_linear_in.xml`
- `@interpolator/fast_out_slow_in.xml`
- `@interpolator/linear_out_slow_in.xml`

Untuk menggunakan objek `PathInterpolator`, teruskan ke metode `Animator.setInterpolator()`.

Kelas `ObjectAnimator` memiliki konstruktor yang bisa Anda gunakan untuk menganimasikan koordinat sepanjang jalur menggunakan dua atau beberapa properti sekaligus. Misalnya, kode Java berikut menggunakan sebuah objek `Path` untuk menganimasikan properti X dan Y suatu tampilan:

```
ObjectAnimator mAnimator;
mAnimator = ObjectAnimator.ofFloat(view, View.X, View.Y, path);
...
mAnimator.start();
```

Animasi khusus lainnya

Animasi khusus lainnya dimungkinkan, termasuk animasi perubahan keadaan (menggunakan kelas `StateListAnimator`) dan animasi sumber daya dapat digambar untuk vektor (menggunakan kelas `AnimatedVectorDrawable`). Untuk detail lengkap, lihat Mendefinisikan Animasi Khusus.

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Desain Material: Daftar, Kartu, dan Warna

Ketahui selengkapnya

- Desain Material untuk Android
- Desain Material untuk Developer

5.3: Menyediakan Sumber Daya untuk Layout Adaptif

Daftar Isi:

Layout adaptif adalah layout yang bekerja dengan baik pada berbagai orientasi dan ukuran layar, berbagai perangkat, berbagai lokal dan bahasa, serta sebagai versi Android.

Dalam bab ini Anda akan mempelajari cara membuat layout adaptif dengan mengeksternalkan dan mengelompokkan sumber daya, menyediakan sumber daya alternatif, dan menyediakan sumber daya default dalam aplikasi.

Mengeksternalkan sumber daya

Saat *mengeksternalkan* sumber daya, Anda harus memisahkannya dari kode aplikasi. Misalnya, sebagai ganti hard-code string ke dalam kode Anda, beri nama string tersebut dan tambahkan ke file res/values/strings.xml.

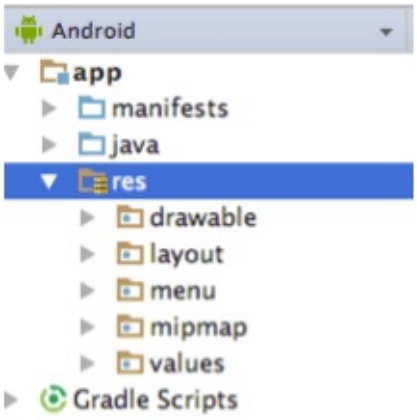
Selalu eksternalkan sumber daya seperti sumber daya dapat digambar, ikon, layout, dan string. Inilah alasan mengapa hal itu penting:

- Anda bisa memelihara sumber daya yang dieksternalkan secara terpisah dari kode lainnya. Jika sumber daya digunakan di sejumlah tempat dalam kode dan perlu diubah, Anda hanya perlu mengubahnya di satu tempat.
- Anda bisa menyediakan sumber daya alternatif yang mendukung konfigurasi perangkat tertentu, misalnya perangkat dengan berbagai bahasa atau ukuran layar. Hal ini menjadi semakin penting karena semakin banyak perangkat berbasis Android yang tersedia.

Mengelompokkan sumber daya

Simpan semua sumber daya Anda dalam folder res/. Atur sumber daya berdasarkan tipe ke dalam folder di /res. Anda harus menggunakan nama standar untuk folder-folder ini.

Misalnya, tangkapan layar berikut menampilkan hierarki file untuk proyek kecil, sebagaimana terlihat dalam tampilan Project "Android" di Android Studio. Folder yang berisi sumber daya default proyek ini menggunakan nama standar:



`drawable`, `layout`, `menu`, `mipmap` (untuk ikon), dan `values`.

Tabel 1 mencantumkan nama folder sumber daya standar. Tipe dijelaskan lebih lengkap dalam panduan Menyediakan Sumber Daya.

Tabel 1: Nama Folder Sumber Daya Standar

Nama	Tipe Sumber Daya
animator/	File XML yang mendefinisikan animasi properti.
anim/	File XML yang mendefinisikan animasi tween.
color/	File XML yang mendefinisikan "daftar keadaan" warna. (Ini berbeda dari file colors.xml dalam folder values/.) Lihat Sumber Daya Daftar Keadaan Warna.
drawable/	File Bitmap (WebP, PNG, 9-patch, JPG, GIF) dan file XML yang dikompilasi menjadi sumber daya dapat digambar. Lihat Sumber Daya Dapat Digambar.
mipmap/	File sumber daya dapat digambar untuk beragam kepadatan ikon peluncur. Lihat Ringkasan Proyek.
layout/	File XML yang mendefinisikan layout antarmuka pengguna. Lihat Sumber Daya Layout.
menu/	File XML yang mendefinisikan menu aplikasi. Lihat Sumber Daya Menu.
raw/	File arbitrer yang disimpan dalam bentuk mentah. Untuk membuka sumber daya ini dengan InputStream mentah, panggil Resources.openRawResource() bersama ID sumber daya, yaitu `R.raw.filename`. Jika Anda memerlukan akses ke nama file dan hierarki file asli, pertimbangkan untuk menyimpan sumber daya dalam folder assets/ sebagai ganti res/raw/. File dalam assets/ tidak diberi ID sumber daya, sehingga Anda bisa membacanya hanya dengan menggunakan AssetManager.
values/	File XML yang berisi nilai-nilai sederhana, seperti string, integer, dan warna. Untuk kejelasan, tempatkan tipe sumber daya yang unik dalam file berbeda. Misalnya, inilah beberapa ketentuan penamaan file untuk sumber daya yang bisa Anda buat dalam folder ini: • arrays.xml untuk larik sumber daya (larik bertipe) • dimens.xml untuk nilai dimensi • strings.xml, colors.xml, styles.xml Lihat Sumber Daya String, Sumber Daya Gaya, dan Tipe Sumber Daya Lainnya.
xml/	File XML arbitrer yang bisa dibaca pada waktu proses dengan memanggil Resources.getXml(). Berbagai file konfigurasi XML, seperti konfigurasi yang dapat ditelusuri, harus disimpan di sini, bersama setelan preferensi.

Sumber daya alternatif

Sebagian besar aplikasi menyediakan *sumber daya alternatif* untuk mendukung konfigurasi perangkat tertentu. Misalnya, aplikasi Anda harus menyertakan sumber daya dapat digambar alternatif untuk kepadatan layar berbeda, dan sumber daya string alternatif untuk bahasa yang berbeda. Pada waktu proses, Android akan mendeteksi konfigurasi perangkat saat ini dan memuat sumber daya yang sesuai untuk aplikasi Anda.

Jika tidak ada sumber daya yang tersedia untuk konfigurasi perangkat tertentu, Android menggunakan sumber daya default yang Anda sertakan dalam aplikasi—sumber daya dapat digambar default, yang terdapat dalam folder res/drawable/, string teks default, yang ada dalam file res/values/strings.xml, dan seterusnya.

Seperti sumber daya default, sumber daya alternatif disimpan dalam folder dalam res/. Folder sumber daya alternatif menggunakan konvensi penamaan berikut:

```
<resource_name>-<config_qualifier>
```

- `<resource_name>` adalah nama folder untuk tipe sumber daya ini, seperti yang ditampilkan dalam Tabel 1. Misalnya, "drawable" atau "values".

- `<config_qualifier>` menetapkan konfigurasi perangkat yang menggunakan sumber daya ini. Qualifier yang memungkinkan ditampilkan dalam Tabel 2.

  Untuk menambahkan beberapa qualifier ke satu nama folder, pisahkan qualifier dengan tanda hubung. Jika menggunakan qualifier untuk folder sumber daya, Anda harus mencantumkannya agar qualifier tersebut dicantumkan dalam Tabel 2.

Contoh dengan satu qualifier:

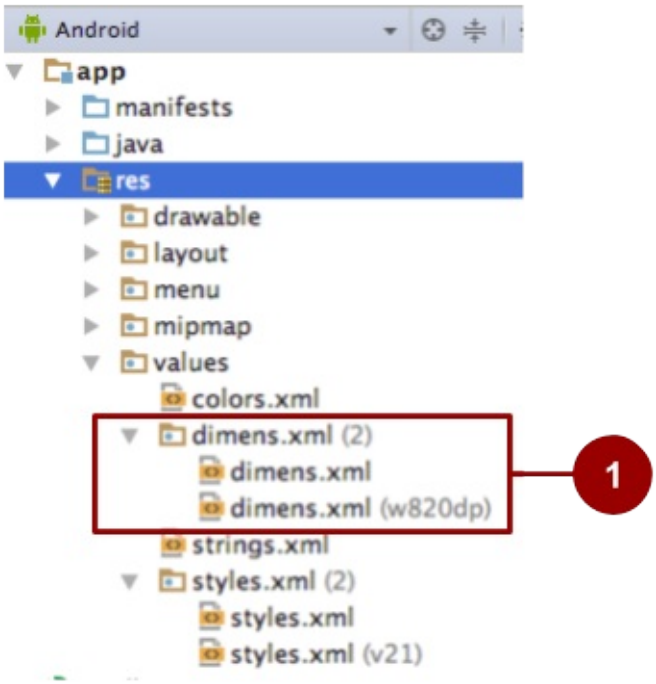
- Sumber daya string yang dilokalkan ke bahasa Jepang akan ada dalam file `res/values-ja/strings.xml` . Sumber daya string default (sumber daya yang akan digunakan bila sumber daya khusus bahasa tidak ditemukan) akan ada dalam `res/values/strings.xml` . Perhatikan, file XML memiliki nama identik, dalam hal ini "strings.xml".
- Sumber daya gaya untuk API level 21 dan yang lebih tinggi akan ada dalam file `res/values-v21/styles.xml` . Sumber daya gaya default akan ada dalam `res/values/styles.xml` .

Contoh dengan beberapa qualifier:

- Sumber daya layout untuk layout kanan ke kiri yang berjalan dalam mode "malam" akan ada dalam folder `res/layout-ldrtl-night/` .

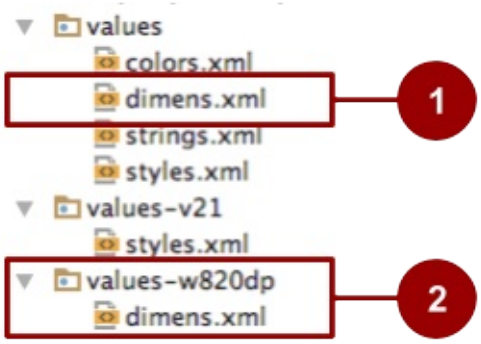
Dalam tampilan "Android" di Android Studio, qualifier tidak ditambahkan ke akhir folder. Sebagai gantinya, qualifier akan ditampilkan berupa label di sisi kanan file dalam tanda kurung. Misalnya, dalam tampilan "Android" di bawah ini, folder `res/values/dimens.xml/` menampilkan dua file:

- File `dimens.xml` , yang menyertakan sumber daya dimensi default.
- File `dimens.xml (w820dp)` , yang menyertakan sumber daya dimensi untuk perangkat dengan lebar layar yang tersedia minimum 820 dp.



1. Dalam tampilan "Android" di Android Studio, sumber daya default untuk dimensi ditampilkan dalam folder yang sama dengan sumber daya alternatif untuk dimensi.

Dalam tampilan "Project" di Android Studio, informasi yang sama disajikan secara berbeda, seperti yang ditampilkan dalam



tangkapan layar di bawah ini.

1. Dalam tampilan "Project" di Android Studio, sumber daya default untuk dimensi ditampilkan dalam folder `res/values` .
2. Sumber daya alternatif untuk dimensi ditampilkan dalam folder `res/values-<qualifier>` .

Tabel 2 menampilkan qualifier konfigurasi yang didukung Android. Qualifier tersebut dicantumkan dalam urutan yang harus Anda gunakan saat mengombinasikan beberapa qualifier dalam satu nama folder. Misalnya dalam `res/layout-ldrtl-night/` , qualifier untuk arah layout dicantumkan sebelum qualifier untuk mode malam, karena arah layout dicantumkan sebelum mode malam dalam tabel.

Qualifier ini dijelaskan secara detail dalam Menyediakan Sumber Daya Alternatif.

Tabel 2: Qualifier untuk Menamakan Sumber Daya Alternatif

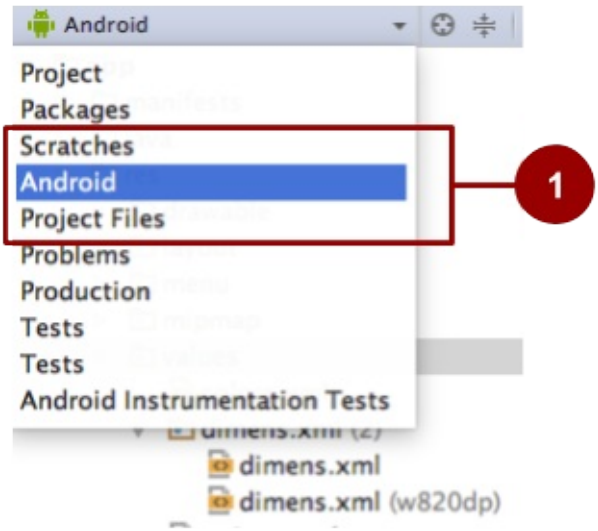
Prioritas	Qualifier	Keterangan
1	MCC dan MNC	Kode negara seluler (MCC), secara opsional diikuti oleh kode jaringan seluler (MNC) dari kartu SIM dalam perangkat. Misalnya, `mcc310` adalah A.S. untuk operator mana saja, `mcc310-mnc004` adalah A.S. untuk Verizon, dan `mcc208-mnc00` adalah Prancis untuk Orange.
2	Pelokalan	Bahasa, atau bahasa dan region Contoh: `en` , `en-rUS` , `fr-rFR` , `fr-rCA` . Dijelaskan dalam Pelokalan, di bawah ini.
3	Arah layout	Arah layout aplikasi Anda. Nilai-nilai yang memungkinkan antara lain `ldltr` (arah layout kiri ke kanan, yang merupakan default) dan `ldrtl` (arah layout kanan ke kiri). Untuk mengaktifkan fitur layout kanan ke kiri, setel supportsRtl ke `"true"` dan setel targetSdkVersion ke 17 atau yang lebih tinggi.
4	Lebar terkecil	Ukuran layar dasar sebagaimana ditunjukkan oleh dimensi terpendek dari area layar yang tersedia. Contoh: `sw320dp` . Dijelaskan dalam Lebar terkecil, di bawah ini.
5	Lebar yang tersedia	Lebar layar minimum yang tersedia tempat sumber daya digunakan. Ditetapkan dalam unit dp. Formatnya adalah `wdp` , misalnya, `w720dp` dan `w1024dp` .
6	Tinggi yang tersedia	Tinggi layar minimum yang tersedia tempat sumber daya digunakan. Ditetapkan dalam unit dp. Formatnya adalah `hdp` , misalnya, `h720dp` dan `h1024dp` .
7	Ukuran layar	Nilai-nilai yang memungkinkan: • `small:` Layar seperti layar berkepadatan rendah QVGA • `normal:` Layar seperti kepadatan medium HVGA • `large:` Layar seperti kepadatan medium VGA • `xlarge:` Layar seperti pada perangkat bergaya tablet
8	Aspek layar	Nilai-nilai yang memungkinkan antara lain `long` (untuk layar seperti WQVGA, WVGA, FWVGA) dan `notlong` (untuk layar seperti QVGA, HVGA, dan VGA).

9	Layar bulat	Nilai-nilai yang memungkinkan antara lain `round` (untuk layar seperti pada perangkat bulat yang dapat dikenakan) dan `notround` (untuk layar persegi panjang seperti ponsel).
10	Orientasi layar	Nilai-nilai yang memungkinkan: `port, land.` Dijelaskan dalam Orientasi layar, di bawah ini.
11	Mode UI	Nilai-nilai yang memungkinkan: • `car:` Perangkat sedang menampilkan di dok mobil • `desk:` Menampilkan di dok meja • `television:` Menampilkan pada layar besar yang jauh dari pengguna, terutama diorientasikan di sekitar D-pad atau interaksi non-pointer lainnya • `appliance:` Perangkat berfungsi sebagai alat, tanpa tampilan • `watch:` Perangkat memiliki tampilan dan dikenakan di pergelangan tangan
12	Mode malam	Nilai-nilai yang memungkinkan: • `night` • `notnight`
13	Kepadatan piksel layar	Nilai-nilai yang memungkinkan: • `ldpi:` Layar berkepadatan rendah; sekitar 120 dpi. • `mdpi:` Layar berkepadatan medium (pada HVGA tradisional); sekitar 160 dpi. • `hdpi:` Layar berkepadatan tinggi; sekitar 240 dpi. • `xhdpi:` Sekitar 320 dpi. Ditambahkan dalam API level 8. • `xxhdpi:` Sekitar 480 dpi. Ditambahkan dalam API level 16. • `xxxhdpi:` Ikon peluncur saja; sekitar 640 dpi. Ditambahkan dalam API level 18. • `nodpi:` Untuk sumber daya bitmap yang tidak ingin Anda skalakan agar cocok dengan kepadatan perangkat. • `tvdpi:` Layar antara mdpi dan hdpi; sekitar 213 dpi. Dimaksudkan untuk televisi, dan sebagian besar aplikasi tidak memerlukannya. Ditambahkan dalam API level 13. • `anydpi:` Cocok dengan semua kepadatan layar dan diprioritaskan di atas qualifier lainnya. Berguna bagi sumber daya dapat digambar untuk vektor. Ditambahkan dalam API level 21. **Catatan:** Menggunakan qualifier kepadatan tidak berarti sumber daya *hanya* untuk layar dengan kepadatan itu saja. Jika Anda tidak menyediakan sumber daya alternatif dengan qualifier yang lebih cocok dengan konfigurasi perangkat saat ini, sistem mungkin akan menggunakan sumber daya mana saja yang paling cocok.
14	Tipe layar sentuh	Nilai-nilai yang memungkinkan antara lain `notouch` (perangkat tidak memiliki layar sentuh) dan `finger` (perangkat memiliki layar sentuh).
15	Ketersediaan keyboard	Nilai-nilai yang memungkinkan: • `keysexposed:` Perangkat menyediakan keyboard. • `keyshidden:` Perangkat memiliki keyboard fisik yang tersedia namun tersembunyi, dan perangkat tidak mengaktifkan keyboard perangkat lunak. • `keyssoft:` Perangkat mengaktifkan keyboard perangkat lunak, baik itu terlihat atau tidak.
16	Metode masukan teks utama	Nilai-nilai yang memungkinkan: • `nokeys:` Perangkat tidak memiliki tombol fisik untuk masukan teks. • `qwerty:` Perangkat memiliki keyboard fisik qwerty, baik terlihat atau tidak kepada pengguna. • `12key:` Perangkat memiliki keyboard fisik 12 tombol, baik terlihat atau tidak kepada pengguna.
17	Ketersediaan tombol navigasi	Nilai-nilai yang memungkinkan antara lain `navexposed` (tombol navigasi tersedia untuk pengguna) dan `navhidden` (tombol navigasi tidak tersedia, misalnya di balik penutup yang tertutup).
		Nilai-nilai yang memungkinkan:

18	Metode navigasi non-sentuh utama	• `nonav:` Perangkat tidak memiliki fasilitas navigasi selain layar sentuh. • `dpad:` Perangkat memiliki pad pengarah (D-pad). • `trackball:` Perangkat memiliki trackball. • `wheel:` Perangkat memiliki roda pengarah untuk navigasi (tidak umum).
19	Versi platform (API level)	Level API yang didukung oleh perangkat. Dijelaskan dalam Versi platform, di bawah ini.

Membuat sumber daya alternatif

Untuk membuat folder sumber daya alternatif paling mudah di Android Studio, gunakan tampilan "Android" dalam jendela

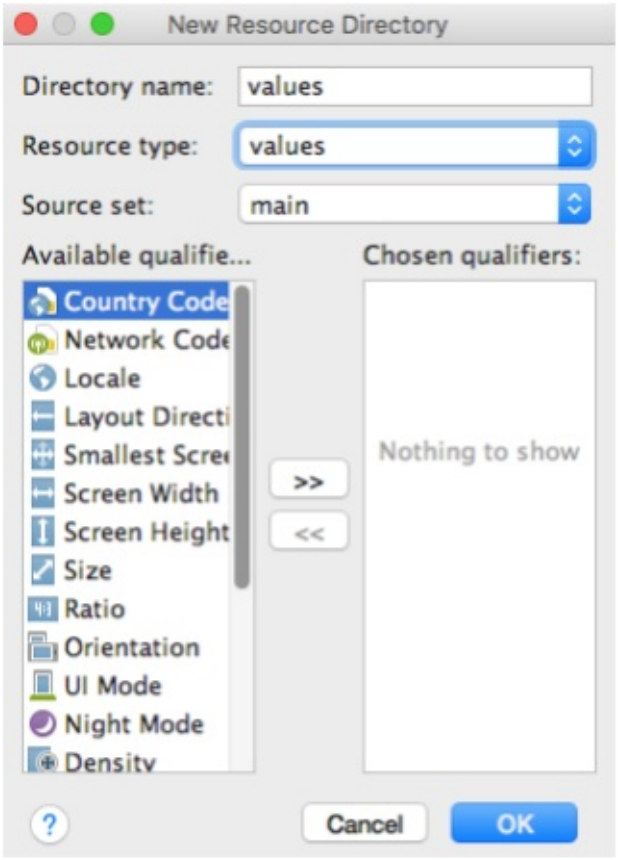


alat (bantu) Project.

1. Memilih tampilan "Android" di Android Studio. Jika Anda tidak melihat opsi ini, pastikan jendela alat (bantu) Project terlihat dengan memilih **View > Tool Windows > Project**.

Untuk menggunakan Android Studio dalam membuat folder sumber daya alternatif khusus konfigurasi yang baru dalam res/:

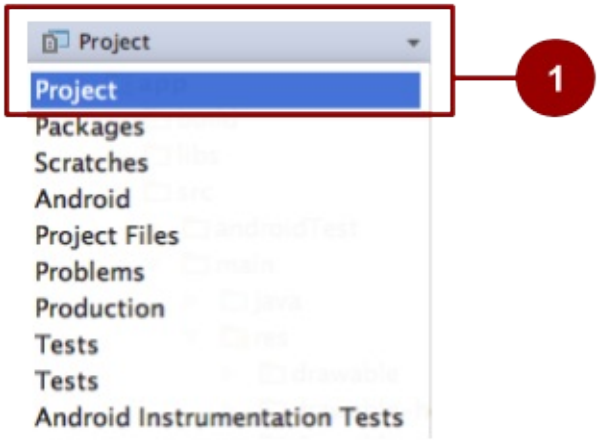
1. Pastikan Anda menggunakan tampilan "Android", seperti yang ditampilkan di atas.
2. Klik kanan folder res/ dan pilih **New > Android resource directory**. Kotak dialog New Resource Directory akan



muncul.

3. Pilih tipe sumber daya (dijelaskan dalam Tabel 1) dan qualifier (dijelaskan dalam Tabel 2) yang diterapkan pada rangkaian sumber daya alternatif ini.
4. Klik OK.

Jika Anda tidak bisa melihat folder baru di jendela alat (bantu) Project di Android Studio, beralihlah ke tampilan "Project", seperti yang ditampilkan dalam tangkapan layar di bawah ini. Jika Anda tidak melihat opsi ini, pastikan jendela alat (bantu) Project terlihat dengan memilih **View > Tool Windows > Project**.



Simpan sumber daya alternatif dalam folder baru File sumber daya alternatif harus diberi nama yang sama persis dengan file sumber daya default, misalnya "styles.xml" atau "dimens.xml".

Untuk dokumentasi lengkap tentang sumber daya alternatif, lihat Menyediakan Sumber Daya Alternatif.

Qualifier sumber daya alternatif umum

Bagian ini menjelaskan beberapa qualifier yang umum digunakan. Tabel 2 menyediakan daftar lengkap.

Orientasi layar

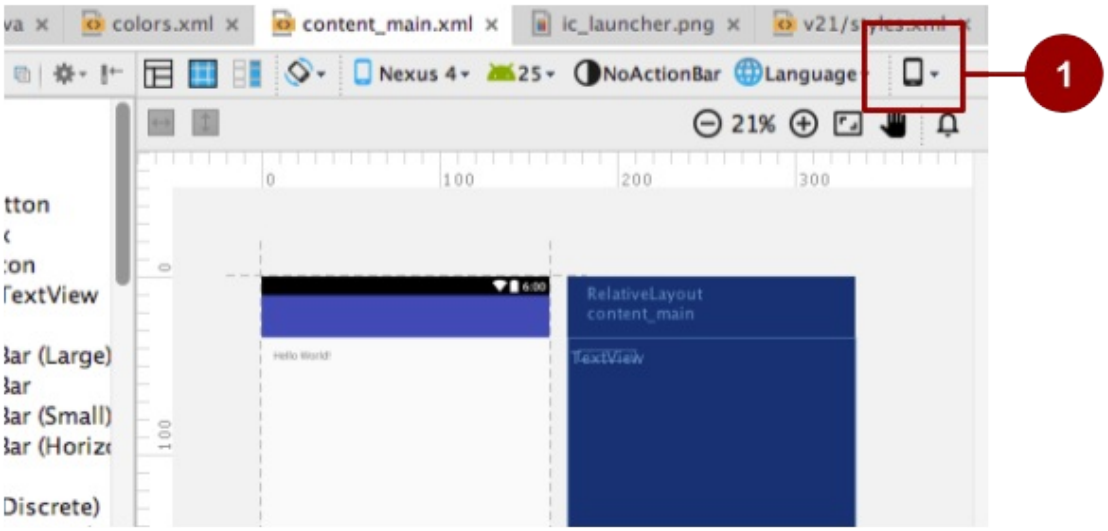
Qualifier orientasi layar memiliki dua nilai yang memungkinkan:

- `port` : Perangkat dalam mode potret (vertikal). Misalnya, `res/layout-port/` akan berisi file layout untuk digunakan saat perangkat dalam mode potret.
- `land` : Perangkat dalam mode lanskap (horizontal). Misalnya, `res/layout-land/` akan berisi file layout untuk digunakan saat perangkat dalam mode lanskap.

Jika pengguna memutar layar saat aplikasi berjalan, dan sumber daya alternatif tersedia, Android akan secara otomatis memuat ulang aplikasi Anda dengan sumber daya alternatif yang cocok dengan konfigurasi perangkat baru. Untuk informasi tentang mengontrol cara aplikasi Anda berperilaku selama perubahan konfigurasi, lihat Menangani Perubahan Waktu Proses.

Untuk membuat varian file XML layout Anda untuk orientasi lanskap dan tampilan yang lebih besar, gunakan editor layout. Untuk menggunakan editor layout:

1. Di Android Studio, buka file XML. Editor layout akan muncul.
2. Dari menu tarik-turun dalam menu **Layout Variants**, pilih opsi seperti **Create Landscape Variant**. Menu **Layout Variants**, yang tampak saat file XML terbuka di Android Studio, disorot dalam tangkapan layar di bawah ini.



Layout untuk orientasi lanskap yang berbeda akan muncul, dan file XML baru akan dibuatkan untuk Anda. Misalnya, Anda memiliki file bernama "activity_main.xml (land)" beserta file "activity_main.xml" asal. Anda bisa menggunakan editor untuk mengubah layout baru tanpa mengubah layout asal.

Lihat praktik sebelumnya tentang layout untuk contoh desain layout.

Lebar terkecil

Qualifier lebar terkecil menetapkan lebar minimum perangkat. Ini adalah tinggi dan lebar layar terpendek yang tersedia, "lebar terkecil" untuk layar. Lebar terkecil adalah karakteristik perangkat yang memiliki ukuran layar tetap, dan tidak berubah saat orientasi layar berubah.

Tetapkan lebar terkecil dalam unit dp, menggunakan format berikut ini:

```
sw<N>dp
```

dalam hal ini: `<N>` adalah lebar minimum. Misalnya, sumber daya dalam suatu file bernama `res/values-sw320dp/styles.xml` digunakan jika lebar layar perangkat selalu setidaknya 320 dp.

Anda bisa menggunakan qualifier ini untuk memastikan layout tertentu tidak akan digunakan kecuali jika lebar yang tersedia setidaknya `<N>` dps, bagaimanapun orientasi layar saat ini.

Beberapa nilai untuk ukuran layar umum:

- 320, untuk perangkat dengan konfigurasi layar seperti
  - 240x320 ldpi (handset QVGA)
  - 320x480 mdpi (handset)
  - 480x800 hdpi (handset kepadatan tinggi)
- 480, untuk layar seperti 480x800 mdpi (tablet/handset).
- 600, untuk layar seperti 600x1024 mdpi (tablet 7").
- 720, untuk layar seperti 720x1280 mdpi (tablet 10").

Bila aplikasi Anda menyediakan beberapa folder sumber daya dengan nilai berbeda untuk qualifier lebar terkecil, sistem akan menggunakan nilai terdekat dengan (tanpa melebihi) lebar terkecil perangkat.

Contoh:

`res/values-sw600dp/dimens.xml` berisi dimensi untuk gambar. Bila aplikasi berjalan pada perangkat dengan lebar terkecil 600 dp atau yang lebih tinggi (seperti tablet), Android akan menggunakan gambar dalam folder ini.

Versi platform

Qualifier versi platform menetapkan API level minimum yang didukung oleh perangkat. Misalnya, gunakan `v11` untuk API level 11 (perangkat dengan Android 3.0 atau yang lebih tinggi). Lihat dokumen Android API level untuk informasi selengkapnya tentang nilai ini.

Gunakan qualifier versi platform bila Anda menggunakan sumber daya untuk fungsionalitas yang tidak tersedia dalam versi Android sebelumnya.

Misalnya, gambar WebP memerlukan API level 14 (Android 4.0) atau yang lebih tinggi, dan untuk dukungan penuh diperlukan API level 17 (Android 4.2) atau yang lebih tinggi. Jika Anda menggunakan gambar WebP:

- Letakkan versi default gambar dalam folder `res/drawable` . Gambar ini harus menggunakan format gambar yang didukung untuk semua API level, misalnya PNG.
- Letakkan versi gambar WebP dalam folder `res/drawable-v17` . Jika perangkat menggunakan API Level 17 atau yang lebih tinggi, Android akan memilih sumber daya ini pada waktu proses.

Pelokalan

Qualifier pelokalan menetapkan bahasa, dan region yang bersifat opsional. Qualifier ini adalah dua huruf kode bahasa ISO 639-1, bisa diikuti oleh dua huruf kode region ISO 3166-1-alpha-2 (diawali oleh huruf kecil `r`).

Anda bisa menetapkan bahasa saja, namun tidak boleh region saja. Contoh:

- `res/values-fr-rFR/strings.xml`

  String dalam file ini digunakan pada perangkat yang dikonfigurasi untuk bahasa Prancis dan regionnya juga disetel ke France.

- `res/mipmap-fr-rCA/`

  Ikon dalam folder ini digunakan pada perangkat yang dikonfigurasi untuk bahasa Prancis dan regionnya disetel ke Canada.

- `res/layout-ja/content_main.xml`

  Layout ini digunakan pada perangkat yang dikonfigurasi untuk bahasa Jepang.

Jika pengguna mengubah bahasa atau region dalam setelan sistem perangkat saat aplikasi berjalan, dan jika sumber daya alternatif tersedia, Android akan secara otomatis memuat ulang aplikasi Anda dengan sumber daya alternatif yang cocok dengan konfigurasi perangkat baru. Untuk informasi tentang mengontrol cara aplikasi Anda berperilaku selama perubahan konfigurasi, lihat Menangani Perubahan Waktu Proses.

Untuk panduan lengkap tentang pelokalan, lihat Melokalkan dengan Sumber Daya.

Menyediakan sumber daya default

Sumber daya default menetapkan desain dan materi default untuk aplikasi Anda. Misalnya, saat aplikasi berjalan dalam lokal yang tidak Anda sediakan teks lokal khususnya, Android akan memuat string default dari res/values/string.xml. Jika file default ini tidak ada, atau jika file kehilangan bahkan satu string yang diperlukan oleh aplikasi, maka aplikasi Anda tidak akan berjalan dan akan menampilkan kesalahan.

Sumber daya default memiliki nama folder sumber daya standar (`values` , misalnya) tanpa qualifier dalam nama folder

atau dalam tanda kurung setelah nama file.

1. Sumber daya default
   Tip: Selalu sediakan sumber daya default, karena aplikasi mungkin berjalan pada konfigurasi perangkat yang tidak Anda antisipasi.

Kadang-kadang versi Android baru menambahkan qualifier konfigurasi yang tidak didukung versi lama. Jika Anda menggunakan qualifier sumber daya baru dan mempertahankan kompatibilitas kode dengan versi Android lama, maka saat versi Android lama menjalankan aplikasi, aplikasi akan mogok kecuali jika sumber daya default tersedia. Hal ini karena versi Android lama tidak bisa menggunakan sumber daya alternatif yang diberi nama dengan qualifier baru.

Misalnya, anggaplah `minSdkVersion` Anda disetel ke `4` dan tetapkan semua sumber daya dapat digambar menggunakan mode malam, yang berarti bahwa Anda meletakkan sumber daya dapat digambar dalam `res/drawable-night/` dan `res/drawable-notnight/` . Dalam contoh ini:

- Bila perangkat API level 4 perangkat menjalankan aplikasi, perangkat tidak bisa mengakses sumber daya dapat digambar. Versi Android tidak mengetahui tentang `night` dan `notnight` , karena qualifier ini tidak ditambahkan hingga API level 8. Aplikasi mogok, karena tidak menyertakan sumber daya default apa pun untuk digunakan.

Dalam contoh ini, Anda mungkin ingin `notnight` menjadi kasus default Anda. Untuk menyelesaikan masalah, keluarkan qualifier `notnight` dan masukkan sumber daya dapat digambar Anda dalam `res/drawable/` dan `res/drawable-night/` . Dengan solusi ini:

- Bila perangkat API level 4 menjalankan aplikasi, aplikasi akan menggunakan sumber daya dalam folder `res/drawable/` default.
- Bila perangkat pada API level 8 atau yang lebih tinggi menggunakan aplikasi, aplikasi akan menggunakan sumber daya dalam folder `res/drawable-night/` kapan pun perangkat dalam mode malam. Pada saat lainnya, perangkat akan menggunakan sumber daya (`notnight`) default.

Untuk menyediakan kompatibilitas perangkat terbaik, sediakan sumber daya default untuk setiap sumber daya yang diperlukan aplikasi Anda. Setelah sumber daya default Anda berada di tempatnya, buat sumber daya alternatif untuk konfigurasi perangkat khusus dengan menggunakan qualifier konfigurasi perangkat khusus seperti yang ditampilkan dalam Tabel 2.

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Mendukung Lanskap, Beberapa Ukuran Layar dan Pelokalan

Ketahui selengkapnya

- Menyediakan Sumber Daya
- Ringkasan Sumber Daya
- Melokalkan dengan Sumber Daya

6.1: Menguji Antarmuka Pengguna

Materi:

- Menguji antarmuka pengguna dengan benar
- Mempersiapkan lingkungan pengujian Anda
- Menggunakan Espresso untuk pengujian yang mencakup satu aplikasi
- Menggunakan UI Automator untuk pengujian yang mencakup beberapa aplikasi
- Praktik terkait
- Ketahui selengkapnya

Menguji antarmuka pengguna dengan benar

Menulis dan menjalankan pengujian merupakan bagian penting dari siklus development aplikasi Android. Pengujian yang ditulis dengan baik bisa membantu menangkap bug sejak dini dalam siklus development, sehingga lebih mudah diperbaiki, dan menambah kepercayaan diri dalam kode Anda.

Pengujian antarmuka pengguna (UI) memfokuskan pada pengujian aspek-aspek antarmuka pengguna dan interaksi dengan pengguna. Mengenali dan menindaklanjuti masukan pengguna merupakan prioritas tinggi dalam pengujian antarmuka pengguna dan validasi. Anda perlu memastikan bahwa aplikasi tidak hanya mengenali tipe masukan, namun juga bertindak dengan tepat. Sebagai developer, Anda harus terbiasa menguji antarmuka pengguna untuk memastikan bahwa pengguna tidak menemukan hasil yang tidak diharapkan atau mendapatkan pengalaman buruk saat berinteraksi dengan aplikasi Anda. Pengujian antarmuka pengguna bisa membantu Anda mengenali kontrol masukan bila masukan yang tidak diharapkan harus ditangani secara halus atau harus memicu validasi masukan.

Catatan: Kami sangat menyarankan Anda menggunakan Android Studio untuk membangun aplikasi pengujian, karena menyediakan persiapan proyek, penyertaan pustaka, dan kemudahan pengemasan. Anda bisa menjalankan pengujian UI pada berbagai perangkat Android fisik maupun virtual, kemudian menganalisis hasilnya serta membuat perubahan kode tanpa meninggalkan lingkungan development.
UI berisi tampilan dengan elemen grafik seperti tombol, menu, dan bidang teks, masing-masing dengan serangkaian properti. Untuk menguji UI dengan benar, Anda perlu:

- Menguji semua kejadian UI bersama tampilan:
  - Ketuk tampilan UI, dan masukkan data atau buat pilihan.
  - Periksa *nilai-nilai properti* setiap tampilan—yang disebut dengan *keadaan* UI—pada waktu yang berbeda selama eksekusi.
- Sediakan masukan ke semua tampilan UI. Gunakan kesempatan ini untuk menguji masukan yang tidak benar, seperti teks saat yang diharapkan adalah angka.
- Periksa keluaran dan representasi data UI—seperti string dan integer—untuk melihat apakah konsisten dengan yang Anda harapkan.

Selain fungsionalitas, pengujian UI mengevaluasi elemen desain seperti layout, warna, font, ukuran font, label, kotak teks, format teks, keterangan, tombol, daftar, ikon, tautan, dan materi.

Pengujian Manual

Sebagai developer aplikasi, Anda mungkin menguji setiap komponen UI secara manual saat menambahkan komponen ke UI aplikasi. Seiring berjalannya development, satu pendekatan terhadap pengujian UI adalah meminta seorang penguji melakukan serangkaian operasi pengguna pada aplikasi target dan memverifikasi apakah aplikasi itu berperilaku dengan benar.

Akan tetapi, pendekatan manual ini bisa menghabiskan waktu, membosankan, dan rawan kesalahan. Dengan menguji UI aplikasi kompleks secara manual, Anda tidak mungkin bisa mencakup semua permutasi interaksi pengguna. Anda juga harus melakukan pengujian berulang ini secara manual pada berbagai konfigurasi perangkat dalam emulator, dan pada berbagai perangkat berbeda. Singkatnya, masalah inheren pada pengujian manual dibagi menjadi dua kategori:

- *Ukuran domain:* UI memiliki banyak operasi yang memerlukan pengujian. Bahkan aplikasi yang relatif kecil bisa memiliki ratusan kemungkinan operasi UI. Selama siklus development, UI bisa berubah secara signifikan, meskipun aplikasi dasarnya tidak berubah. Pengujian manual dengan petunjuk untuk mengikuti jalur tertentu melalui UI bisa gagal sewaktu-waktu, karena tombol, item menu, atau dialog bisa mengubah lokasi atau penampilan.
- *Urutan:* Beberapa fungsionalitas aplikasi hanya bisa tercapai dengan urutan kejadian UI. Misalnya, untuk menambahkan gambar pada pesan yang akan dikirim, pengguna mungkin harus mengetuk tombol kamera dan menggunakan kamera untuk mengambil gambar, atau tombol foto untuk memilih gambar yang sudah ada, kemudian menghubungkan gambar tersebut dengan pesan—biasanya dengan mengetuk tombol bagikan atau kirim. Bertambahnya jumlah kemungkinan operasi juga menambah masalah pengurutan.

Pengujian otomatis

Saat mengotomatiskan pengujian interaksi pengguna, Anda membebaskan diri dan sumber daya untuk pekerjaan lainnya. Untuk menghasilkan serangkaian kasus pengujian, desainer pengujian berupaya mencakup semua fungsionalitas sistem dan menguji keseluruhan UI. Melakukan semua interaksi UI secara otomatis memudahkan dalam menjalankan pengujian untuk berbagai keadaan perangkat (seperti orientasi) dan berbagai konfigurasi.

Untuk menguji aplikasi Android, umumnya Anda harus membuat tipe pengujian UI otomatis ini:

- *Pengujian UI yang bekerja dalam satu aplikasi:* Memverifikasi apakah aplikasi berperilaku seperti yang diharapkan bila pengguna melakukan aksi tertentu atau memasukkan masukan tertentu. Pengujian ini memungkinkan Anda memeriksa apakah aplikasi mengembalikan keluaran UI yang benar sebagai respons terhadap interaksi pengguna dalam aktivitas aplikasi. Kerangka kerja pengujian UI seperti Espresso yang memungkinkan Anda melakukan simulasi tindakan pengguna lewat program dan menguji interaksi pengguna dalam-aplikasi yang kompleks.
- *Pengujian UI yang mencakup beberapa aplikasi:* Memverifikasi perilaku interaksi yang tepat di antara beberapa aplikasi pengguna atau antara aplikasi pengguna dan aplikasi sistem. Misalnya, Anda bisa menguji suatu aplikasi yang meluncurkan aplikasi Maps untuk menampilkan arah, atau meluncurkan picker kontak Android untuk memilih penerima pesan. Kerangka kerja pengujian UI yang mendukung interaksi lintas-aplikasi, seperti UI Automator, memungkinkan Anda membuat pengujian untuk skenario berdasarkan pengguna.

Menggunakan Espresso untuk pengujian yang mencakup satu aplikasi

Kerangka kerja pengujian Espresso dalam Android Testing Support Library menyediakan API untuk menulis pengujian UI guna menyimulasikan interaksi pengguna dalam satu aplikasi. Pengujian Espresso berjalan pada perangkat sesungguhnya atau emulator dan berperilaku seolah-olah pengguna sesungguhnya sedang menggunakan aplikasi.

Anda bisa menggunakan Espresso untuk membuat pengujian UI guna memverifikasi hal-hal berikut secara otomatis:

- Aplikasi mengembalikan keluaran UI yang tepat sebagai respons terhadap urutan tindakan pengguna pada suatu perangkat.
- Navigasi dan kontrol masukan aplikasi menampilkan aktivitas, tampilan, dan bidang yang tepat.
- Aplikasi akan merespons secara tepat dengan dependensi tiruan, seperti data dari server luar, atau bisa bekerja dengan metode backend yang dimatikan untuk menyimulasikan interaksi sungguhan dengan komponen backend yang bisa diprogram untuk menjawab dengan serangkaian respons yang telah didefinisikan.

Manfaat utama menggunakan Espresso adalah akses ke informasi instrumentasi, seperti konteks aplikasi, sehingga Anda bisa memantau semua interaksi sistem yang dimiliki sistem Android pada aplikasi. Manfaat utama lainnya adalah secara otomatis menyinkronkan tindakan pengujian dengan UI aplikasi. Espresso mendeteksi kapan thread utama tidak digunakan, sehingga bisa menjalankan pengujian Anda pada waktu yang tepat, yang akan memperbaiki keandalan pengujian Anda. Kemampuan ini juga membebaskan Anda dari keharusan menambahkan solusi pengaturan waktu, seperti masa tidur, dalam kode pengujian.

Kerangka kerja pengujian Espresso dapat digunakan bersama runner pengujian AndroidJUnitRunner dan memerlukan instrumentasi, yang nanti akan dijelaskan di bagian ini. Pengujian Espresso bisa dijalankan pada perangkat yang menjalankan Android 2.2 (API level 8) dan yang lebih tinggi.

Menggunakan UI Automator untuk pengujian yang mencakup beberapa aplikasi

Kerangka kerja pengujian UI Automator dalam Android Testing Support Library bisa membantu Anda memverifikasi perilaku interaksi yang tepat di antara berbagai aplikasi pengguna atau antara aplikasi pengguna dan aplikasi sistem. UI Automator juga menampilkan apa yang terjadi pada perangkat sebelum dan setelah aplikasi diluncurkan.

UI Automator API memungkinkan Anda berinteraksi dengan elemen yang terlihat pada perangkat. Pengujian Anda bisa mencari komponen UI dengan menggunakan deskriptor seperti teks yang ditampilkan dalam komponen itu atau keterangan materinya. Alat (bantu) penampil menyediakan antarmuka visual untuk memeriksa hierarki layout dan menampilkan properti komponen UI yang tampak di latar depan perangkat.

Berikut ini adalah fungsi penting dari UI Automator:

- Seperti Espresso, UI Automator memiliki akses ke informasi interaksi sistem sehingga Anda bisa memantau semua interaksi yang dimiliki sistem Android dengan aplikasi.
- Pengujian Anda bisa mengirim Maksud atau meluncurkan Aktivitas (tanpa menggunakan perintah shell) dengan mendapatkan objek Context melalui getContext().
- Anda bisa menyimulasikan interaksi pengguna pada sekumpulan item, seperti lagu dalam album musik atau daftar email dalam inbox.
- Anda bisa menyimulasikan pengguliran vertikal atau horizontal pada tampilan.
- Anda bisa menggunakan metode JUnit Assert untuk menguji apakah komponen UI dalam aplikasi mengembalikan hasil yang diharapkan.

Kerangka kerja pengujian UI Automator dapat digunakan bersama runner pengujian AndroidJUnitRunner dan memerlukan instrumentasi, yang dijelaskan di bagian berikutnya. Pengujian UI Automator bisa dijalankan pada perangkat yang menjalankan Android 4.3 (API level 18) atau yang lebih tinggi.

Apa yang dimaksud dengan instrumentasi?

Instrumentasi Android adalah serangkaian metode kontrol, atau *kait*, dalam sistem Android, yang mengontrol komponen Android dan cara sistem Android memuat aplikasi.

Biasnaya, sistem menjalankan semua komponen aplikasi dalam proses yang sama. Anda bisa mengizinkan beberapa komponen, seperti penyedia materi, untuk berjalan dalam proses terpisah, namun biasanya tidak bisa memaksa aplikasi ke proses yang sama dengan aplikasi yang berjalan lainnya.

Akan tetapi, pengujian instrumentasi bisa memuat paket pengujian sekaligus aplikasi ke dalam proses yang sama. Karena komponen aplikasi dan pengujiannya dalam proses yang sama, pengujian Anda bisa memanggil metode dalam komponen, dan memodifikasi serta memeriksa bidang dalam komponen.

Instrumentasi memungkinkan Anda memantau semua interaksi yang dimiliki sistem Android pada aplikasi, dan memungkinkan pengujian memanggil metode dalam aplikasi, dan memodifikasi serta memeriksa bidang dalam aplikasi, tanpa bergantung pada daur hidup normal aplikasi.

Biasanya, komponen Android berjalan dalam daur hidup yang ditentukan sistem. Misalnya, daur hidup objek Activity dimulai bila sebuah Maksud mengaktifkan Aktivitas. Sistem akan memanggil metode onCreate() objek, kemudian metode onResume(). Bila pengguna memulai aplikasi lain, sistem akan memanggil metode onPause(). Jika kode Aktivitas memanggil metode finish(), sistem akan memanggil metode onDestroy().

Android Framework API tidak menyediakan cara yang bisa digunakan kode aplikasi untuk memanggil metode callback ini secara langsung, namun Anda bisa melakukannya menggunakan pengujian Espresso atau UI Automator bersama instrumentasi.

Mempersiapkan lingkungan pengujian Anda

Untuk menggunakan kerangka kerja Espresso dan UI Automator, Anda perlu menyimpan file sumber daya untuk pengujian instrumentasi di ***module-name*/src/androidTests/java/**. Direktori ini sudah ada bila Anda membuat proyek Android Studio baru. Dalam tampilan Project di Android Studio, direktori ini ditampilkan di **app > java** sebagai ***modul-name* (androidTest)**.

Anda juga perlu melakukan yang berikut ini:

- Pasang Android Support Repository dan Android Testing Support Library.
- Tambahkan dependensi ke file build.gradle proyek.
- Buat file pengujian di direktori **androidTest**.

Memasang Android Support Repository dan Testing Support Library

Anda mungkin sudah memiliki Android Support Repository dan Android Testing Support Library yang dipasang bersama Android Studio. Untuk memeriksa Android Support Repository, ikuti langkah-langkah ini:

1. Di Android Studio pilih **Tools > Android > SDK Manager**.
2. Klik tab **SDK Tools**, dan cari Support Repository.
3. Jika perlu, perbarui atau pasang pustaka tersebut.

Menambahkan dependensi

Bila Anda memulai proyek untuk faktor bentuk Ponsel dan Tablet menggunakan **API 15: Android 4.0.3 (Ice Cream Sandwich)** sebagai SDK minimum, Android Studio versi 2.2 dan yang lebih baru secara otomatis menyertakan dependensi yang Anda perlukan untuk menggunakan Espresso. Untuk memastikan bahwa Anda memiliki dependensi ini, ikuti langkah-langkah ini:

1. Buka file **build.gradle (Modul: app)** dalam proyek untuk memastikan yang berikut ini telah disertakan (bersama dependensi lain) di bagian `dependencies` pada file **build.gradle (Modul: app)** Anda:

   ```
   androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
           exclude group: 'com.android.support', module: 'support-annotations'
       })
   testCompile 'junit:junit:4.12'
   ```

2. Android Studio juga menambahkan pernyataan instrumentasi berikut ke akhir bagian `defaultConfig` :

   ```
   testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
   ```

   Catatan: Jika telah membuat proyek di versi Android Studio sebelumnya, Anda mungkin harus menambahkan dependensi dan pernyataan instrumentasi sendiri.

1. Bila selesai, klik tautan **Sync Now** dalam notifikasi tentang file Gradle di sudut kanan atas jendela.

Mempersiapkan aturan pengujian dan anotasi

Untuk menulis pengujian, Espresso dan UI Automator, gunakan JUnit sebagai kerangka kerja pengujian. JUnit adalah kerangka kerja pengujian unit yang paling populer dan banyak digunakan untuk Java. Kelas pengujian Anda yang menggunakan Espresso atau UI Automator harus dituliskan sebagai kelas pengujian JUnit 4. Jika belum memiliki JUnit, Anda bisa mendapatkannya di http://junit.org/junit4.

Catatan: Revisi JUnit terbaru adalah JUnit 5. Akan tetapi, untuk tujuan penggunaan Espresso atau UI Automator, yang disarankan adalah versi 4.12.
Untuk membuat kelas pengujian JUnit 4 dasar, buat kelas Java untuk pengujian dalam direktori yang ditetapkan pada awal bagian ini. Kelas ini harus berisi satu atau beberapa metode dan aturan perilaku yang didefinisikan oleh anotasi JUnit.

Misalnya, cuplikan berikut menampilkan definisi kelas pengujian dengan anotasi:

```
@RunWith(AndroidJUnit4.class)
public class RecyclerViewTest {

   @Rule
   public ActivityTestRule<MainActivity> mActivityTestRule =
            new ActivityTestRule<>(MainActivity.class);

   @Test
   public void recyclerViewTest() {
      ...
   }
}
```

Anotasi berikut berguna untuk pengujian:

@RunWith

Untuk membuat kelas pengujian JUnit 4 terinstrumentasi, tambahkan anotasi `@RunWith(AndroidJUnit4.class)` di awal definisi kelas pengujian Anda, yang menunjukkan runner yang akan digunakan untuk menjalankan pengujian dalam kelas ini. Runner pengujian adalah pustaka atau serangkaian alat (bantu) yang memungkinkan pengujian terjadi dan hasilnya dicetak ke log.

@SmallTest, @MediumTest, dan @LargeTest

Anotasi Android `@SmallTest`, `@MediumTest`, dan `@LargeTest` menyediakan beberapa kejelasan tentang sumber daya dan fitur yang digunakan pengujian. Misalnya, anotasi `@SmallTest` akan memberi tahu Anda bahwa pengujian tidak berinteraksi dengan sistem file atau jaringan.

Yang berikut ini merangkum apa yang dimaksud:

Fitur	Kecil	Medium	Besar
Akses jaringan	Tidak	localhost saja	Ya
Database	Tidak	Ya	Ya
Akses sistem file	Tidak	Ya	Ya
Gunakan sistem eksternal	Tidak	Tidak disarankan	Ya
Beberapa thread	Tidak	Ya	Ya
Pernyataan tidur	Tidak	Ya	Ya
Properti sistem	Tidak	Ya	Ya
Batas waktu (detik)	60	300	900+

Untuk keterangan anotasi Android `@SmallTest`, `@MediumTest`, dan `@LargeTest` lihat "Ukuran Pengujian" dalam Blog Pengujian Google. Sebagai rangkuman anotasi JUnit, lihat Package org.junit.

@Rule

Sebelum mendeklarasikan metode pengujian, gunakan anotasi `@Rule` seperti ActivityTestRule atau ServiceTestRule. Aturan `@Rule` membuat konteks untuk kode pengujian. Misalnya:

```
@Rule
public ActivityTestRule mActivityRule = new ActivityTestRule<>(
                    MainActivity.class);
```

Aturan ini menggunakan objek `ActivityTestRule` , yang menyediakan pengujian fungsional Aktivitas tunggal—dalam hal ini, `MainActivity.class` .

`ServiceTestRule` adalah aturan JUnit yang menyediakan mekanisme sederhana untuk memulai dan mematikan layanan sebelum dan setelah durasi pengujian Anda. Aturan ini juga menjamin layanan berhasil terhubung saat memulai (atau mengikat ke) suatu layanan.

@Test

Metode pengujian dimulai dengan anotasi `@Test` dan berisi kode untuk dijalankan dan memverifikasi satu fungsi dalam komponen yang ingin Anda uji:

```
@Test
public void testActivityLaunch() { … }
```

Aktivitas dalam pengujian akan diluncurkan sebelum masing-masing pengujian dianotasikan dengan `@Test` . Selama durasi pengujian, Anda akan dapat memanipulasi Aktivitas secara langsung.

@Before dan @After

Dalam kasus yang jarang, Anda perlu mempersiapkan variabel-variabel atau mengeksekusi urutan langkah *sebelum* atau *setelah* melakukan pengujian antarmuka pengguna. Anda bisa menetapkan metode yang akan dijalankan sebelum menjalankan metode `@Test` , dengan menggunakan anotasi `@Before` , dan metode yang akan dijalankan setelahnya, dengan menggunakan anotasi `@After` .

- `@Before` : Metode `@Test` akan mengeksekusi setelah metode yang ditetapkan oleh anotasi `@Before` . Metode `@Before` menghentikan sebelum eksekusi metode `@Test` .
- `@After` : Metode `@Test` akan mengeksekusi sebelum metode ditetapkan oleh anotasi `@After` .

Menggunakan Espresso untuk pengujian yang mencakup satu aplikasi

Saat menulis pengujian, kadang-kadang sulit mendapatkan keseimbangan antara berlebihan dalam menetapkan pengujian atau kurang dalam menetapkan. Berlebihan menetapkan pengujian bisa membuatnya rawan perubahan, sementara kurang dalam menetapkan bisa membuat pengujian kurang berharga, karena akan tetap lulus pengujian bahkan bila elemen UI dan kode yang diuji rusak.

Untuk membuat pengujian yang seimbang, sebaiknya miliki alat yang memungkinkan Anda memilih dengan tepat aspek pengujian dan menjelaskan nilai yang harus dimiliki. Pengujian akan gagal bila perilaku aspek yang diuji menyimpang dari perilaku yang diharapkan, walaupun tetap lulus saat dibuat perubahan minor yang tidak terkait dengan perilaku. Alat (bantu) yang tersedia untuk Espresso adalah kerangka kerja Hamcrest.

Hamcrest (anagram dari "matchers") adalah kerangka kerja yang membantu penulisan pengujian perangkat lunak di Java. Kerangka kerja ini memungkinkan Anda membuat matcher pernyataan khusus, yang memungkinkan aturan pencocokan didefinisikan secara deklaratif.

Tip: Untuk mengetahui selengkapnya tentang Hamcrest, lihat Tutorial Hamcrest.

Menulis pengujian Espresso dengan Hamcrest Matchers

Tulis pengujian Espresso berdasarkan apa yang mungkin dilakukan pengguna saat berinteraksi dengan aplikasi Anda. Konsep utamanya adalah *mencari* kemudian *berinteraksi* dengan elemen UI. Inilah langkah-langkah dasarnya:

1. **Cocokkan tampilan:** Menemukan tampilan.
2. **Lakukan tindakan:** Melakukan klik atau aksi lain yang memicu sebuah kejadian dengan tampilan.
3. **Nyatakan dan verifikasi hasilnya:** Memeriksa keadaan tampilan untuk mengetahui apakah telah mencerminkan keadaan atau perilaku yang diharapkan, sebagaimana yang didefinisikan oleh pernyataan.

Untuk membuat metode pengujian, Anda menggunakan tipe ekspresi Hamcrest berikut untuk membantu menemukan tampilan dan berinteraksi dengannya:

- *ViewMatchers:* Ekspresi ViewMatcher yang akan digunakan bersama `onView()` untuk mencocokkan tampilan dalam hierarki tampilan saat ini. Gunakan `onView()` bersama ViewMatcher agar Anda bisa memeriksa sesuatu atau melakukan beberapa aksi. Yang paling umum adalah:
  - `withId()` : Temukan tampilan dengan `android:id` tertentu (yang biasanya didefinisikan dalam file XML layout). Misalnya:

    ```
    onView(withId(R.id.my_view))
    ```

  - `withText()` : Temukan tampilan dengan teks tertentu, biasanya digunakan bersama `allOf()` dan `withId()` . Misalnya, yang berikut ini menggunakan `allOf` untuk menyebabkan kecocokan jika objek yang diperiksa cocok dengan *semua* ketentuan yang ditetapkan—tampilan menggunakan ID `word` (`withId`), dan tampilan memiliki teks `"Clicked! Word 15"` (`withText`):

    ```
    onView(allOf(withId(R.id.word),
         withText("Clicked! Word 15"), isDisplayed()))
    ```

  - Yang lainnya menyertakan matchers untuk keadaan (`selected` , `focused` , `enabled`), dan keterangan materi serta hierarki (`root` dan `children`).
- *ViewActions:* Ekspresi ViewAction memungkinkan Anda melakukan suatu aksi pada tampilan yang sudah ditemukan oleh ViewMatcher. Aksi tersebut bisa berupa salah satu aksi yang bisa dilakukan pada tampilan, misalnya klik. Misalnya:

  ```
  .perform(click())
  ```

- *ViewAssertions:* Ekspresi ViewAssertion memungkinkan Anda menyatakan atau memeriksa keadaan tampilan yang ditemukan oleh ViewMatcher. Misalnya:

  ```
  .check(matches(isDisplayed()))
  ```

Umumnya Anda harus mengombinasikan ViewMatcher dan ViewAction dalam pernyataan tunggal, diikuti oleh ekspresi ViewAssertion dalam pernyataan terpisah atau disertakan dalam pernyataan yang sama.

Anda bisa melihat cara kerja ketiga ekspresi dalam pernyataan berikut, yang mengombinasikan ViewMatcher untuk menemukan tampilan, ViewAction untuk melakukan aksi, dan ViewAssertion untuk memeriksa apakah hasil aksi cocok dengan pernyataan:

```
onView(withId(R.id.my_view))           // withId(R.id.my_view) is a ViewMatcher
        .perform(click())              // click() is a ViewAction
        .check(matches(isDisplayed())); // matches(isDisplayed()) is a ViewAssertion
```

Mengapa kerangka kerja Hamcrest berguna untuk pengujian? Sebuah pernyataan sederhana, misalnya `assert (x == y)` , memungkinkan Anda menyatakan selama pengujian bahwa ketentuan tertentu harus bernilai true. Jika ketentuan bernilai false, maka pengujian gagal. Namun, pernyataan sederhana tidak menyediakan pesan kesalahan yang berguna. Dengan sekelompok pernyataan, Anda bisa menghasilkan pesan kesalahan yang lebih berguna, namun hal ini akan menyebabkan ledakan jumlah pernyataan.

Dengan kerangka kerja Hamcrest, dimungkinkan mendefinisikan operasi yang memerlukan matcher sebagai argumen dan mengembalikannya sebagai hasil, sehingga menghasilkan tata bahasa yang bisa menghasilkan banyak sekali kemungkinan ekspresi matcher dari sedikit matcher primitif.

Untuk tutorial Hamcrest, lihat Tutorial Hamcrest. Untuk rangkuman singkat ekspresi matcher Hamcrest, lihat daftar rujukan Espresso.

Menguji AdapterView

Dalam AdapterView seperti spinner, tampilan diisi dengan tampilan anak secara dinamis pada waktu proses. Jika tampilan target yang ingin Anda uji ada dalam spinner, metode `onView()` mungkin tidak akan bekerja karena hanya subset tampilan yang bisa dimuat dalam hierarki tampilan saat ini.

Espresso menangani hal ini dengan menyediakan metode `onData()` terpisah, yang dapat memuat item adapter terlebih dulu dan membuatnya difokus sebelum mengoperasikan padanya atau salah satu tampilan anaknya. Metode `onData()` menggunakan objek DataInteraction dan metodenya, misalnya `atPosition()` , `check()` , dan `perform()` untuk mengakses tampilan target. Espresso menangani pemuatan elemen tampilan target ke dalam hierarki tampilan saat ini, pengguliran ke tampilan anak target, dan penempatan tampilan tersebut ke dalam fokus.

Misalnya, pernyataan `onView()` dan `onData()` berikut akan menguji klik item spinner:

1. Temukan dan klik spinner itu sendiri (pengujian harus mengeklik spinner itu sendiri terlebih dulu agar dapat mengeklik item lainnya dalam spinner):

   ```
   onView(withId(R.id.spinner_simple)).perform(click());
   ```

2. Temukan kemudian klik item dalam spinner yang cocok dengan *semua* ketentuan berikut:
   - Item yang berupa `String`
   - Item yang setara dengan String "Americano"

     ```
     onData(allOf(is(instanceOf(String.class)),
                          is("Americano"))).perform(click());
     ```

Seperti yang bisa Anda lihat dalam pernyataan di atas, ekspresi matcher bisa dikombinasikan untuk membuat ekspresi maksud yang fleksibel:

- `allOf` : Menghasilkan kecocokan jika objek yang diperiksa cocok dengan *semua* matcher yang ditetapkan. Anda bisa menggunakan `allOf()` untuk mengombinasikan beberapa matcher, misalnya `containsString()` dan `instanceOf()` .
- `is` : Hamcrest berusaha keras untuk membuat keterbacaan pengujian Anda setinggi mungkin. Matcher `is` adalah wrapper yang tidak menambahkan perilaku ekstra pada matcher dasar, melainkan membuat kode pengujian Anda menjadi lebih terbaca.
- `instanceOf` : Menyebabkan kecocokan jika objek yang diperiksa merupakan instance dari tipe yang ditetapkan; dalam hal ini adalah sebuah string. Kecocokan ini ditentukan dengan memanggil metode Class.isInstance(Object), dengan meneruskan objek yang akan diperiksa.

Contoh berikut mengilustrasikan cara menguji spinner menggunakan kombinasi metode `onView()` dan `onData()` :

```
@RunWith(AndroidJUnit4.class)
public class SpinnerSelectionTest {

   @Rule
   public ActivityTestRule mActivityRule = new ActivityTestRule<>(
            MainActivity.class);

   @Test
   public void iterateSpinnerItems() {
      String[] myArray = mActivityRule.getActivity().getResources()
                              .getStringArray(R.array.labels_array);

      // Iterate through the spinner array of items.
      int size = myArray.length;
      for (int i=0; i<size; i++) {
         // Find the spinner and click on it.
         onView(withId(R.id.label_spinner)).perform(click());
         // Find the spinner item and click on it.
         onData(is(myArray[i])).perform(click());
         // Find the button and click on it.
         onView(withId(R.id.button_main)).perform(click());
         // Find the text view and check that the spinner item
         // is part of the string.
         onView(withId(R.id.text_phonelabel))
                    .check(matches(withText(containsString(myArray[i]))));
      }
   }
}
```

Pengujian mengeklik setiap item spinner dari atas ke bawah, yang akan memeriksa apakah item muncul di bidang teks. Berapa pun banyaknya item spinner yang didefinisikan dalam larik, atau bahasa apa pun yang digunakan untuk item spinner—pengujian akan menjalankan semuanya dan memeriksa keluarannya terhadap larik.

Berikut ini adalah keterangan langkah-demi-langkah pengujian di atas:

1. Metode `iterateSpinnerItems()` dimulai dengan mendapatkan larik yang digunakan untuk item spinner:

   ```
   public void iterateSpinnerItems() {
   String[] myArray =
        mActivityRule.getActivity().getResources()
        .getStringArray(R.array.labels_array);
   ...
   ```

   Dalam pernyataan di atas, pengujian mengakses larik aplikasi (dengan ID `labels_array`) dengan membuat konteks menggunakan metode `getActivity()` kelas ActivityTestRule, dan mendapatkan instance sumber daya dalam paket aplikasi dengan menggunakan `getResources()`.

2. Menggunakan panjang (`size`) larik, loop `for` berulang pada setiap item spinner.

   ```
   ...
   int size = myArray.length;
   for (int i=0; i<size; i++) {
   // Find the spinner and click on it.
   ...
   ```

3. The `onView()` statement within the `for` loop finds the spinner and clicks on it. The test must click the spinner itself in order click any item in the spinner:

   ```
   ...
   // Find the spinner and click on it.
   onView(withId(R.id.label_spinner)).perform(click());
   ...
   ```

4. The `onData()` statement finds and clicks a spinner item:

```
...
// Find the spinner item and click on it.
onData(is(myArray[i])).perform(click());
...
```

Spinner diisi dari larik `myArray` , sehingga `myArray[i]` menyatakan elemen spinner dari larik. Karena loop `for` mengulangi `for (int i=0; i<size; i++)` , loop akan melakukan klik pada setiap elemen spinner (`myArray[i])` yang ditemukannya.

5. Pernyataan `onView()` terakhir menemukan tampilan teks (`text_phonelabel`) dan memeriksa apakah item spinner tersebut adalah bagian dari string:

```
...
onView(withId(R.id.text_phonelabel))
     .check(matches(withText(containsString(myArray[i]))));
...
```

Menggunakan RecyclerViewActions untuk Tampilan Recycler

RecyclerView berguna bila Anda memiliki sekumpulan data dengan elemen yang berubah pada waktu proses berdasarkan aksi pengguna atau kejadian jaringan. RecyclerView adalah komponen UI yang didesain untuk merender sekumpulan data, dan bukan subkelas dari AdapterView melainkan ViewGroup. Ini berarti Anda tidak bisa menggunakan `onData()` , yang khusus untuk AdapterView, untuk berinteraksi dengan item daftar.

Akan tetapi, ada kelas bernama RecyclerViewActions yang mengekspos API kecil untuk beroperasi pada RecyclerView. Misalnya, pengujian berikut mengeklik item dari daftar berdasarkan posisi:

```
onView(withId(R.id.recyclerView))
      .perform(RecyclerViewActions.actionOnItemAtPosition(0, click()));
```

Kelas pengujian berikut memperagakan cara menggunakan RecyclerViewActions untuk menguji RecyclerView. Aplikasi memungkinkan Anda menggulir daftar kata. Bila Anda mengeklik kata, seperti **Word 15**, kata dalam daftar akan berubah menjadi "Clicked! Word 15":

```
@RunWith(AndroidJUnit4.class)
public class RecyclerView {

   @Rule
   public ActivityTestRule<MainActivity> mActivityTestRule =
            new ActivityTestRule<>(MainActivity.class);

   @Test
   public void recyclerViewTest() {
      ViewInteraction recyclerView = onView(
            allOf(withId(R.id.recyclerview), isDisplayed()));
      recyclerView.perform(actionOnItemAtPosition(15, click()));

      ViewInteraction textView = onView(
            allOf(withId(R.id.word), withText("Clicked! Word 15"),
                 childAtPosition(
                         childAtPosition(
                                 withId(R.id.recyclerview),
                                 11),
                         0),
                 isDisplayed()));
      textView.check(matches(withText("Clicked! Word 15")));

   }

   private static Matcher<View> childAtPosition(
            final Matcher<View> parentMatcher, final int position) {

      return new TypeSafeMatcher<View>() {
         @Override
         public void describeTo(Description description) {
            description.appendText("Child at position "
                                        + position + " in parent ");
            parentMatcher.describeTo(description);
         }

         @Override
         public boolean matchesSafely(View view) {
            ViewParent parent = view.getParent();
            return parent instanceof ViewGroup &&
                                  parentMatcher.matches(parent)
                                  && view.equals(((ViewGroup)
                                  parent).getChildAt(position));
         }
      };
   }
}
```

Pengujian menggunakan objek recyclerView kelas ViewInteraction, yang merupakan antarmuka utama untuk melakukan tindakan atau pernyataan pada tampilan, yang menyediakan metode `check()` dan `perform()` . Setiap interaksi dikaitkan dengan tampilan yang diidentifikasi oleh matcher tampilan:

- Kode di bawah ini menggunakan metode `perform()` dan metode actionOnItemAtPosition() kelas RecyclerViewActions untuk menggulir ke posisi (15) dan mengeklik item:

  ```
  ViewInteraction recyclerView = onView(
        allOf(withId(R.id.recyclerview), isDisplayed()));
  recyclerView.perform(actionOnItemAtPosition(15, click()));
  ```

- Kode di bawah ini memeriksa untuk mengetahui apakah item yang diklik cocok dengan pernyataan bahwa kode tersebut harus `"Clicked! Word 15"` :

```
      ViewInteraction textView = onView(
            allOf(withId(R.id.word), withText("Clicked! Word 15"),
                 childAtPosition(
                        childAtPosition(
                               withId(R.id.recyclerview),
                               11),
                        0),
                 isDisplayed()));
      textView.check(matches(withText("Clicked! Word 15")));
```

- Kode di atas menggunakan sebuah metode bernama `childAtPosition()`, yang didefinisikan sebagai `Matcher` khusus:

```
   private static Matcher<View> childAtPosition(
         final Matcher<View> parentMatcher, final int position) {
      // TypeSafeMatcher() returned
      ...
      }
```

Matcher khusus memperluas kelas TypeSaveMatcher abstrak dan mengharuskan Anda mengimplementasikan yang berikut ini:

- Metode `matchesSafely()` untuk mendefinisikan cara memeriksa tampilan dalam RecyclerView.
- Metode `describeTo()` untuk mendefinisikan cara Espresso menjelaskan matcher keluaran dalam panel Run di bagian bawah Android Studio jika terjadi kegagalan.

```
      ...
      // TypeSafeMatcher() returned
      return new TypeSafeMatcher<View>() {
         @Override
         public void describeTo(Description description) {
            description.appendText("Child at position "
                                       + position + " in parent ");
            parentMatcher.describeTo(description);
         }

         @Override
         public boolean matchesSafely(View view) {
            ViewParent parent = view.getParent();
            return parent instanceof ViewGroup &&
                                    parentMatcher.matches(parent)
                                    && view.equals(((ViewGroup)
                                    parent).getChildAt(position));
         }
      };
   }
}
```

Merekam pengujian

Sebuah fitur Android Studio (di versi 2.2 dan lebih baru) memungkinkan Anda *merekam* pengujian Espresso, dengan membuat pengujian secara otomatis.

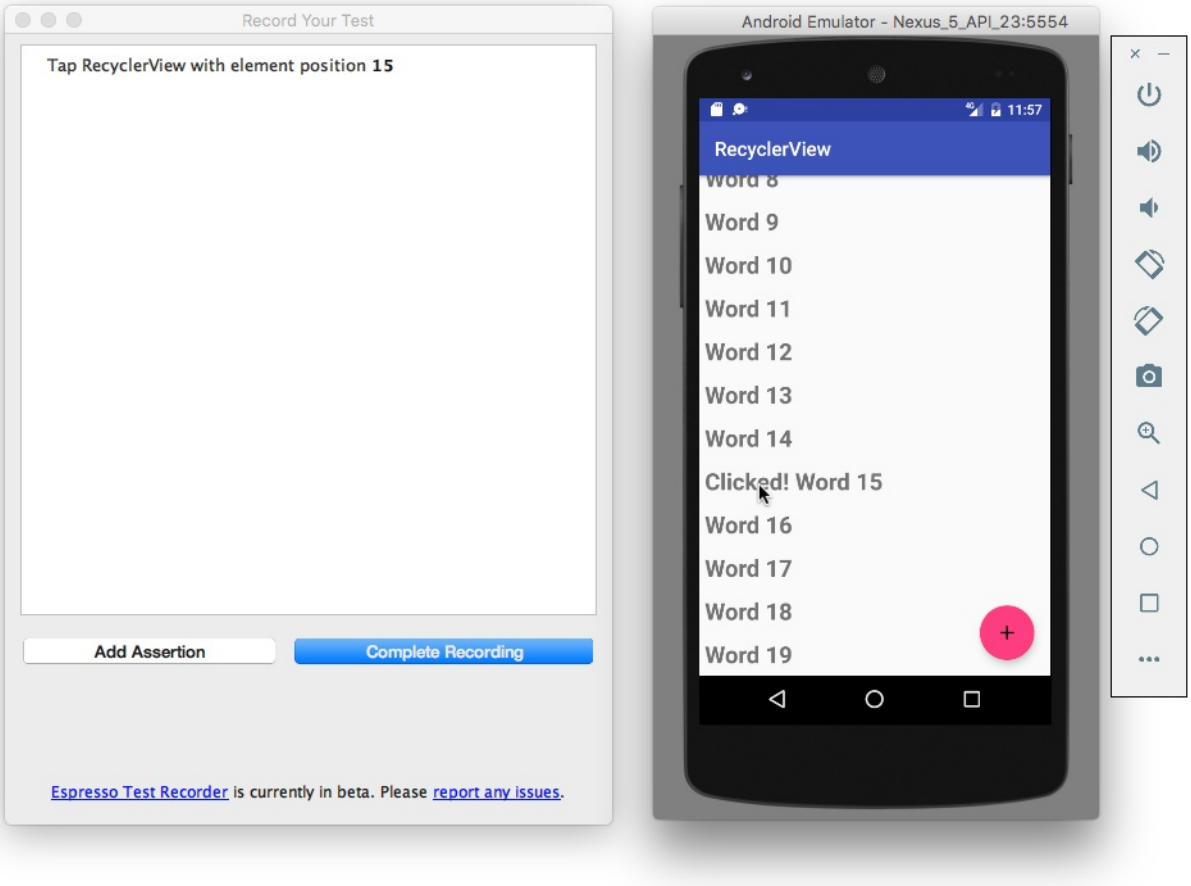
Setelah memilih untuk merekam pengujian, gunakan aplikasi Anda seperti halnya pengguna biasa. Saat Anda mengeklik UI aplikasi, kode pengujian yang bisa diedit akan dibuatkan untuk Anda. Tambahkan pernyataan untuk memeriksa apakah tampilan berisi nilai tertentu.

Anda bisa merekam beberapa interaksi sekaligus bersama UI dalam satu sesi perekaman. Anda juga bisa merekam beberapa pengujian dan mengedit pengujian untuk melakukan tindakan lainnya, menggunakan kode yang direkam sebagai cuplikan untuk disalin, ditempel, dan diedit.

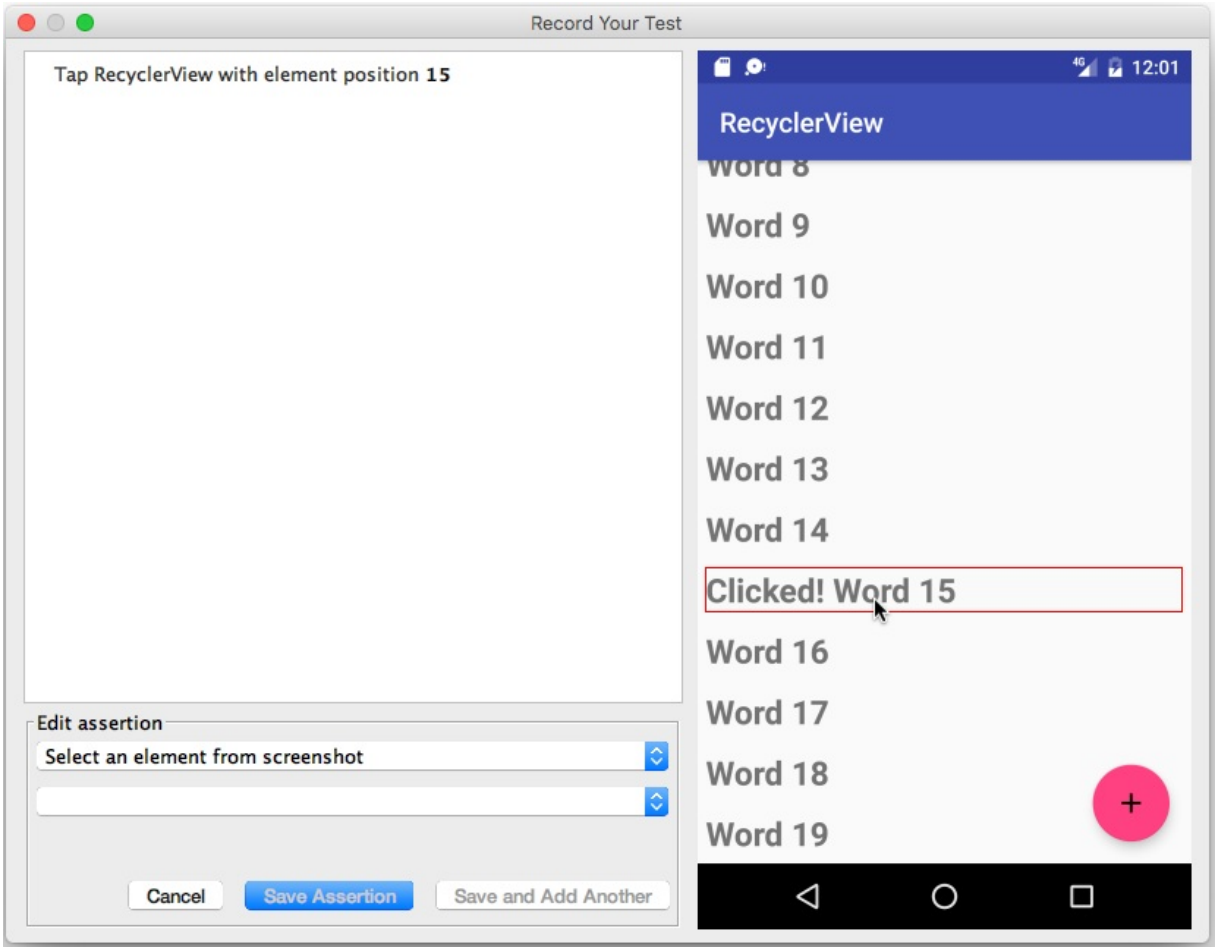
Ikuti langkah-langkah ini untuk merekam pengujian, dengan menggunakan aplikasi RecyclerView app sebagai contoh:

Android Studio Project: RecyclerView

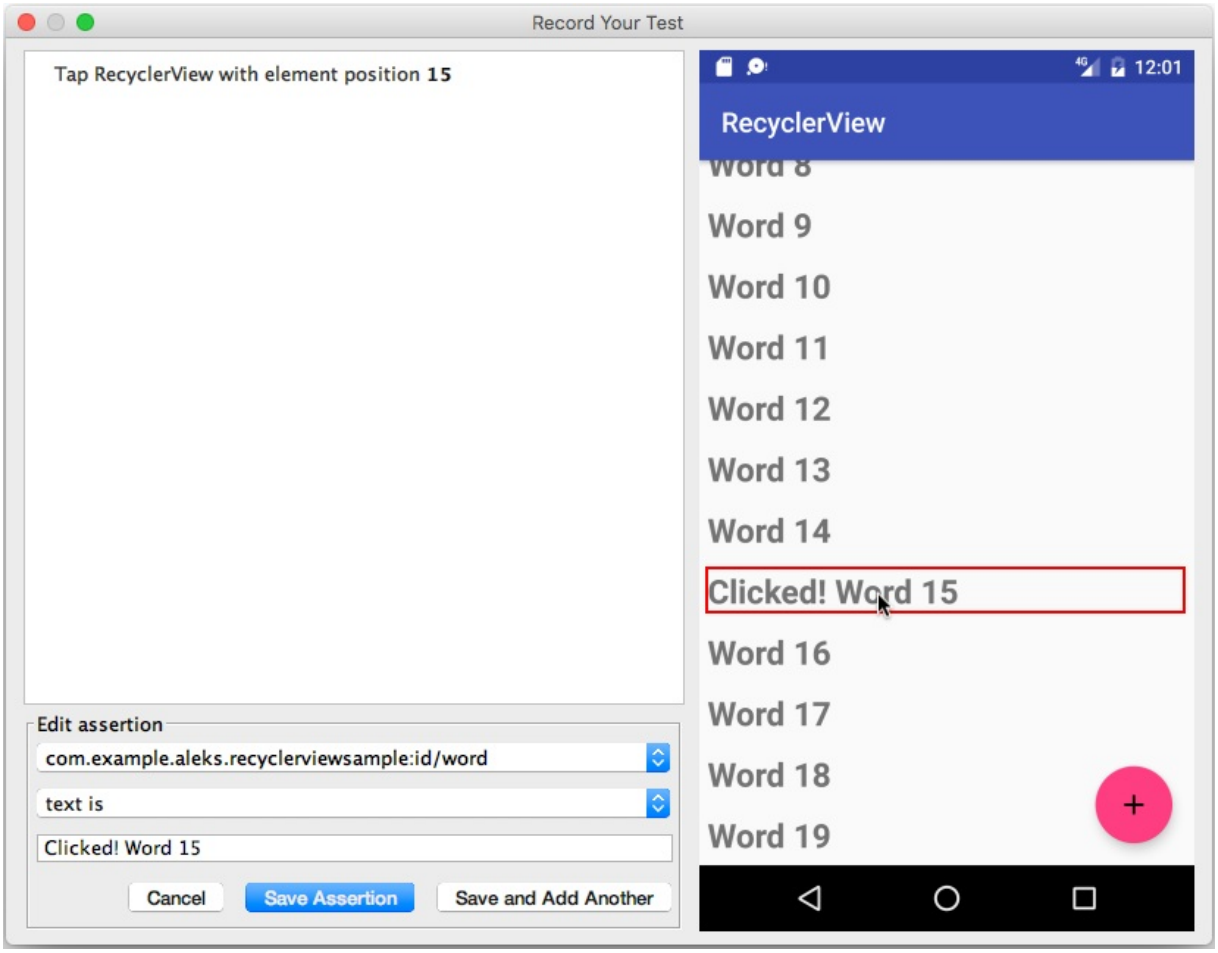
1. Pilih **Run > Record Espresso Test**, pilih target penerapan Anda (emulator atau perangkat) dan klik **OK**.
2. Berinteraksilah dengan UI untuk melakukan apa yang ingin Anda uji. Dalam hal ini, gulir daftar kata di aplikasi pada emulator atau perangkat, dan ketuk **Word 15**. Jendela Record Your Test akan menampilkan aksi yang direkam ("Tap RecyclerView with element position 15").



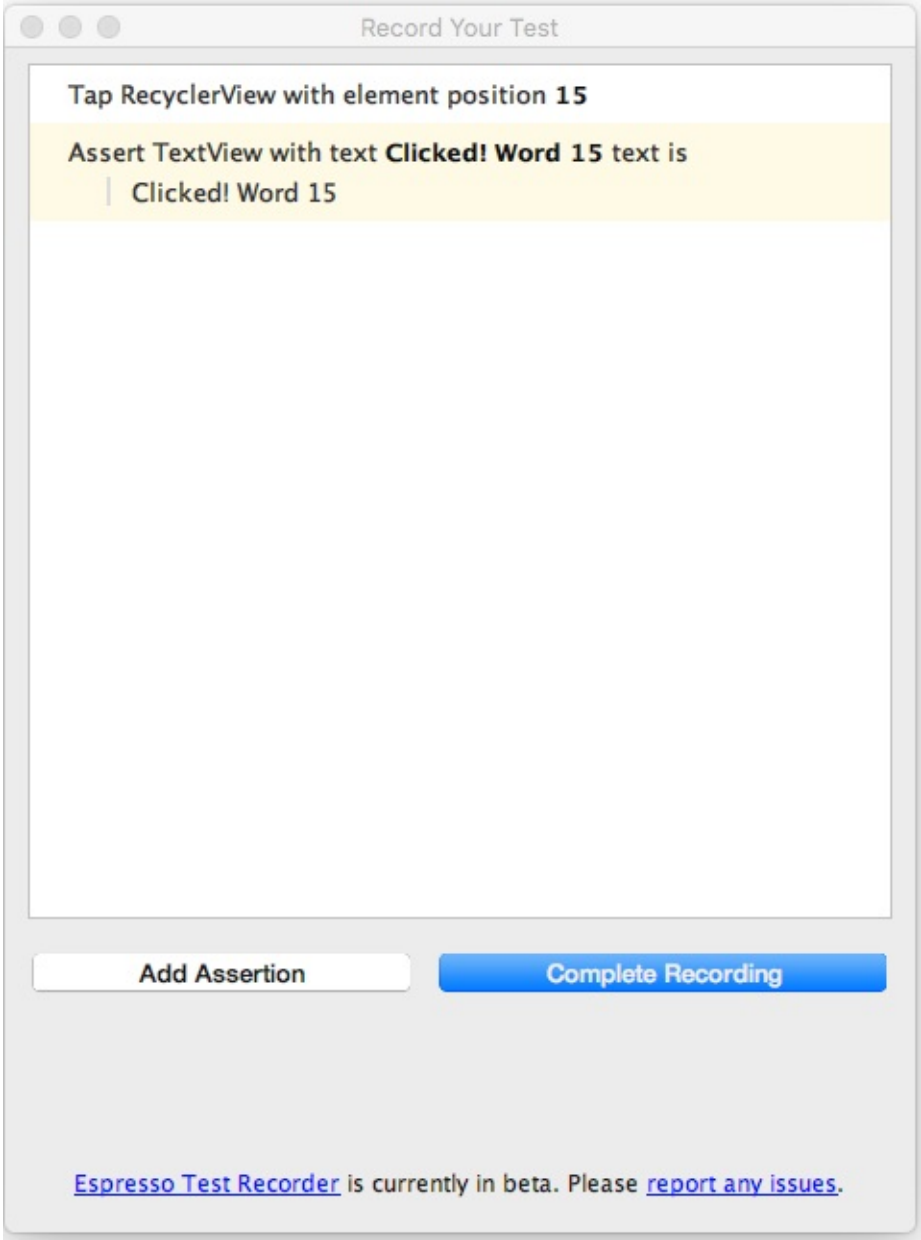
3. Klik **Add Assertion** dalam jendela Record Your Test. Tangkapan layar UI aplikasi akan muncul di panel kanan jendela. Pilih **Clicked! Word 15** di tangkapan layar sebagai elemen UI yang ingin Anda periksa. Langkah ini akan menghasilkan sebuah pernyataan untuk tampilan elemen yang dipilih.



4. Pilih **text is** dari menu tarik-turun kedua, dan masukkan teks yang ingin Anda lihat dalam elemen UI tersebut.



5. Klik **Save Assertion**, kemudian klik **Complete Recording**.



6. Dalam dialog yang muncul, Anda bisa mengedit nama pengujian, atau menerima nama yang disarankan (seperti **MainActivityTest**).
7. Android Studio mungkin menampilkan permintaan untuk menambahkan lebih banyak dependensi ke file Gradle Build Anda. Klik **Yes** untuk menambahkan dependensi.

Menggunakan UI Automator untuk pengujian yang mencakup beberapa aplikasi

UI Automator adalah serangkaian API yang bisa membantu Anda memverifikasi perilaku interaksi yang tepat di antara berbagai aplikasi pengguna, atau antara aplikasi pengguna dan aplikasi sistem. UI Automator memungkinkan Anda berinteraksi dengan elemen yang terlihat pada suatu perangkat. Alat (bantu) penampil menyediakan antarmuka visual untuk memeriksa hierarki layout dan menampilkan properti komponen UI yang tampak di latar depan perangkat. Seperti Espresso, UI Automator memiliki akses ke informasi interaksi sistem sehingga Anda bisa memantau semua interaksi yang dimiliki sistem Android dengan aplikasi.

Untuk menggunakan UI Automator, Anda harus sudah menyetel lingkungan pengujian Anda dengan cara yang sama untuk Espresso:

- Pasang Android Support Repository dan Android Testing Support Library.
- Tambahkan dependensi berikut ke file build.gradle proyek:

```
androidTestCompile
     'com.android.support.test.uiautomator:uiautomator-v18:2.1.2'
```

Gunakan UI Automator Viewer untuk Memeriksa UI pada perangkat

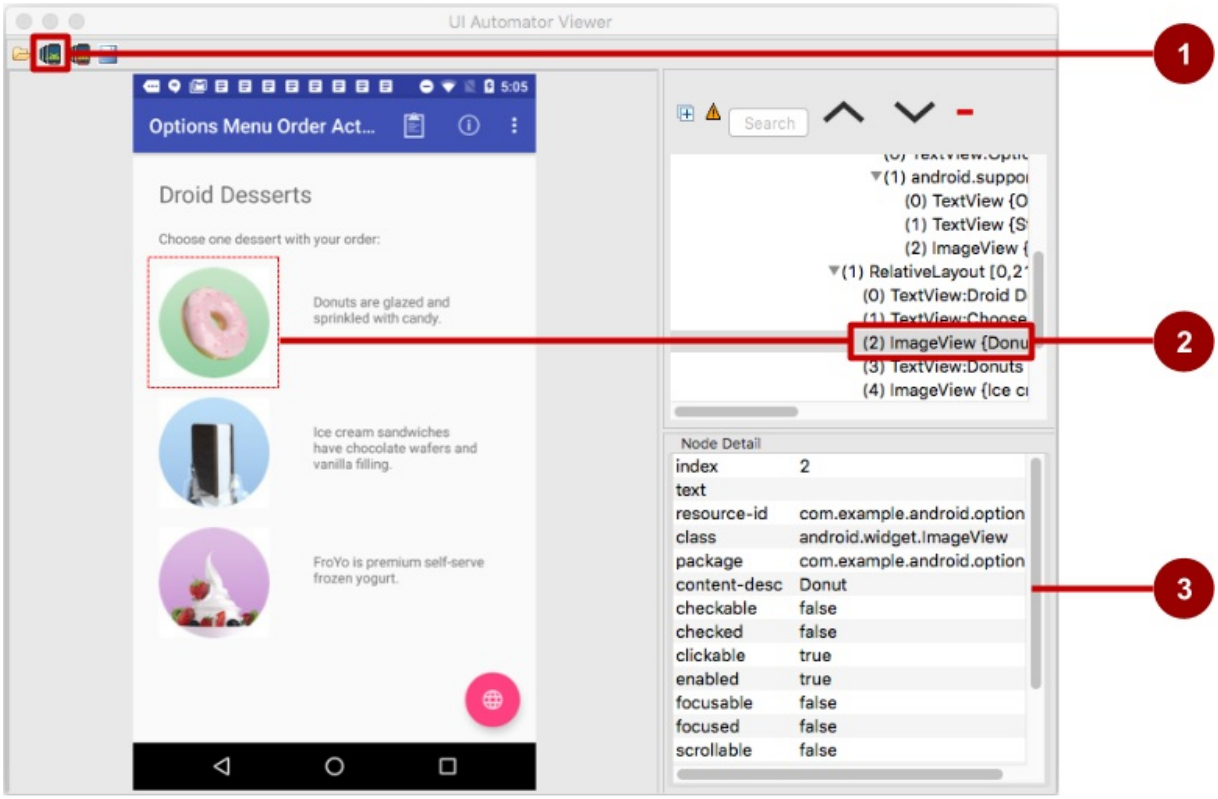
UI Automator Viewer (uiautomatorviewer) menyediakan antarmuka visual yang nyaman untuk memeriksa hierarki layout dan melihat properti komponen UI yang tampak di latar depan perangkat.

Untuk meluncurkan alat (bantu) uiautomatorviewer, ikuti langkah-langkah ini:

1. Pasang kemudian luncurkan aplikasi pada perangkat fisik seperti ponsel cerdas.
2. Hubungkan perangkat ke komputer development Anda.
3. Buka jendela terminal dan arahkan ke direktori **/tools/**. Untuk menemukan jalur tertentu, pilih **Preferences** di Android Studio, dan klik **Appearance & Behavior > System Settings> Android SDK**. Jalur lengkap untuk akan muncul dalam kotak Android SDK Location di bagian atas layar.
4. Jalankan alat (bantu) dengan perintah ini: **uiautomatorviewer**

Untuk memastikan bahwa pengujian UI Automator bisa mengakses elemen UI aplikasi, periksa apakah elemen memiliki label teks yang terlihat, nilai-nilai android:contentDescription, atau keduanya. Anda bisa menampilkan properti elemen UI dengan mengikuti langkah-langkah ini (lihat gambar di bawah ini):

1. Setelah meluncurkan uiautomatorviewer, tampilan akan kosong. Klik tombol **Device Screenshot**.
2. Arahkan ke atas elemen UI di cuplikan pada panel sebelah kiri untuk melihat elemen dalam hierarki layout pada panel kanan atas.
3. Atribut layout dan properti lainnya untuk elemen UI akan muncul di panel kanan bawah. Gunakan informasi ini untuk membuat pengujian yang memilih elemen UI yang cocok dengan atribut atau properti tertentu yang terlihat.



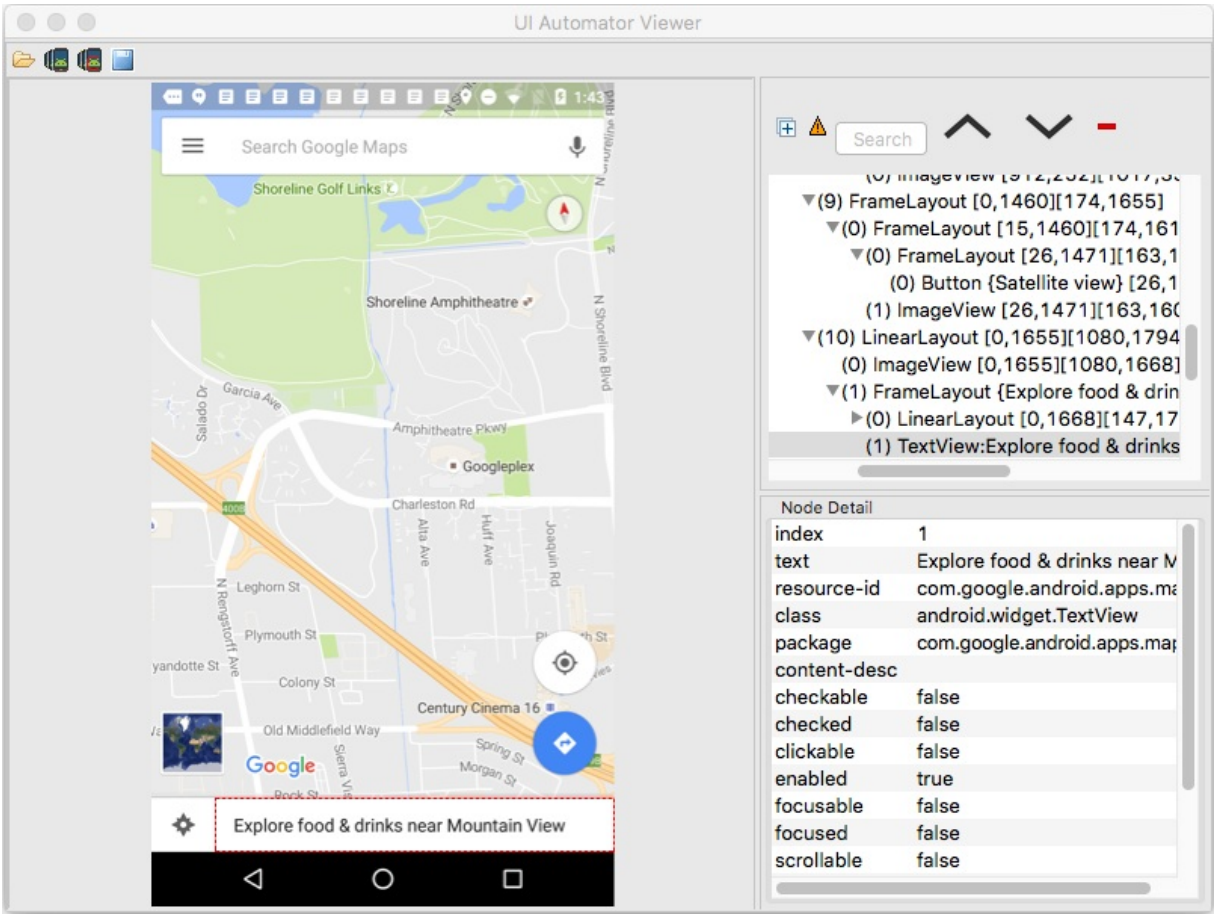
Dalam gambar di atas:

1. Tombol **Device Screenshot**
2. Komponen yang dipilih dalam cuplikan dan dalam hierarki layout

3. Properti untuk komponen yang dipilih

Dalam aplikasi yang ditampilkan di atas, tombol aksi mengambang warna merah meluncurkan aplikasi Maps. Ikuti langkah-langkah ini untuk menguji aksi yang dilakukan tombol aksi mengambang:

1. Ketuk tombol aksi mengambang.
2. Kode untuk tombol akan membuat maksud implisit meluncurkan aplikasi Maps.
3. Klik tombol **Device Screenshot** untuk melihat hasil maksud implisit (layar Maps).



Dengan menggunakan viewer, Anda bisa menentukan elemen UI mana yang bisa diakses untuk kerangka kerja UI Automator.

Persiapan: Memastikan bahwa elemen UI bisa diakses

Kerangka kerja UI Automator bergantung pada fitur aksesibilitas kerangka kerja Android untuk mencari elemen UI individual. Implementasikan properti tampilan seperti berikut:

- Sertakan atribut android:contentDescription dalam layout XML untuk memberi label ImageButton, ImageView, CheckBox, dan kontrol masukan pengguna lainnya. Yang berikut ini menampilkan atribut `android:contentDescription` yang ditambahkan ke `RadioButton` dengan menggunakan sumber daya string yang sama dengan yang digunakan untuk atribut `android:text` :

```
<RadioButton
  android:id="@+id/sameday"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="@string/same_day_messenger_service"
  android:contentDescription="@string/same_day_messenger_service"
  android:onClick="onRadioButtonClicked"/>
```

Tip: Anda bisa membuat kontrol masukan menjadi lebih mudah diakses oleh orang yang memiliki gangguan penglihatan, dengan menggunakan atribut layout XML `android:contentDescription` . Teks dalam atribut ini tidak muncul pada layar, namun jika pengguna mengaktifkan layanan aksesibilitas yang menyediakan perintah yang bisa didengar, maka bila pengguna mengarahkan ke kontrol itu, teksnya akan dibacakan.

- Sediakan atribut android:hint untuk bidang EditText (selain `android:contentDescription`, yang berguna untuk layanan aksesibilitas). Dengan bidang EditText, UI Automator mencari atribut `android:hint` .
- Hubungkan atribut `android:hint` dengan ikon grafik apa pun yang digunakan oleh kontrol yang menyediakan masukan kepada pengguna (misalnya, status atau informasi keadaan).

Sebagai developer, Anda harus mengimplementasikan optimalisasi minimum di atas untuk mendukung pengguna serta UI Automator.

Membuat kelas pengujian

Kelas pengujian UI Automator umumnya mengikuti model pemrograman ini:

1. **Akses perangkat yang akan diuji:** Instance kelas InstrumentRegistry yang berisi referensi ke instrumentasi yang berjalan dalam proses bersama dengan argumen instrumentasi. Kelas ini juga menyediakan cara yang mudah bagi pemanggil untuk mendapatkan instrumentasi, konteks aplikasi, dan Bundle argumen instrumentasi. Anda bisa mendapatkan objek UiDevice dengan memanggil metode getInstance() dan meneruskan objek Instrumentasi kepadanya— `InstrumentationRegistry.getInstrumentation()` —sebagai argumen. Misalnya:

   ```
   mDevice = UiDevice.getInstance(InstrumentationRegistry.getInstrumentation());
   ```

2. **Akses elemen UI yang ditampilkan pada perangkat:** Dapatkan UiObject dengan memanggil metode findObject(). Misalnya:

   ```
   UiObject okButton = mDevice.findObject(new UiSelector()
       .text("OK"))
       .className("android.widget.Button"));
   ```

3. **Lakukan aksi:** Simulasikan interaksi pengguna tertentu untuk dilakukan pada elemen UI itu dengan memanggil metode UiObject. Misalnya:

   ```
   if(okButton.exists() && okButton.isEnabled()) {
       okButton.click();
   }
   ```

   Anda bisa memanggil setText() untuk mengedit bidang teks, atau performMultiPointerGesture() untuk menyimulasikan isyarat multisentuh.

   Anda bisa mengulang langkah 2 dan 3 sesuai kebutuhan untuk menguji interaksi pengguna lebih kompleks yang melibatkan beberapa komponen UI atau urutan tindakan pengguna.

4. **Verifikasi hasilnya:** Periksa apakah UI mencerminkan keadaan atau perilaku yang diharapkan setelah interaksi pengguna ini dilakukan. Anda bisa menggunakan metode JUnit Assert untuk menguji apakah komponen UI dalam aplikasi mengembalikan hasil yang diharapkan. Misalnya:

   ```
   UiObject result = mDevice.findObject(By.res(CALC_PACKAGE, "result"));
   assertEquals("5", result.getText());
   ```

Mengakses perangkat

Kelas UiDevice menyediakan metode untuk mengakses dan memanipulasi keadaan perangkat. Tidak seperti Espresso, UI Automator bisa memverifikasi perilaku interaksi yang tepat di antara berbagai aplikasi pengguna, atau antara aplikasi pengguna dan aplikasi sistem. UI Automator memungkinkan Anda berinteraksi dengan elemen yang terlihat pada suatu perangkat. Dalam pengujian, Anda bisa memanggil metode UiDevice untuk memeriksa keadaan beragam properti,

misalnya orientasi saat ini atau ukuran tampilan. Pengujian Anda bisa menggunakan objek UiDevice untuk melakukan tindakan tingkat perangkat, misalnya memaksa perangkat ke rotasi tertentu, menekan tombol perangkat keras D-pad, dan menekan tombol Beranda.

Praktik yang baik adalah memulai pengujian Anda dari layar utama perangkat. Dari layar utama, Anda bisa memanggil metode yang disediakan oleh UI Automator API untuk memilih dan berinteraksi dengan elemen UI tertentu. Cuplikan kode berikut menampilkan bagaimana pengujian Anda bisa mendapatkan instance UiDevice, menyimulasikan penekanan tombol Beranda, dan meluncurkan aplikasi:

```
import org.junit.Before;
import android.support.test.runner.AndroidJUnit4;
import android.support.test.uiautomator.UiDevice;
import android.support.test.uiautomator.By;
import android.support.test.uiautomator.Until;
...

@RunWith(AndroidJUnit4.class)
@SdkSuppress(minSdkVersion = 18)
public class ChangeTextBehaviorTest {

    private static final String BASIC_SAMPLE_PACKAGE
            = "com.example.android.testing.uiautomator.BasicSample";
    private static final int LAUNCH_TIMEOUT = 5000;
    private static final String STRING_TO_BE_TYPED = "UiAutomator";
    private UiDevice mDevice;

    @Before
    public void startMainActivityFromHomeScreen() {
        // Initialize UiDevice instance
        mDevice =
          UiDevice.getInstance(InstrumentationRegistry.getInstrumentation());

        // Start from the home screen
        mDevice.pressHome();

        // Wait for launcher
        final String launcherPackage = mDevice.getLauncherPackageName();
        assertThat(launcherPackage, notNullValue());
        mDevice.wait(Until.hasObject(By.pkg(launcherPackage).depth(0)),
                LAUNCH_TIMEOUT);

        // Launch the app
        Context context = InstrumentationRegistry.getContext();
        final Intent intent = context.getPackageManager()
                .getLaunchIntentForPackage(BASIC_SAMPLE_PACKAGE);
        // Clear out any previous instances
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
        context.startActivity(intent);

        // Wait for the app to appear
        mDevice.wait(Until.hasObject(By.pkg(BASIC_SAMPLE_PACKAGE).depth(0)),
                LAUNCH_TIMEOUT);
    }
}
```

Anotasi `@SdkSuppress(minSdkVersion = 18)` memastikan bahwa pengujian hanya akan berjalan pada perangkat dengan Android 4.3 (API level 18) atau yang lebih tinggi, sebagaimana disyaratkan oleh kerangka kerja UI Automator.

Mengakses elemen UI

Gunakan metode findObject() kelas UiObject untuk mengambil instance UiObject yang menyatakan elemen UI yang cocok dengan kriteria pemilih. Untuk mengakses elemen UI tertentu, gunakan kelas UiSelector yang menyatakan kueri untuk elemen tertentu dalam UI yang ditampilkan saat ini.

Anda bisa menggunakan kembali instance UiObject yang dibuat di bagian lain pengujian aplikasi. Kerangka kerja pengujian UI Automator akan menelusuri tampilan saat ini untuk menemukan suatu kecocokan setiap kali pengujian Anda menggunakan instance UiObject untuk mengeklik elemen UI elemen atau menjalankan kueri atas suatu atribut.

Yang berikut ini menampilkan bagaimana pengujian Anda akan membentuk instance UiObject dengan menggunakan `findObject()` bersama UiSelector yang menyatakan tombol **Cancel**, dan yang menyatakan tombol **OK**:

```
UiObject cancelButton = mDevice.findObject(new UiSelector()
        .text("Cancel"))
        .className("android.widget.Button"));
UiObject okButton = mDevice.findObject(new UiSelector()
        .text("OK"))
        .className("android.widget.Button"));
```

Jika lebih dari satu elemen yang cocok, elemen pertama yang cocok dalam hierarki layout (ditemukan dengan bergerak dari atas ke bawah, kiri ke kanan) akan dikembalikan sebagai target UiObject. Saat membentuk UiSelector, Anda bisa merangkai beberapa atribut dan properti untuk memperbaiki penelusuran. Jika tidak ditemukan elemen UI yang cocok, pengecualian (UiAutomatorObjectNotFoundException) akan dilontarkan.

Untuk menyarangkan beberapa instance UiSelector, gunakan metode childSelector() dengan kelas UiSelector. Misalnya, yang berikut ini menampilkan bagaimana pengujian Anda bisa menetapkan suatu penelusuran untuk menemukan ListView pertama dalam UI yang ditampilkan saat ini, kemudian menelusuri dalam ListView untuk menemukan komponen UI dengan atribut `android:text` `"List Item 14"` :

```
UiObject appItem = new UiObject(new UiSelector()
        .className("android.widget.ListView")
        .instance(1)
        .childSelector(new UiSelector()
        .text("List Item 14")));
```

Walaupun mungkin berguna untuk merujuk ke atribut `android:text` dari elemen ListView atau RecycleView karena tidak ada ID sumber daya (atribut `android:id`) untuk elemen semacam itu, sebaiknya gunakan ID sumber daya saat menetapkan pemilih, bukan atribut `android:text` atau `android:contentDescription` . Tidak semua elemen memiliki atribut teks (misalnya, ikon dalam bilah alat). Pengujian bisa gagal jika ada perubahan kecil pada teks komponen UI, dan pengujian tidak akan berguna bagi aplikasi yang diterjemahkan ke dalam bahasa lain karena pemilih teks Anda tidak akan cocok dengan sumber daya string yang diterjemahkan.

Menjalankan tindakan

Setelah pengujian mengambil objek UiObject, Anda bisa memanggil metode dalam kelas UiObject untuk melakukan interaksi pengguna pada komponen UI yang dinyatakan oleh objek itu. Misalnya, instance UiObject yang dibuat di bagian sebelumnya untuk tombol OK dan Cancel bisa digunakan untuk melakukan klik:

```
// Simulate a user-click on the OK button, if found.
if(okButton.exists() && okButton.isEnabled()) {
    okButton.click();
}
```

Anda bisa menggunakan metode UiObject untuk melakukan tindakan seperti:

- click(): Mengeklik di tengah lingkaran elemen UI yang terlihat.
- dragTo(): Menyeret objek ke koordinat arbitrer.
- setText(): Menyetel teks dalam bidang yang bisa diedit, setelah mengosongkan isinya. Sebaliknya, gunakan metode clearTextField() untuk mengosongkan teks yang ada dalam bidang yang bisa diedit.
- swipeUp(): Melakukan aksi gesek pada UiObject. Begitu juga, metode swipeDown(), swipeLeft(), dan swipeRight() akan melakukan tindakan yang sesuai.

Mengirim maksud atau meluncurkan aktivitas

Kerangka kerja pengujian UI Automator memungkinkan Anda mengirim Maksud atau meluncurkan Aktivitas tanpa menggunakan perintah shell, dengan mendapatkan objek Context melalui metode getContext(). Misalnya, yang berikut ini menampilkan bagaimana pengujian Anda bisa menggunakan Maksud untuk meluncurkan aplikasi yang diuji:

```
public void setUp() {
    ...
    // Launch a simple calculator app
    Context context = getInstrumentation().getContext();
    Intent intent = context.getPackageManager()
            .getLaunchIntentForPackage(CALC_PACKAGE);
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
            // Clear out any previous instances
    context.startActivity(intent);
    mDevice.wait(Until.hasObject(By.pkg(CALC_PACKAGE).depth(0)), TIMEOUT);
}
```

Melakukan tindakan atas suatu kumpulan

Gunakan kelas UiCollection jika Anda ingin menyimulasikan interaksi pengguna pada sekumpulan elemen UI (misalnya, judul lagu atau daftar email). Untuk membuat objek UiCollection, tetapkan UiSelector yang akan menelusuri kontainer UI atau wrapper elemen UI anak lainnya, misalnya grup layout yang berisi elemen UI anak.

Yang berikut ini menampilkan bagaimana pengujian Anda bisa menggunakan UiCollection untuk menyatakan album video yang ditampilkan dalam FrameLayout:

```
UiCollection videos = new UiCollection(new UiSelector()
        .className("android.widget.FrameLayout"));

// Retrieve the number of videos in this collection:
int count = videos.getChildCount(new UiSelector()
        .className("android.widget.LinearLayout"));

// Find a specific video and simulate a user-click on it
UiObject video = videos.getChildByText(new UiSelector()
        .className("android.widget.LinearLayout"), "Cute Baby Laughing");
video.click();

// Simulate selecting a checkbox that is associated with the video
UiObject checkBox = video.getChild(new UiSelector()
        .className("android.widget.Checkbox"));
if(!checkBox.isSelected()) checkbox.click();
```

Melakukan tindakan pada tampilan yang bisa digulir

Gunakan kelas UiScrollable untuk menyimulasikan pengguliran vertikal atau horizontal pada tampilan. Teknik ini berguna bila elemen UI diposisikan tidak tampak dan Anda perlu menggulirnya untuk menampilkannya. Misalnya, cuplikan kode berikut menampilkan cara menyimulasikan pengguliran ke bawah menu **Settings** dan pengeklikan opsi **About phone**:

```
UiScrollable settingsItem = new UiScrollable(new UiSelector()
        .className("android.widget.ListView"));
UiObject about = settingsItem.getChildByText(new UiSelector()
        .className("android.widget.LinearLayout"), "About phone");
about.click();
```

Memverifikasi hasilnya

Anda bisa menggunakan metode JUnit Assert untuk menguji apakah komponen UI dalam aplikasi mengembalikan hasil yang diharapkan. Misalnya, Anda bisa menggunakan assertFalse() untuk menyatakan bahwa suatu ketentuan bernilai false untuk menguji apakah syarat tersebut benar-benar menghasilkan nilai false. Gunakan assertEquals() untuk menguji apakah hasil angka titik-mengambang sama dengan pernyataan:

```
assertEquals("5", result.getText());
```

Yang berikut ini menampilkan bagaimana pengujian Anda bisa menemukan sejumlah tombol dalam aplikasi kalkulator, mengekliknya berurutan, kemudian memverifikasi apakah hasil ditampilkan benar:

```
private static final String CALC_PACKAGE = "com.myexample.calc";
public void testTwoPlusThreeEqualsFive() {
    // Enter an equation: 2 + 3 = ?
    mDevice.findObject(new UiSelector()
            .packageName(CALC_PACKAGE).resourceId("two")).click();
    mDevice.findObject(new UiSelector()
            .packageName(CALC_PACKAGE).resourceId("plus")).click();
    mDevice.findObject(new UiSelector()
            .packageName(CALC_PACKAGE).resourceId("three")).click();
    mDevice.findObject(new UiSelector()
            .packageName(CALC_PACKAGE).resourceId("equals")).click();

    // Verify the result = 5
    UiObject result = mDevice.findObject(By.res(CALC_PACKAGE, "result"));
    assertEquals("5", result.getText());
}
```

Menjalankan pengujian instrumentasi

Untuk menjalankan pengujian tunggal, klik-kanan (atau Control+klik) pengujian di Android Studio dan pilih **Run** dari menu munculan.

Untuk menguji metode dalam kelas pengujian, klik-kanan pada metode dalam file pengujian dan klik **Run**.

Untuk menjalankan semua pengujian di sebuah direktori, klik-kanan pada direktori dan pilih **Run tests**.

Android Studio akan menampilkan hasil pengujian dalam jendela Run.

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Gunakan Espresso untuk Menguji UI Anda

Ketahui selengkapnya

Dokumentasi Android Studio:

- Uji Aplikasi Anda

Dokumentasi Developer Android:

- Praktik Terbaik untuk Pengujian
- Memulai Pengujian
- Menguji UI untuk Aplikasi Tunggal—Espresso
- Menguji UI untuk Beberapa Aplikasi—UI Automator
- Membangun Pengujian Unit Instrumentasi
- Contoh Lanjutan Espresso
- Tutorial Hamcrest
- Hamcrest API dan Kelas Utilitas
- Test Support API

Android Testing Support Library:

- Dokumentasi Espresso
- Contoh Espresso
- Dasar-dasar Espresso
- Rujukan ringkas Espresso

Video

- Dukungan Pengujian Android - Pola Pengujian Android #1 (pengantar)
- Dukungan Pengujian Android - Pola Pengujian Android #2 (pencocokan tampilan onView)
- Dukungan Pengujian Android - Pola Pengujian Android #3 (tampilan onData dan adapter)

Lainnya:

- Blog Pengujian Google:
  - Pengujian Otomatis untuk UI Android
  - Ukuran Pengujian
- Atomic Object: "Espresso – Menguji RecyclerViews pada Posisi Tertentu"
- Stack Overflow: "Bagaimana menyatakan di dalam RecyclerView pada Espresso?"
- GitHub: Contoh Pengujian Android
- Google Codelabs: Codelab Pengujian Android
- Situs web JUnit
- Anotasi JUnit: Package org.junit

7.1: AsyncTask dan AsyncTaskLoader

Materi:

- Thread UI
- AsyncTask
- Penggunaan AsyncTask
- Contoh AsyncTask
- Mengeksekusi AsyncTask
- Membatalkan AsyncTask
- Keterbatasan AsyncTask
- Loader
- AsyncTaskLoader
- Penggunaan AsyncTaskLoader
- Praktik terkait
- Ketahui selengkapnya

Ada dua cara untuk melakukan pemrosesan latar belakang di Android: menggunakan kelas `AsyncTask` , atau menggunakan kerangka kerja `Loader` , yang menyertakan kelas `AsyncTaskLoader` yang menggunakan `AsyncTask` . Di sebagian besar situasi, Anda akan memilih kerangka kerja `Loader` , namun penting untuk mengetahui cara kerja `AsyncTask` , sehingga Anda bisa membuat pilihan yang bagus.

Dalam bab ini Anda akan mempelajari alasan pentingnya memproses beberapa tugas di latar belakang, di luar thread UI. Anda akan mempelajari cara menggunakan `AsyncTask` , bila *tidak* menggunakan `AsyncTask` , dan dasar-dasar penggunaan loader.

Thread UI

Bila aplikasi Android dimulai, aplikasi membuat *thread utama*, yang sering disebut *thread UI*. Thread UI mengirimkan kejadian ke widget antarmuka pengguna (UI) yang sesuai, dan ini merupakan tempat aplikasi Anda berinteraksi dengan komponen dari toolkit UI Android (komponen dari paket android.widget dan android.view).

Model thread Android memiliki dua aturan:

1. Jangan memblokir thread UI.

Thread UI perlu memberikan perhatiannya untuk menggambar UI dan menjaga aplikasi tetap responsif terhadap masukan pengguna. Jika semuanya terjadi di thread UI, operasi panjang seperti akses jaringan atau kueri database bisa memblokir seluruh UI. Dari perspektif pengguna, aplikasi tersebut akan mogok. Lebih buruk lagi, jika thread UI diblokir selama lebih dari beberapa detik (saat ini sekitar 5 detik) pengguna akan ditampilkan dialog "application not responding" (ANR). Pengguna bisa memutuskan untuk keluar dari aplikasi dan mencopot pemasangannya.

Untuk memastikan aplikasi Anda tidak memblokir thread UI:

- Selesaikan semua pekerjaan dalam waktu kurang dari 16 md untuk setiap layar UI.
- Jangan menjalankan tugas asinkron dan tugas lain yang berjalan lama pada thread UI. Sebagai gantinya, implementasikan tugas pada thread latar belakang menggunakan `AsyncTask` (untuk tugas singkat atau yang bisa disela) atau `AsyncTaskLoader` (untuk tugas berprioritas tinggi, atau tugas yang perlu melaporkan kembali ke pengguna atau UI). **2. Lakukan pekerjaan UI hanya pada thread UI.**

Jangan menggunakan thread latar belakang untuk memanipulasi UI Anda, karena toolkit UI Android bukan thread-safe.

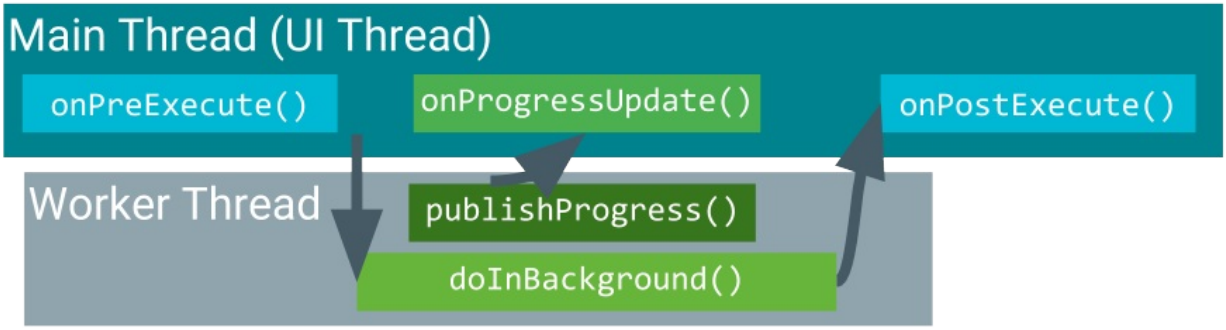
AsyncTask

Gunakan kelas `AsyncTask` untuk mengimplementasikan tugas asinkron yang berjalan lama di thread pekerja. (*Thread pekerja* adalah thread yang bukan thread utama atau thread UI.) `AsyncTask` memungkinkan Anda menjalankan operasi latar belakang dan mempublikasikan hasil di thread UI tanpa memanipulasi thread atau penangan.

Bila `AsyncTask` dieksekusi, maka akan melalui empat langkah:

1. `onPreExecute()` dipanggil di thread UI sebelum tugas dieksekusi. Langkah ini biasanya digunakan untuk mempersiapkan tugas, misalnya dengan menampilkan bilah kemajuan di UI.
2. `doInBackground(Params...)` dipanggil pada thread latar belakang segera setelah `onPreExecute()` selesai. Langkah ini menjalankan komputasi latar belakang, mengembalikan hasil, dan meneruskan hasilnya ke `onPostExecute()`. Metode `doInBackground()` juga bisa memanggil `publishProgress(Progress...)` untuk mempublikasikan satu atau beberapa unit kemajuan.
3. `onProgressUpdate(Progress...)` berjalan di thread UI setelah `publishProgress(Progress...)` dipanggil. Gunakan `onProgressUpdate()` untuk melaporkan suatu bentuk kemajuan ke thread UI sewaktu komputasi latar belakang dieksekusi. Misalnya, Anda bisa menggunakannya untuk meneruskan data guna menganimasikan bilah kemajuan atau menampilkan log di bidang teks.
4. `onPostExecute(Result)` berjalan di thread UI setelah komputasi latar belakang selesai.

Untuk detail selengkapnya mengenai metode ini, lihat `referensi AsyncTask`. Di bawah ini adalah diagram urutan pemanggilan.



Penggunaan AsyncTask

Untuk menggunakan kelas `AsyncTask`, definisikan subkelas `AsyncTask` yang menggantikan metode `doInBackground(Params...)` (dan biasanya juga metode `onPostExecute(Result)`). Bagian ini menjelaskan parameter dan penggunaan `AsyncTask`, kemudian menampilkan contoh lengkap.

Parameter AsyncTask

Di subkelas `AsyncTask`, sediakan tipe data untuk tiga jenis parameter.

- "Params" menetapkan tipe parameter yang diteruskan ke `doInBackground()` sebagai larik.
- "Progress" menetapkan tipe parameter yang diteruskan ke `publishProgress()` di thread latar belakang. Parameter ini selanjutnya diteruskan ke metode `onProgressUpdate()` di thread utama.
- "Result" menetapkan tipe parameter yang dikembalikan `doInBackground()`. Parameter ini secara otomatis diteruskan ke `onPostExecute()` di thread utama.

Tetapkan tipe data untuk setiap tipe parameter ini, atau gunakan `Void` jika tipe parameter tidak akan digunakan. Misalnya:

```
public class MyAsyncTask extends AsyncTask <String, Void, Bitmap>{}
```

Dalam deklarasi kelas ini:

- Tipe parameter "Params" adalah `String`, yang berarti bahwa `MyAsyncTask` memerlukan satu atau beberapa string sebagai parameter di `doInBackground()`, misalnya untuk digunakan di kueri.
- Tipe parameter "Progress" adalah `Void`, yang berarti bahwa `MyAsyncTask` tidak akan menggunakan metode

`publishProgress()` atau `onProgressUpdate()` .

- Tipe parameter "Result" adalah `Bitmap` . `MyAsyncTask` mengembalikan Bitmap di `doInbackground()` , yang diteruskan ke dalam `onPostExecute()` .

Contoh AsyncTask

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
     protected Long doInBackground(URL... urls) {
         int count = urls.length;
         long totalSize = 0;
         for (int i = 0; i < count; i++) {
             totalSize += Downloader.downloadFile(urls[i]);
             publishProgress((int) ((i / (float) count) * 100));
             // Escape early if cancel() is called
             if (isCancelled()) break;
         }
         return totalSize;
     }

     protected void onProgressUpdate(Integer... progress) {
         setProgressPercent(progress[0]);
     }

     protected void onPostExecute(Long result) {
         showDialog("Downloaded " + result + " bytes");
     }
 }
```

Contoh di atas melewati tiga dari empat langkah-langkah AsyncTask dasar:

- `doInBackground()` mengunduh materi, tugas yang berjalan lama. Langkah ini menghitung persentase file yang diunduh dari indeks loop `for` dan meneruskannya ke `publishProgress()` . Pemeriksaan untuk `isCancelled()` di dalam loop `for` memastikan bahwa tugas telah dibatalkan, sistem tidak menunggu hingga loop selesai.
- `onProgressUpdate()` memperbarui kemajuan persentase. Ini dipanggil setiap kali metode `publishProgress()` dipanggil di dalam `doInBackground()` , yang memperbarui kemajuan persentase.
- `doInBackground()` menghitung jumlah total byte yang diunduh dan mengembalikannya. `onPostExecute()` menerima hasil yang dikembalikan dan meneruskannya ke dalam `onPostExecute()` , yang ditampilkan di dialog.

Tipe parameter yang digunakan dalam contoh ini adalah:

- `URL` untuk tipe parameter "Params". Tipe `URL` berarti Anda bisa meneruskan sejumlah URL ke dalam panggilan, dan URL secara otomatis diteruskan ke dalam metode `doInBackground()` sebagai larik.
- `Integer` untuk tipe parameter "Progress".
- `Long` untuk tipe parameter "Result".

Mengeksekusi AsyncTask

Setelah Anda mendefinisikan subkelas `AsyncTask` , buat instance-nya di thread UI. Kemudian panggil `execute()` di instance, dengan meneruskan sejumlah parameter. (Parameter tersebut sesuai dengan tipe parameter "Params" yang dibahas di atas).

Misalnya, untuk mengeksekusi tugas DownloadFilesTask yang didefinisikan di atas, gunakan baris kode berikut:

```
new DownloadFilesTask().execute(url1, url2, url3);
```

Membatalkan AsyncTask

Anda bisa membatalkan tugas kapan saja, dari thread apa pun, dengan memanggil metode `cancel()` .

- Metode `cancel()` akan mengembalikan `false` jika tugas tidak bisa dibatalkan, biasanya karena sudah diselesaikan secara normal. Jika tidak, `cancel()` akan mengembalikan `true` .
- Untuk mengetahui apakah tugas sudah dibatalkan, periksa nilai yang dikembalikan `isCancelled()` secara berkala dari `doInBackground(Object[])` , misalnya dari dalam loop seperti yang ditampilkan dalam contoh di atas. Metode `isCancelled()` akan mengembalikan `true` jika tugas dibatalkan sebelum diselesaikan secara normal.
- Setelah tugas `AsyncTask` dibatalkan, `onPostExecute()` tidak akan digunakan setelah `doInBackground()` dikembalikan. Sebagai gantinya, `onCancelled(Object)` akan dipanggil. Implementasi default `onCancelled(Object)` cukup memanggil `onCancelled()` dan mengabaikan hasil.
- Secara default, tugas yang sedang diproses boleh diselesaikan. Untuk memperbolehkan `cancel()` menyela thread yang sedang mengeksekusi tugas, teruskan `true` untuk nilai `mayInterruptIfRunning` .

Keterbatasan AsyncTask

`AsyncTask` tidak praktis untuk beberapa kasus penggunaan:

- Perubahan pada konfigurasi perangkat menyebabkan masalah.

  Bila konfigurasi perangkat berubah sewaktu `AsyncTask` berjalan, misalnya jika pengguna mengubah orientasi layar, aktivitas yang membuat `AsyncTask` akan dimusnahkan dan dibuat ulang. Metode `AsyncTask` tidak dapat mengakses aktivitas yang baru saja dibuat dan hasil `AsyncTask` tidak akan dipublikasikan.

- Objek `AsyncTask` lama tetap ada, dan aplikasi Anda bisa kehabisan memori atau mogok.

  Jika aktivitas yang membuat `AsyncTask` dimusnahkan, `AsyncTask` tidak akan dimusnahkan bersamanya. Misalnya, jika pengguna keluar dari aplikasi setelah `AsyncTask` dimulai, `AsyncTask` akan terus menggunakan sumber daya kecuali jika Anda memanggil `cancel()` .

Bila menggunakan `AsyncTask` :

- Tugas singkat atau yang bisa disela.
- Tugas yang tidak perlu untuk melaporkan kembali ke UI atau pengguna.
- Tugas dengan prioritas rendah yang bisa ditinggalkan sebelum selesai.

Untuk semua situasi lainnya, gunakan `AsyncTaskLoader` , adalah bagian dari kerangka kerja `Loader` yang akan dijelaskan berikutnya.

Loader

Tugas latar belakang biasanya digunakan untuk memuat data seperti laporan prakiraan cuaca atau ulasan film. Pemuatan data bisa jadi banyak menggunakan memori, dan Anda ingin data tersedia sekalipun jika konfigurasi perangkat berubah. Untuk situasi ini, gunakan *loader*, yang berupa rangkaian kelas yang memfasilitasi pemuatan data ke dalam aktivitas.

Loader menggunakan kelas `LoaderManager` untuk mengelola satu atau beberapa loader. `LoaderManager` menyertakan serangkaian callback bila loader telah dibuat, bila pemuatan datanya selesai, dan bila disetel ulang.

Memulai loader

Gunakan kelas `LoaderManager` untuk mengelola satu atau beberapa instance `Loader` dalam aktivitas atau fragmen. Gunakan `initLoader()` untuk melakukan inisialisasi dan mengaktifkannya. Biasanya, Anda melakukan ini dalam metode `onCreate()` aktivitas. Misalnya:

```
// Prepare the loader.  Either reconnect with an existing one,
// or start a new one.
getLoaderManager().initLoader(0, null, this);
```

Jika Anda menggunakan Pustaka Dukungan, buat panggilan ini menggunakan `getSupportLoaderManager()` sebagai ganti `getLoaderManager()`. Misalnya:

```
getSupportLoaderManager().initLoader(0, null, this);
```

Metode `initLoader()` memerlukan tiga parameter:

- ID unik yang mengidentifikasi loader. ID ini bisa berupa apa saja yang Anda inginkan.
- Argumen opsional yang disediakan ke loader saat pembuatan, dalam bentuk `Bundle`. Jika loader sudah ada, parameter ini akan diabaikan.
- Implementasi `LoaderCallbacks`, yang dipanggil oleh `LoaderManager` untuk melaporkan kejadian loader. Dalam contoh ini, kelas lokal mengimplementasikan antarmuka `LoaderManager.LoaderCallbacks`, sehingga meneruskan referensi ke dirinya sendiri, `this`.

Panggilan `initLoader()` memiliki dua kemungkinan hasil:

- Jika loader yang ditetapkan melalui ID sudah ada, maka loader yang dibuat terakhir menggunakan ID itu akan digunakan kembali.
- Jika loader yang ditetapkan melalui ID tidak ada, `initLoader()` akan memicu metode `onCreateLoader).` Di sinilah Anda mengimplementasikan kode untuk membuat instance dan mengembalikan loader baru.

  Catatan: Bila `initLoader()` membuat loader atau menggunakan kembali loader yang ada, implementasi `LoaderCallbacks` yang diberikan akan dikaitkan dengan loader dan dipanggil bila keadaan loader berubah. Jika loader yang diminta sudah ada dan sudah menghasilkan data, maka sistem segera memanggil `onLoadFinished()` (selama `initLoader()`), jadi bersiaplah jika hal ini terjadi.

  Masukkan panggilan ke `initLoader()` di `onCreate()` sehingga aktivitas bisa dihubungkan kembali ke loader *yang sama* bila konfigurasi berubah. Dengan cara itu, loader tidak kehilangan data yang sudah dimuatnya.

Memulai ulang loader

Bila `initLoader()` menggunakan kembali loader yang ada, maka data yang telah dimuat loader tidak akan diganti, namun kadang-kadang Anda perlu menggantinya. Misalnya, bila Anda menggunakan kueri pengguna untuk melakukan penelusuran dan pengguna memasukkan kueri baru, Anda perlu memuat ulang data dengan menggunakan istilah penelusuran baru. Dalam situasi ini, gunakan metode `restartLoader()` dan teruskan ID loader yang ingin dimulai ulang. Hal ini akan memaksa muatan data lain dengan data masukan baru.

Tentang metode `restartLoader()`:

- `restartLoader()` menggunakan argumen yang sama dengan `initLoader()`.
- `restartLoader()` akan memicu metode `onCreateLoader()`, seperti yang dilakukan `initLoader()` saat membuat loader baru.
- Jika sudah ada loader dengan ID yang diberikan, `restartLoader()` akan memulai ulang loader yang diidentifikasi dan mengganti datanya.
- Jika tidak ada loader dengan ID yang diberikan, `restartLoader()` akan memulai loader baru.

Callback LoaderManager

Objek `LoaderManager` secara otomatis memanggil `onStartLoading()` saat membuat loader. Setelah itu, `LoaderManager` akan mengelola keadaan loader berdasarkan pada keadaan aktivitas dan data, misalnya dengan memanggil `onLoadFinished()` bila data telah dimuat.

Untuk berinteraksi dengan loader, gunakan salah satu `callback LoaderManager` di aktivitas yang memerlukan data:

- Panggil `onCreateLoader()` agar bisa membuat instance dan mengembalikan loader baru untuk ID yang diberikan.
- Panggil `onLoadFinished()` bila loader yang dibuat sebelumnya selesai memuat. Di sinilah Anda biasanya ingin memindahkan data ke dalam tampilan aktivitas.

- Panggil `onLoaderReset()` bila loader yang dibuat sebelumnya sedang disetel ulang, sehingga datanya tidak tersedia. Di sinilah aplikasi harus membuang semua referensi apa pun yang dimilikinya ke data loader.

Subkelas `Loader` bertanggung jawab atas pemuatan data sebenarnya. Subkelas `Loader` yang Anda gunakan bergantung pada tipe data yang dimuat, namun salah satu yang paling mudah adalah `AsyncTaskLoader`, yang akan dijelaskan berikutnya. `AsyncTaskLoader` menggunakan `AsyncTask` untuk menjalankan tugas pada thread pekerja.

AsyncTaskLoader

`AsyncTaskLoader` adalah loader yang setara dengan `AsyncTask`. `AsyncTaskLoader` menyediakan metode, `loadInBackground()`, yang dijalankan di thread terpisah. Hasil `loadInBackground()` secara otomatis dikirimkan ke thread UI, melalui `onLoadFinished()` `LoaderManager` callback.

Penggunaan AsyncTaskLoader

Untuk mendefinisikan subkelas `AsyncTaskLoader`, buat kelas yang memperluas `AsyncTaskLoader<D>`, dalam hal ini `D` adalah tipe data yang sedang Anda muat. Misalnya, `AsyncTaskLoader` ini akan memuat daftar string:

```
public static class StringListLoader extends AsyncTaskLoader<List<String>> {}
```

Berikutnya, implementasikan konstruktor yang cocok dengan implementasi super kelas:

- Konstruktor menggunakan konteks aplikasi sebagai argumen dan meneruskannya ke panggilan untuk `super()`.
- Jika loader Anda memerlukan informasi tambahan untuk melakukan pemuatan, konstruktor bisa mengambil argumen tambahan. Dalam contoh yang ditampilkan di bawah ini, konstruktor menggunakan sebuah istilah kueri.

```
public StringListLoader(Context context, String queryString) {
   super(context);
   mQueryString = queryString;
}
```

Untuk melakukan pemuatan, gunakan metode penggantian `loadInBackground()`, akibat metode `doInBackground() dari AsyncTask`. Misalnya:

```
@Override
public List<String> loadInBackground() {
    List<String> data = new ArrayList<String>;
    //TODO: Load the data from the network or from a database
    return data;
}
```

Mengimplementasikan callback

Gunakan konstruktor di callback `onCreateLoader()` `LoaderManager`, yang merupakan tempat membuat loader baru. Misalnya, callback `onCreateLoader()` ini menggunakan konstruktor `StringListLoader` yang didefinisikan di atas:

```
@Override
public Loader<List<String>> onCreateLoader(int id, Bundle args) {
   return new StringListLoader(this, args.getString("queryString"));
}
```

Hasil `loadInBackground()` secara otomatis diteruskan ke dalam callback `onLoadFinished()`, di sinilah Anda bisa menampilkan hasil di UI. Misalnya:

```
public void onLoadFinished(Loader<List<String>> loader, List<String> data) {
    mAdapter.setData(data);
}
```

Callback `onLoaderReset()` hanya dipanggil bila loader akan dimusnahkan, sehingga seringkali Anda bisa mengosongkan `onLoaderReset()` , karena Anda tidak akan mencoba mengakses data setelah loader ini dimusnahkan.

Bila Anda menggunakan `AsyncTaskLoader` , data Anda akan bertahan bila ada perubahan konfigurasi perangkat. Jika aktivitas Anda dmusnahkan secara permanen, loader ini akan dimusnahkan bersamanya, tanpa tugas yang menanti dan mengonsumsi sumber daya sistem.

Loader juga memiliki manfaat lain, misalnya loader bisa memantau perubahan sumber data dan memuat ulang data jika terjadi perubahan. Anda akan mengetahui selengkapnya tentang loader tertentu di pelajaran berikutnya.

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Buat AsyncTask

Ketahui selengkapnya

- Referensi AsyncTask
- Referensi AsyncTaskLoader
- Referensi LoaderManager
- Proses dan Thread
- Panduan loader

7.2: Hubungkan ke Internet

Materi:

Sebagian besar aplikasi Android memiliki beberapa data yang berinteraksi dengan pengguna; seperti artikel berita, informasi cuaca, kontak, data game, informasi pengguna, dan lainnya. Seringkali, data ini disediakan melalui jaringan oleh API web.

Dalam pelajaran ini, Anda akan mempelajari tentang keamanan jaringan dan cara membuat panggilan jaringan, yang melibatkan tugas-tugas ini:

1. Sertakan izin dalam file AndroidManifest.xml.
2. Pada thread pekerja, buat koneksi klien HTTP yang menghubungkan ke jaringan dan mengunduh (atau mengunggah) data.
3. Parse hasil, yang biasanya dalam format JSON.
4. Periksa keadaan jaringan dan tanggapi sebagaimana semestinya.

Keamanan jaringan

Transaksi jaringan berisiko inheren, karena mereka melibatkan transmisi data yang boleh jadi bersifat privat untuk pengguna. Orang semakin sadar dengan risiko ini, terutama ketika perangkat mereka melakukan transaksi jaringan, jadi sangat penting jika aplikasi Anda mengimplementasikan praktik terbaik untuk menjaga data pengguna tetap aman setiap saat.

Praktik terbaik keamanan untuk operasi jaringan:

- Gunakan protokol yang sesuai untuk data sensitif. Misalnya untuk lalu lintas web aman, gunakan subkelas `HttpsURLConnection` dari `HttpURLConnection`.
- Gunakan HTTPS sebagai ganti HTTP di mana saja HTTPS didukung pada server, karena perangkat seluler sering menghubungkan pada jaringan tidak aman seperti hotspot Wi-Fi publik. Pertimbangkan penggunaan `SSLSocketClass` untuk mengimplementasikan komunikasi level soket yang diautentikasi dan dienkripsi.
- Jangan gunakan porta jaringan host lokal untuk menangani komunikasi interproses sensitif (IPC), karena aplikasi lain di perangkat bisa mengakses porta lokal ini. Sebagai gantinya, gunakan mekanisme yang memungkinkan Anda menggunakan autentikasi, misalnya `Service`.
- Jangan mempercayai data yang telah diunduh dari HTTP atau protokol tidak aman lainnya. Validasi masukan yang dimasukkan ke dalam `WebView` dan respons ke maksud yang Anda keluarkan terhadap HTTP.

Untuk praktik terbaik dan tips keamanan, lihat artikel Tips Keamanan.

Menyertakan izin dalam manifes

Sebelum aplikasi bisa melakukan panggilan jaringan, Anda perlu menyertakan izin dalam file AndroidManifest.xml. Tambahkan tag berikut di dalam tag `<manifest>` :

```
<uses-permission android:name="android.permission.INTERNET" />
```

Saat menggunakan jaringan, praktik terbaiknya adalah memantau keadaan jaringan perangkat sehingga Anda tidak berupaya membuat panggilan jaringan ketika jaringan tidak tersedia. Untuk mengakses keadaan jaringan perangkat, aplikasi memerlukan izin tambahan:

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Menjalankan operasi jaringan di thread pekerja

Selalu jalankan operasi jaringan di thread pekerja, terpisah dari UI. Misalnya, di kode Java, Anda bisa membuat implementasi `AsyncTask` (atau `AsyncTaskLoader`) yang membuka koneksi jaringan dan mengkueri API. Kode utama Anda akan memeriksa apakah koneksi jaringan aktif. Jika aktif, maka akan menjalankan `AsyncTask` di thread terpisah, kemudian menampilkan hasil di UI.

Catatan: Jika menjalankan operasi jaringan di thread utama sebagai ganti di thread pekerja, Anda akan menerima kesalahan.

Membuat koneksi HTTP

Sebagian besar aplikasi Android yang terhubung ke jaringan menggunakan HTTP dan HTTPS untuk mengirim dan menerima data melalui jaringan. Untuk penyegaran di HTTP, kunjungi Pelajari tutorial HTTP.

Catatan: Jika server web menawarkan HTTPS, Anda harus menggunakannya sebagai ganti HTTP untuk peningkatan keamanan.
Klien Android `HttpURLConnection` mendukung HTTPS, unggahan dan unduhan streaming, waktu tunggu yang dapat dikonfigurasi, IPv6, dan penjajakan koneksi. Untuk menggunakan klien `HttpURLConnection` , bangun URI (tujuan permintaan). Selanjutnya dapatkan koneksi, kirim permintaan dan header permintaan apa pun, unduh dan baca respons dan header respons apa pun, dan putuskan koneksi.

Membangun URI Anda

Untuk membuka koneksi HTTP, Anda perlu membangun URI permintaan. URI biasanya dibuat dari URL dasar dan sekumpulan parameter kueri yang menetapkan sumber daya yang dimaksud. Misalnya untuk menelusuri lima hasil buku pertama untuk "Pride and Prejudice" di Google Books API, gunakan URI berikut:

```
https://www.googleapis.com/books/v1/volumes?q=pride+prejudice&maxResults=5&printType=books
```

Untuk menyusun URI permintaan lewat program, gunakan metode `URI.parse()` dengan metode `buildUpon()` dan `appendQueryParameter()` . Kode berikut membangun URI lengkap yang ditampilkan di atas:

```
// Base URL for the Books API.
final String BOOK_BASE_URL =  "https://www.googleapis.com/books/v1/volumes?";

final String QUERY_PARAM = "q"; // Parameter for the search string
final String MAX_RESULTS = "maxResults"; // Parameter to limit search results.
final String PRINT_TYPE = "printType"; // Parameter to filter by print type

// Build up the query URI, limiting results to 5 items and printed books.
Uri builtURI = Uri.parse(BOOK_BASE_URL).buildUpon()
       .appendQueryParameter(QUERY_PARAM, "pride+prejudice")
       .appendQueryParameter(MAX_RESULTS, "5")
       .appendQueryParameter(PRINT_TYPE, "books")
       .build();
```

Untuk mengubah URI menjadi string, gunakan metode `toString()` :

```
String myurl = builtURI.toString();
```

Hubungkan dan unduh data

Di thread pekerja yang menjalankan transaksi jaringan Anda, misalnya dalam penggantian metode `doInBackground()` di `AsyncTask` , gunakan kelas `HttpURLConnection` untuk menjalankan permintaan `GET` HTTP dan mengunduh data yang diperlukan aplikasi. Begini caranya:

1. Untuk memperoleh `HttpURLConnection` baru, panggil `URL.openConnection()` menggunakan URI yang Anda bangun. Transmisikan hasilnya ke `HttpURLConnection` .

   URI adalah properti utama permintaan, namun header permintaan juga bisa menyertakan metadata seperti kredensial, tipe materi pilihan, dan cookie sesi.

2. Setel parameter opsional:
   - Untuk koneksi lambat, Anda mungkin menginginkan waktu tunggu koneksi yang lama (waktu untuk membuat koneksi awal ke sumber daya) atau waktu tunggu baca (waktu untuk benar-benar membaca data).
   - Untuk mengubah metode permintaan ke selain `GET` , gunakan `setRequestMethod()` .
   - Jika Anda tidak ingin menggunakan jaringan untuk masukan, setel `setDoInput` ke `false` . (Defaultnya adalah `true` .)
   - Untuk metode lain yang bisa Anda setel, lihat dokumentasi referensi `HttpURLConnection` dan `URLConnection.`
3. Buka aliran masukan menggunakan `getInputStream()` , kemudian baca respons dan ubah menjadi string. Header respons biasanya menyertakan metadata seperti tipe materi dan panjang isi respons, tanggal modifikasi, dan cookie sesi. Jika respons tidak memiliki isi, `getInputStream()` akan mengembalikan aliran kosong.
4. Panggil `disconnect()` untuk menutup koneksi. Memutus koneksi akan melepas sumber daya yang ditahan oleh koneksi sehingga sumber daya bisa ditutup atau digunakan kembali.

Langkah-langkah ini ditampilkan dalam Contoh permintaan, di bawah.

Jika mengeposkan data melalui jaringan dan tidak hanya menerima data, Anda perlu mengunggah *isi permintaan*, yang menampung data yang akan diposkan. Caranya:

1. Konfigurasikan koneksi sehingga keluaran dimungkinkan dengan memanggil `setDoOutput(true)` . (Secara default, `HttpURLConnection` menggunakan permintaan `GET` HTTP. Bila `setDoOutput` adalah `true` , `HttpURLConnection` menggunakan permintaan `POST` HTTP secara default.)
2. Buka aliran keluaran dengan memanggil `getOutputStream()` .

Untuk informasi selengkapnya tentang mengeposkan data ke jaringan, lihat "Mengeposkan Materi" dalam `dokumentasi HttpURLConnection` .

Catatan: Semua panggilan jaringan harus dijalankan di thread pekerja dan bukan di thread UI.

Contoh permintaan

Contoh berikut mengirimkan permintaan ke URL yang dibangun di bagian Membangun URI Anda, di atas. Permintaan memperoleh `HttpURLConnection` baru, membuka aliran masukan, membaca respons, mengubah respons menjadi string, dan menutup koneksi.

```
private String downloadUrl(String myurl) throws IOException {
    InputStream inputStream = null;
    // Only display the first 500 characters of the retrieved
    // web page content.
    int len = 500;

    try {
        URL url = new URL(myurl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(10000 /* milliseconds */);
        conn.setConnectTimeout(15000 /* milliseconds */);
        // Start the query
        conn.connect();
        int response = conn.getResponseCode();
        Log.d(DEBUG_TAG, "The response is: " + response);
        inputStream = conn.getInputStream();

        // Convert the InputStream into a string
        String contentAsString = convertInputToString(inputStream, len);
        return contentAsString;

    // Close the InputStream and connection
    } finally {
      conn.disconnect();
        if (inputStream != null) {
            inputStream.close();
        }
    }
}
```

Mengubah InputStream menjadi string

`InputStream` merupakan sumber byte yang dapat dibaca. Setelah Anda mendapatkan `InputStream`, biasanya kemudian mendekode atau mengubahnya menjadi tipe data yang diperlukan. Dalam contoh di atas, `InputStream` menyatakan teks biasa dari laman web yang berada di https://www.googleapis.com/books/v1/volumes?q=pride+prejudice&maxResults=5&printType=books.

Metode `convertInputToString` yang didefinisikan di bawah ini mengubah `InputStream` menjadi string sehingga aktivitas bisa menampilkanya di UI. Metode ini menggunakan instance `InputStreamReader` untuk membaca byte dan mendekode byte menjadi karakter:

```
// Reads an InputStream and converts it to a String.
public String convertInputToString(InputStream stream, int len)
           throws IOException, UnsupportedEncodingException {
    Reader reader = null;
    reader = new InputStreamReader(stream, "UTF-8");
    char[] buffer = new char[len];
    reader.read(buffer);
    return new String(buffer);
}
```

Catatan: Jika Anda mengharapkan respons yang panjang, bungkus `InputStreamReader` di dalam `BufferedReader` agar pembacaan karakter, larik, dan baris menjadi lebih efisien. Misalnya:
`reader = new BufferedReader(new InputStreamReader(stream, "UTF-8"));`

Mem-parse hasil

Bila Anda membuat kueri API web, hasilnya seringkali dalam format JSON.

Di bawah ini adalah contoh respons JSON dari permintaan HTTP. Respons ini menampilkan nama tiga item menu di menu munculan dan metode yang dipicu bila item menu diklik:

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}
```

Untuk menemukan nilai item dalam respons, gunakan metode dari kelas `JSONObject` dan `JSONArray`. Misalnya, inilah cara untuk menemukan nilai `"onclick"` dari item ketiga di larik `"menuitem"` :

```
JSONObject data = new JSONObject(responseString);
JSONArray menuItemArray = data.getJSONArray("menuitem");
JSONObject thirdItem = menuItemArray.getJSONObject(2);
String onClick = thirdItem.getString("onclick");
```

Mengelola keadaan jaringan

Membuat panggilan jaringan bisa mahal dan lambat, terutama jika perangkat memiliki konektivitas yang kecil. Mewaspadai keadaan koneksi jaringan bisa mencegah aplikasi Anda dari berupaya membuat panggilan jaringan saat jaringan tidak tersedia.

Kadang-kadang juga penting bagi aplikasi untuk mengetahui jenis konektivitas yang dimiliki perangkat: Jaringan Wi-Fi biasanya lebih cepat daripada jaringan data, dan jaringan data biasanya seringkali berkuota dan mahal. Untuk mengontrol kapan tugas tertentu dijalankan, pantau keadaan jaringan dan tanggapi sebagaimana mestinya. Misalnya, Anda mungkin ingin menunggu hingga perangkat terhubung ke Wi-Fi untuk melakukan pengunduhan file besar.

Untuk memeriksa koneksi jaringan, gunakan kelas berikut:

- `ConnectivityManager` akan menjawab kueri tentang keadaan konektivitas jaringan. Juga memberi tahu aplikasi bila konektivitas jaringan berubah.
- `NetworkInfo` akan menjelaskan status antarmuka jaringan dari tipe yang diberikan (saat ini seluler atau Wi-Fi).

Cuplikan kode berikut menguji apakah Wi-Fi dan seluler terhubung. Dalam kode:

- Metode `getSystemService` mendapatkan instance `ConnectivityManager` .
- Metode `getNetworkInfo` mendapatkan status koneksi Wi-Fi perangkat, kemudian koneksi selulernya. Metode `getNetworkInfo` mengembalikan objek `NetworkInfo` , yang berisi informasi tentang status koneksi jaringan yang diberikan (apakah menganggur, terhubung, dan seterusnya).
- Metode `networkInfo.isConnected()` akan mengembalikan `true` jika jaringan yang diberikan terhubung. Jika jaringan terhubung, maka bisa digunakan untuk membangun soket dan meneruskan data.

```
private static final String DEBUG_TAG = "NetworkStatusExample";
ConnectivityManager connMgr = (ConnectivityManager)
    getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
boolean isWifiConn = networkInfo.isConnected();
networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
boolean isMobileConn = networkInfo.isConnected();
Log.d(DEBUG_TAG, "Wifi connected: " + isWifiConn);
Log.d(DEBUG_TAG, "Mobile connected: " + isMobileConn);
```

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Menghubungkan ke Internet dengan AsyncTask dan AsyncTaskLoader

Ketahui selengkapnya

- Menghubungkan ke Jaringan
- Mengelola Penggunaan Jaringan
- Referensi HttpURLConnection
- Referensi ConnectivityManager
- Referensi InputStream

7.3: Penerima Siaran

Materi:

- Pengantar
- Maksud siaran
- Penerima siaran
- Panduan keamanan
- LocalBroadcastManager
- Praktik terkait
- Ketahui selengkapnya

Maksud eksplisit digunakan untuk memulai aktivitas tertentu yang benar-benar memenuhi syarat, serta untuk meneruskan informasi di antara aktivitas di aplikasi Anda. *Maksud implisit* digunakan untuk memulai aktivitas berdasarkan komponen terdaftar yang diketahui sistem, misalnya fungsionalitas umum.

Dalam pelajaran ini Anda akan mempelajari tentang *maksud siaran*, yang tidak memulai aktivitas namun dikirimkan ke *penerima siaran*.

Maksud siaran

Maksud yang telah Anda lihat sejauh ini selalu menghasilkan aktivitas yang akan diluncurkan, baik aktivitas tertentu dari aplikasi Anda maupun aktivitas dari aplikasi lain yang bisa memenuhi aksi yang diminta. Namun kadang-kadang maksud tidak memiliki penerima tertentu, dan kadang-kadang Anda tidak ingin aktivitas diluncurkan sebagai respons terhadap maksud. Misalnya, bila aplikasi Anda menerima maksud sistem yang menunjukkan bahwa keadaan jaringan di perangkat sudah berubah, Anda mungkin tidak ingin meluncurkan aktivitas, melainkan mungkin ingin menonaktifkan beberapa fungsionalitas aplikasi.

Karena alasan ini, ada tipe maksud ketiga yang bisa dikirimkan ke aplikasi yang tertarik: *maksud siaran*. Meskipun maksud siaran didefinisikan dengan cara yang sama seperti maksud implisit, setiap tipe maksud memiliki karakteristik pembeda yang penting:

- Maksud siaran dikirim menggunakan `sendBroadcast()` atau metode terkait, sementara tipe maksud lainnya menggunakan `startActivity()` untuk memulai aktivitas. Bila menyiarkan maksud, Anda tidak akan pernah menemukan atau memulai aktivitas. Demikian juga, tidak ada cara bagi penerima siaran untuk melihat atau menangkap maksud yang digunakan bersama `startActivity()` .
- Maksud siaran adalah operasi latar belakang yang biasanya tidak diketahui pengguna. Memulai aktivitas bersama maksud, di sisi lain, adalah operasi latar depan yang memodifikasi apa yang sedang berinteraksi dengan pengguna.

Ada dua tipe maksud siaran, yang dikirimkan oleh sistem (maksud siaran sistem), dan yang dikirimkan oleh aplikasi Anda (maksud siaran khusus).

Maksud siaran sistem

Sistem mengirimkan *maksud siaran sistem* bila ada kejadian sistem yang mungkin menarik aplikasi. Misalnya:

- Bila perangkat sedang booting, sistem akan mengirim sebuah maksud siaran sistem `ACTION_BOOT_COMPLETED` . Maksud ini berisi nilai konstanta `"android.intent.action.BOOT_COMPLETED"` .
- Bila perangkat terhubung ke daya eksternal, sistem akan mengirim `ACTION_POWER_CONNECTED` , yang berisi nilai konstanta `"android.intent.action.ACTION_POWER_CONNECTED"` . Bila perangkat terlepas dari daya eksternal, sistem akan mengirim `ACTION_POWER_DISCONNECTED` .
- Bila memori perangkat minim, sistem akan mengirim `ACTION_DEVICE_STORAGE_LOW` . Maksud ini berisi nilai konstanta `"android.intent.action.DEVICE_STORAGE_LOW"` .

`ACTION_DEVICE_STORAGE_LOW` merupakan siaran *melekat*, maksudnya adalah nilai siaran bertahan di cache. Jika Anda perlu mengetahui apakah penerima siaran memproses nilai yang ada di cache (melekat) atau nilai yang disiarkan pada saat ini, gunakan `isInitialStickyBroadcast()`.

Untuk informasi selengkapnya tentang siaran sistem umum, kunjungi referensi `Intent`.

Untuk menerima maksud siaran sistem, Anda perlu membuat penerima siaran.

Maksud siaran khusus

Maksud siaran khusus adalah maksud siaran yang dikirim aplikasi Anda. Gunakan maksud siaran khusus bila Anda ingin agar aplikasi mengambil suatu aksi tanpa meluncurkan aktivitas, misalnya bila Anda ingin membiarkan aplikasi lainnya mengetahui bahwa data telah diunduh ke perangkat dan tersedia untuk digunakan.

Untuk membuat maksud siaran khusus, buat aksi maksud khusus. Untuk mengirim siaran khusus ke aplikasi lainnya, teruskan maksud ke `sendBroadcast()`, `sendOrderedBroadcast()`, atau `sendStickyBroadcast()`. (Untuk detail tentang metode ini, lihat dokumentasi referensi `Context`.)

Misalnya, metode berikut membuat sebuah maksud dan menyiarkannya ke semua [penerima siaran] yang tertarik(#broadcast_receivers):

```
public void sendBroadcastIntent() {
   Intent intent = new Intent();
   intent.setAction("com.example.myproject.ACTION_SHOW_TOAST");
   sendBroadcast(intent);
}
```

Catatan: Bila Anda menetapkan aksi untuk maksud, gunakan nama paket unik (`com.example.myproject` dalam contoh) untuk memastikan maksud tersebut tidak konflik dengan maksud yang disiarkan dari aplikasi lain atau dari sistem.

Penerima siaran

Maksud siaran tidak ditargetkan pada penerima tertentu. Sebagai gantinya, aplikasi yang tertarik mendaftarkan komponen untuk "mendengarkan" jenis maksud ini. Komponen pendengar ini disebut *penerima siaran*.

Gunakan penerima siaran untuk merespons pesan yang disiarkan dari aplikasi lain atau dari sistem. Untuk membuat penerima siaran:

1. Definisikan sub kelas dari kelas `BroadcastReceiver` dan implementasikan metode `onReceive()`.
2. Daftarkan penerima siaran secara dinamis di Java, atau secara statis di file manifes aplikasi Anda.

   Sebagai bagian dari langkah ini, gunakan *filter maksud* untuk menetapkan jenis maksud siaran yang ingin Anda terima.

Langkah-langkah ini dijelaskan di bawah.

Definisikan subkelas BroadcastReceiver

Untuk membuat penerima siaran, definisikan subkelas dari kelas `BroadcastReceiver`. Di subkelas inilah maksud dikirim (jika cocok dengan filter maksud yang Anda setel saat mendaftarkan subkelas, yang terjadi di langkah berikutnya).

Dalam definisi subkelas Anda:

- Implementasikan metode `onReceive()` yang diperlukan.
- Sertakan logika apa pun yang diperlukan penerima siaran Anda.

Contoh: Buat penerima siaran

Dalam contoh ini, subkelas `AlarmReceiver` dari `BroadcastReceiver` menampilkan pesan Toast jika maksud siaran yang masuk memiliki aksi `ACTION_SHOW_TOAST`:

```
private class AlarmReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
       if (intent.getAction().equals(ACTION_SHOW_TOAST)) {
            CharSequence text = "Broadcast Received!";
            int duration = Toast.LENGTH_SHORT;

            Toast toast = Toast.makeText(context, text, duration);
            toast.show();
       }
   }
}
```

Catatan: Jangan menggunakan operasi asinkron dalam implementasi `onReceive()`, karena setelah kode Anda kembali dari `onReceive()`, sistem akan menganggap objek `BroadcastReceiver` akan diselesaikan. Jika `onReceive()` memulai operasi asinkron, sistem akan mematikan proses `BroadcastReceiver` sebelum operasi asinkron memiliki kesempatan untuk diselesaikan.
Jika Anda memerlukan operasi yang berjalan lama yang tidak memerlukan UI, gunakan Service yang diluncurkan dari penerima siaran. Khususnya:

- Anda tidak bisa menampilkan dialog dari dalam `BroadcastReceiver`. Sebagai gantinya, gunakan `NotificationManager` API.
- Anda tidak bisa mengikat ke layanan dari dalam `BroadcastReceiver`. Sebagai gantinya, gunakan `Context.startService()` untuk mengirim perintah ke layanan.

Mendaftarkan penerima siaran Anda dan menyetel filter maksud

Ada dua cara untuk mendaftarkan penerima siaran: secara statis di manifes, atau secara dinamis di aktivitas Anda.

Pendaftaran statis

Untuk mendaftarkan penerima siaran secara statis, tambahkan elemen `<receiver>` ke file AndroidManifest.xml Anda. Dalam elemen `<receiver>`:

- Gunakan jalur ke subkelas `BroadcastReceiver` sebagai atribut `android:name`.
- Untuk mencegah aplikasi lain mengirim siaran ke penerima Anda, setel atribut `android:exported` opsional ke `false`. Ini adalah [panduan keamanan] penting(#security).
- Untuk menetapkan tipe maksud yang akan didengarkan komponen, gunakan elemen `<intent-filter>` yang disarangkan.

Contoh: Pendaftaran statis dan filter maksud untuk maksud siaran khusus

Cuplikan kode berikut adalah contoh pendaftaran statis untuk penerima siaran yang mendengarkan maksud siaran khusus bersama " `ACTION_SHOW_TOAST` " dalam nama aksinya:

- `name` penerima adalah nama modul ditambah nama subkelas `BroadcastReceiver` yang didefinisikan di atas (`AlarmReceiver`).
- Penerima tidak diekspor, berarti tidak ada aplikasi lain yang bisa mengirim siaran ke aplikasi ini.
- Penerima menyertakan filter maksud yang memeriksa apakah maksud yang masuk menyertakan aksi bernama `ACTION_SHOW_TOAST`.

```
<receiver
android:name="com.example.myproject.AlarmReceiver"
android:exported="false">
<intent-filter>
   <action android:name="com.example.myproject.intent.action.ACTION_SHOW_TOAST"/>
</intent-filter>
</receiver>
```

Filter maksud

Bila sistem menerima maksud implisit untuk memulai aktivitas, sistem akan menelusuri aktivitas terbaik bagi maksud tersebut dengan membandingkannya dengan filter maksud, berdasarkan tiga aspek:

- Aksi: Apakah aksi yang ditetapkan dalam maksud cocok dengan salah satu nama `<action>` yang tercantum dalam filter? Dalam contoh di atas, hanya maksud yang berisi `ACTION_SHOW_TOAST` dalam nama aksinya yang akan cocok dengan filter tersebut.
- Data: Apakah data dalam maksud cocok dengan salah satu tipe `<data>` yang tercantum dalam filter?
- Kategori: Apakah setiap kategori dalam maksud cocok dengan `<category>` yang namanya disebutkan dalam filter?

Contoh: Filter maksud untuk maksud siaran sistem

Contoh ini menampilkan filter maksud bagi penerima yang mendengarkan perangkat untuk menyelesaikan booting:

```
<intent-filter>
    <action android:name="android.intent.action.BOOT_COMPLETED"/>
</intent-filter>
```

Untuk mengetahui selengkapnya tentang menggunakan filter maksud guna memilih maksud, lihat bagian Resolusi Maksud pada panduan maksud.

Jika tidak ada filter maksud yang ditetapkan, penerima siaran hanya bisa diaktifkan dengan maksud siaran eksplisit yang memberi nama komponen berdasarkan nama. (Tindakan ini mirip dengan cara Anda meluncurkan aktivitas melalui nama kelasnya bersama maksud eksplisit.)

Jika Anda menggunakan pendaftaran statis untuk penerima siaran, sistem Android akan membuat proses baru untuk menjalankan penerima siaran jika tidak ada proses berjalan yang dikaitkan dengan aplikasi Anda. Ini berarti penerima akan merespons, sekalipun aplikasi Anda tidak berjalan.

Pendaftaran dan menghapus pendaftaran dinamis

Anda juga bisa mendaftarkan penerima siaran secara dinamis, yang mengikat operasinya ke daur hidup aktivitas. Untuk mendaftarkan penerima Anda secara dinamis, panggil `registerReceiver()` dan teruskan objek `BroadcastReceiver` dan filter maksud. Misalnya:

```
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(ACTION_SHOW_TOAST);

mReceiver = new AlarmReceiver();
registerReceiver(mReceiver, intentFilter);
```

Catatan: Jika mendaftarkan penerima untuk menerima siaran lokal saja, Anda harus mendaftarkannya secara dinamis; pendaftaran statis bukanlah opsi.
Anda juga perlu mencabut pendaftaran penerima dengan memanggil `unregisterReceiver()` dan meneruskan objek `BroadcastReceiver` . Misalnya:

```
unregisterReceiver(mReceiver);
```

Tempat Anda memanggil metode ini bergantung pada daur hidup yang diinginkan objek `BroadcastReceiver` :

- Jika penerima hanya diperlukan bila aktivitas terlihat (misalnya, untuk menonaktifkan fungsi jaringan bila jaringan tidak tersedia), maka daftarkan penerima di `onResume()` . Hapus pendaftaran penerima di `onPause()` .
- Anda juga bisa menggunakan pasangan metode `onStart()` / `onStop()` atau `onCreate()/onDestoy()` , jika pasangan tersebut lebih tepat untuk kasus penggunaan Anda.

Panduan keamanan

Bila Anda menggunakan maksud siaran dan penerima siaran, informasi akan dikirim di antara aplikasi, yang akan menimbulkan risiko keamanan. Untuk menghindari risiko ini, Anda bisa menggunakan `LocalBroadcastManager` (dijelaskan di bawah ini), atau Anda bisa mengikuti panduan ini:

- Pastikan nama tindakan maksud dan string lainnya ada dalam namespace yang Anda miliki, jika tidak maka Anda akan mengalami konflik dengan aplikasi lain secara tidak sengaja. Namespace maksud bersifat global.
- Bila Anda menggunakan `registerReceiver()` , aplikasi apa pun bisa mengirim siaran ke penerima yang didaftarkan itu. Untuk mengontrol siapa yang bisa mengirimkan siaran, gunakan izin yang dijelaskan di bawah ini.
- Bila Anda mendaftarkan penerima siaran secara statis, aplikasi lainnya bisa mengirim siaran ke penerima tersebut, filter apa pun yang Anda tetapkan. Untuk mencegah aplikasi lain mengirim ke penerima tersebut, jadikan aplikasi itu tidak tersedia dengan `android:exported="false"` .
- Bila Anda menggunakan `sendBroadcast()` atau metode terkait, aplikasi lain akan bisa menerima siaran Anda. Untuk mengontrol siapa saja yang bisa menerima siaran tersebut, gunakan izin yang dijelaskan di bawah ini.

Baik pengirim maupun penerima siaran bisa memberlakukan izin akses:

- Untuk memberlakukan izin saat **mengirim** siaran, berikan argumen izin bukan-nol ke `sendBroadcast()` .

  Hanya penerima yang telah diberi izin ini (dengan memintanya melalui tag `<uses-permission>` di AndroidManifest.xml) yang akan bisa menerima siaran.

- Untuk memberlakukan izin saat **menerima** siaran, berikan izin bukan-nol saat mendaftarkan penerima Anda, baik saat memanggil `registerReceiver()` maupun dalam tag statis `<receiver>` di AndroidManifest.xml.

  Hanya penyiar yang telah diberi izin ini (dengan memintanya melalui tag `<uses-permission>` di AndroidManifest.xml) yang akan bisa mengirimkan maksud ke penerima. Penerima harus meminta izin di manifes, terlepas apakah pengirim didaftarkan secara statis atau dinamis.

[LocalBroadcastManager]

Agar tidak perlu menangani aspek keamanan yang dijelaskan dalam Panduan keamanan, gunakan kelas `LocalBroadcastManager` . `LocalBroadcastManager` memungkinkan Anda mengirim dan menerima siaran dalam satu proses dan satu aplikasi, sehingga Anda tidak perlu khawatir tentang keamanan lintas aplikasi.

Mengirim siaran lokal

Untuk mengirim siaran menggunakan `LocalBroadcastManager` :

1. Dapatkan instance `LocalBroadcastManager` dengan memanggil `getInstance()` dan meneruskan konteks aplikasi.
2. Panggil `sendBroadcast()` di instance, dengan meneruskan maksud yang ingin disiarkan.

Misalnya:

```
LocalBroadcastManager.getInstance(this).sendBroadcast(customBroadcastIntent);
```

Mendaftarkan penerima Anda untuk siaran lokal

Untuk mendaftarkan penerima Anda guna menerima siaran lokal saja:

1. Dapatkan instance `LocalBroadcastManager` dengan memanggil `getInstance()` dan meneruskan konteks aplikasi.
2. Panggil `registerReceiver()`, dengan meneruskan penerima dan filter maksud seperti yang diinginkan untuk penerima siaran biasa. Anda harus mendaftarkan penerima lokal secara dinamis, karena pendaftaran statis di manifes tidak tersedia.

Misalnya:

```
LocalBroadcastManager.getInstance(this).registerReceiver
       (mReceiver, new IntentFilter(CustomReceiver.ACTION_CUSTOM_BROADCAST));
```

Untuk mencabut pendaftaran penerima siaran:

```
LocalBroadcastManager.getInstance(this).unregisterReceiver(mReceiver);
```

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Penerima Siaran

Ketahui selengkapnya

- Referensi BroadcastReceiver
- Panduan Maksud dan Filter Maksud
- Referensi LocalBroadcastManager

7.4: Layanan

Materi:

Dalam bab ini, Anda akan mempelajari tentang berbagai tipe layanan, cara menggunakannya, dan cara mengelola daur hidup layanan dalam aplikasi.

Apa yang dimaksud dengan layanan?

Layanan adalah komponen aplikasi yang menjalankan operasi yang berjalan lama, biasanya di latar belakang. Layanan tidak menyediakan antarmuka pengguna (UI). (*Aktivitas*, di sisi lain, menyediakan UI.)

Layanan bisa *dimulai*, *diikat*, atau keduanya:

- *Layanan yang dimulai* adalah layanan yang komponen aplikasinya memulai dengan memanggil `startService()`.

  Gunakan layanan yang dimulai untuk tugas yang berjalan di latar belakang guna menjalankan operasi yang berjalan lama. Selain itu, gunakan layanan yang dimulai untuk tugas yang menjalankan pekerjaan untuk proses jarak jauh.

- *Layanan terikat* adalah layanan yang komponen aplikasinya diikat ke dirinya sendiri dengan memanggil `bindService()`.

  Gunakan layanan terikat untuk tugas yang berinteraksi dengan komponen aplikasi lain guna menjalankan komunikasi interproses (IPC). Misalnya, layanan terikat mungkin menangani transaksi jaringan, menjalankan I/O file, memutar musik, atau berinteraksi dengan penyedia materi.

Catatan: Layanan berjalan di thread utama dari proses hosting-nya—layanan tidak membuat thread sendiri dan tidak berjalan di proses terpisah kecuali jika Anda menetapkannya.

Jika layanan Anda akan melakukan pekerjaan yang banyak membutuhkan CPU atau operasi pemblokiran (seperti pemutaran MP3 atau jaringan), buat thread baru dalam layanan untuk melakukan pekerjaan itu. Dengan menggunakan thread terpisah, Anda akan mengurangi risiko kesalahan Aplikasi Tidak Merespons (Application Not Responding/ANR) dan thread utama aplikasi bisa terus disediakan untuk interaksi pengguna dengan aktivitas Anda.
Untuk mengimplementasikan suatu jenis layanan dalam aplikasi Anda:

1. Deklarasikan layanan di manifes.
2. Buat kode implementasi, seperti yang dijelaskan dalam Layanan yang dimulai dan Layanan terikat, di bawah ini.
3. Kelola daur hidup layanan.

Mendeklarasikan layanan di manifes

Sebagaimana dengan aktivitas dan komponen lainnya, Anda harus mendeklarasikan semua layanan dalam file manifes aplikasi. Untuk mendeklarasikan layanan, tambahkan elemen `<service>` sebagai anak dari elemen `<application>`. Misalnya:

```
<manifest ... >
  ...
  <application ... >
      <service android:name="ExampleService"
               android:exported="false" />
      ...
 </application>
</manifest>
```

Untuk memblokir akses ke layanan dari aplikasi lainnya, deklarasikan layanan sebagai privat. Caranya, setel atribut `android:exported` ke `false`. Ini akan menghentikan aplikasi lain dari memulai layanan Anda, bahkan bila menggunakan maksud eksplisit.

Layanan yang dimulai

Cara memulai layanan:

1. Komponen aplikasi seperti aktivitas memanggil `startService()` dan meneruskannya di `Intent`. Dalam hal ini `Intent` menetapkan layanan dan menyertakan data yang akan digunakan oleh layanan.
2. Sistem akan memanggil metode `onCreate()` layanan dan callback lainnya yang sesuai di thread utama. Tergantung layanan untuk mengimplementasikan callback tersebut dengan perilaku yang sesuai, seperti membuat thread sekunder yang akan digunakan.
3. Sistem akan memanggil metode `onStartCommand()` layanan, dengan meneruskan `Intent` yang disediakan oleh klien di langkah 1. (*Klien* dalam konteks ini adalah komponen aplikasi yang memanggil layanan.)

Setelah dimulai, layanan bisa berjalan di latar belakang tanpa dibatasi waktu, bahkan jika komponen yang memulainya telah dimusnahkan. Biasanya, layanan yang dimulai menjalankan operasi tunggal dan tidak mengembalikan hasil ke pemanggil. Misalnya, layanan dapat mengunduh atau mengunggah file melalui jaringan. Bila operasi selesai, layanan harus berhenti sendiri dengan memanggil `stopSelf()`, atau komponen lain bisa menghentikannya dengan memanggil `stopService()`.

Misalnya, anggaplah aktivitas perlu menyimpan data ke database online. Aktivitas akan memulai layanan pendamping dengan meneruskan `Intent` ke `startService()`. Layanan menerima maksud di `onStartCommand()`, menghubungkan ke Internet, dan menjalankan transaksi database. Bila transaksi selesai, layanan akan menggunakan `stopSelf()` untuk menghentikan dirinya sendiri dan dimusnahkan. (Ini adalah contoh layanan yang ingin Anda jalankan di thread pekerja, sebagai ganti thread utama.)

IntentService

Sebagian besar layanan yang dimulai tidak perlu menangani beberapa permintaan secara bersamaan, dan jika layanan melakukannya, maka akan mengakibatkan skenario multi-threading yang berbahaya. Karena itu, sebaiknya Anda mengimplementasikan layanan menggunakan kelas `IntentService`.

`IntentService` adalah subkelas yang berguna dari `Service`:

- `IntentService` secara otomatis menyediakan thread pekerja untuk menangani `Intent`.
- `IntentService` menangani beberapa kode boilerplate yang diperlukan layanan umum (seperti memulai dan menghentikan layanan).
- `IntentService` bisa membuat antrean pekerjaan yang meneruskan satu maksud untuk setiap kalinya ke implementasi `onHandleIntent()`, sehingga Anda tidak perlu mengkhawatirkan multi-threading.

Untuk mengimplementasikan `IntentService`:

1. Sediakan konstruktor kecil untuk layanan.
2. Buat implementasi `onHandleIntent()` untuk melakukan pekerjaan yang disediakan klien.

Inilah contoh implementasi `IntentService`:

```
public class HelloIntentService extends IntentService {
  /**
   * A constructor is required, and must call the super IntentService(String)
   * 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 we would do some work here, like download a file.
      // For our sample, we just sleep for 5 seconds.
      try {
          Thread.sleep(5000);
      } catch (InterruptedException e) {
          // Restore interrupt status.
          Thread.currentThread().interrupt();
      }
  }
}
```

Layanan terikat

Layanan "terikat" bila komponen aplikasi mengikatnya dengan memanggil `bindService()` . Layanan terikat menawarkan antarmuka klien-server yang memungkinkan komponen berinteraksi dengan layanan, mengirim permintaan, dan mendapatkan hasil, kadang-kadang menggunakan komunikasi interproses (IPC) untuk mengirim dan menerima informasi di seluruh proses. Layanan terikat hanya berjalan selama komponen aplikasi terikat padanya. Beberapa komponen bisa diikat ke layanan sekaligus, namun bila semuanya telah dilepas, layanan akan dimusnahkan.

Layanan terikat umumnya tidak mengizinkan memulai komponen dengan memanggil `startService()` .

Mengimplementasikan layanan terikat

Untuk mengimplementasikan layanan terikat, definisikan antarmuka yang menetapkan cara klien bisa berkomunikasi dengan layanan. Antarmuka ini, yang dikembalikan layanan Anda dari metode callback `onBind()` , harus berupa implementasi `IBinder` .

Untuk mengambil antarmuka `IBinder` , komponen aplikasi klien memanggil `bindService()` . Setelah klien menerima `IBinder` , klien berinteraksi dengan layanan melalui antarmuka itu.

Ada sejumlah cara untuk mengimplementasikan layanan terikat, dan implementasi tersebut lebih rumit daripada layanan yang dimulai. Untuk detail selengkapnya tentang layanan terikat, lihat Layanan Terikat.

Mengikat ke layanan

Untuk mengikat ke layanan yang dideklarasikan di manifes dan diimplementasikan oleh komponen aplikasi, gunakan `bindService()` dengan `Intent` eksplisit.

Perhatian: Jangan gunakan maksud implisit untuk mengikat ke layanan. Melakukannya adalah bahaya keamanan, karena Anda tidak bisa memastikan layanan yang akan merespons maksud tersebut, dan pengguna tidak bisa melihat layanan mana yang dimulai. Mulai dengan Android 5.0 (API level 21), sistem membuat pengecualian jika Anda memanggil `bindService()` dengan `Intent` implisit.

Daur hidup layanan

Daur hidup layanan lebih sederhana daripada aktivitas. Akan tetapi, ini jauh lebih penting karena Anda memerhatikan dari dekat cara layanan dibuat dan dimusnahkan. Karena tidak memiliki UI, layanan bisa terus berjalan di latar belakang tanpa diketahui pengguna, bahkan jika pengguna beralih ke aplikasi lain. Ini menghabiskan sumber daya dan menguras baterai.

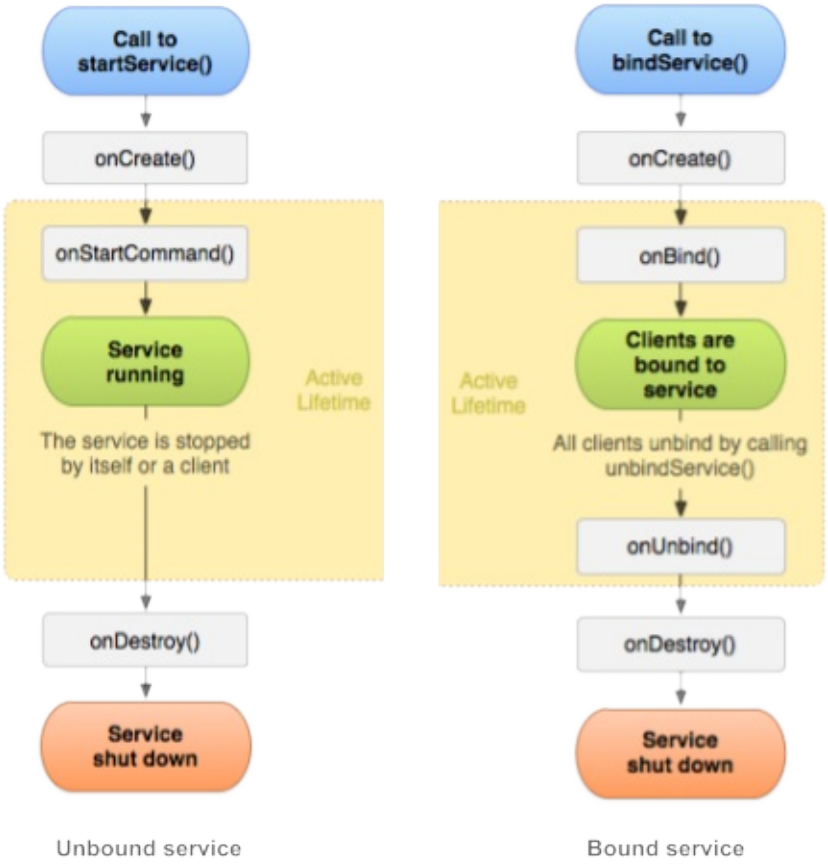
Seperti aktivitas, layanan memiliki metode callback daur hidup yang bisa Anda implementasikan untuk memantau perubahan keadaan layanan dan melakukan pekerjaan pada waktu yang sesuai. Layanan kerangka berikut memperagakan setiap metode daur hidup:

```
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
    }
}
```

Daur hidup layanan yang dimulai vs. layanan terikat

Layanan terikat hanya tersedia untuk menyajikan komponen aplikasi yang terikat padanya, sehingga bila tidak ada lagi komponen yang diikat ke layanan tersebut, sistem akan memusnahkannya. Layanan terikat tidak perlu dihentikan secara eksplisit seperti halnya layanan yang dimulai (menggunakan `stopService()` atau `stopSelf()`).

Diagram di bawah ini menampilkan perbandingan antara daur hidup layanan yang dimulai dan terikat.



Layanan latar depan

Walaupun sebagian besar layanan berjalan di latar belakang, sebagian lagi ada yang berjalan di latar depan. *Layanan latar depan* adalah layanan yang diketahui pengguna, jadi ini bukan layanan yang bakal dimatikan sistem bila memori tinggal sedikit.

Misalnya, pemutar musik yang memutar musik dari layanan harus disetel untuk berjalan di latar depan, karena pengguna mengetahui operasinya. Notifikasi di bilah status dapat menunjukkan lagu saat ini dan memungkinkan pengguna meluncurkan aktivitas untuk berinteraksi dengan pemutar musik.

Untuk meminta agar layanan berjalan di latar depan, panggil `startForeground()` sebagai ganti `startService()`. Metode ini menggunakan dua parameter: integer yang secara unik mengidentifikasi notifikasi dan `Notification` untuk bilah status. Notifikasi ini *sedang berlangsung*, artinya tidak bisa ditutup. Notifikasi tetap berada di bilah status hingga layanan dihentikan atau dibuang dari latar depan.

Misalnya:

```
NotificationCompat.Builder mBuilder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("My notification")
        .setContentText("Hello World!");
startForeground(ONGOING_NOTIFICATION_ID, mBuilder.build());
```

Catatan: ID integer yang Anda berikan ke `startForeground()` tidak boleh 0.
Untuk membuang layanan dari latar depan, panggil `stopForeground()`. Metode ini memerlukan boolean, yang menunjukkan apakah akan membuang notifikasi bilah status atau tidak. Metode ini tidak menghentikan layanan. Akan tetapi, jika Anda menghentikan layanan sewaktu masih berjalan di latar depan, maka notifikasi juga akan dibuang.

Layanan terjadwal

Untuk API level 21 dan yang lebih tinggi, Anda bisa meluncurkan layanan menggunakan `JobScheduler` API. Untuk menggunakan `JobScheduler` , Anda perlu mendaftarkan tugas dan menetapkan persyaratannya untuk jaringan dan pengaturan waktu. Sistem menjadwalkan tugas untuk dieksekusi di waktu yang tepat.

Antarmuka `JobScheduler` menyediakan banyak metode untuk mendefinisikan ketentuan eksekusi layanan. Untuk detailnya, lihat `JobScheduler reference` .

Ketahui selengkapnya

- Panduan layanan
- Menjalankan Layanan Latar Belakang

8.1: Notifikasi

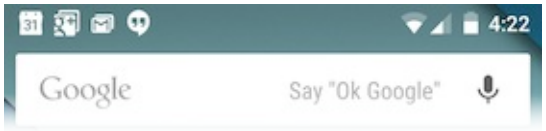
Materi:

- Pengantar
- Apa yang dimaksud dengan notifikasi?
- Membuat notifikasi
- Mengirim notifikasi
- Menggunakan kembali notifikasi
- Mengosongkan notifikasi
- Kompatibilitas notifikasi
- Panduan desain notifikasi
- Praktik terkait
- Ketahui selengkapnya

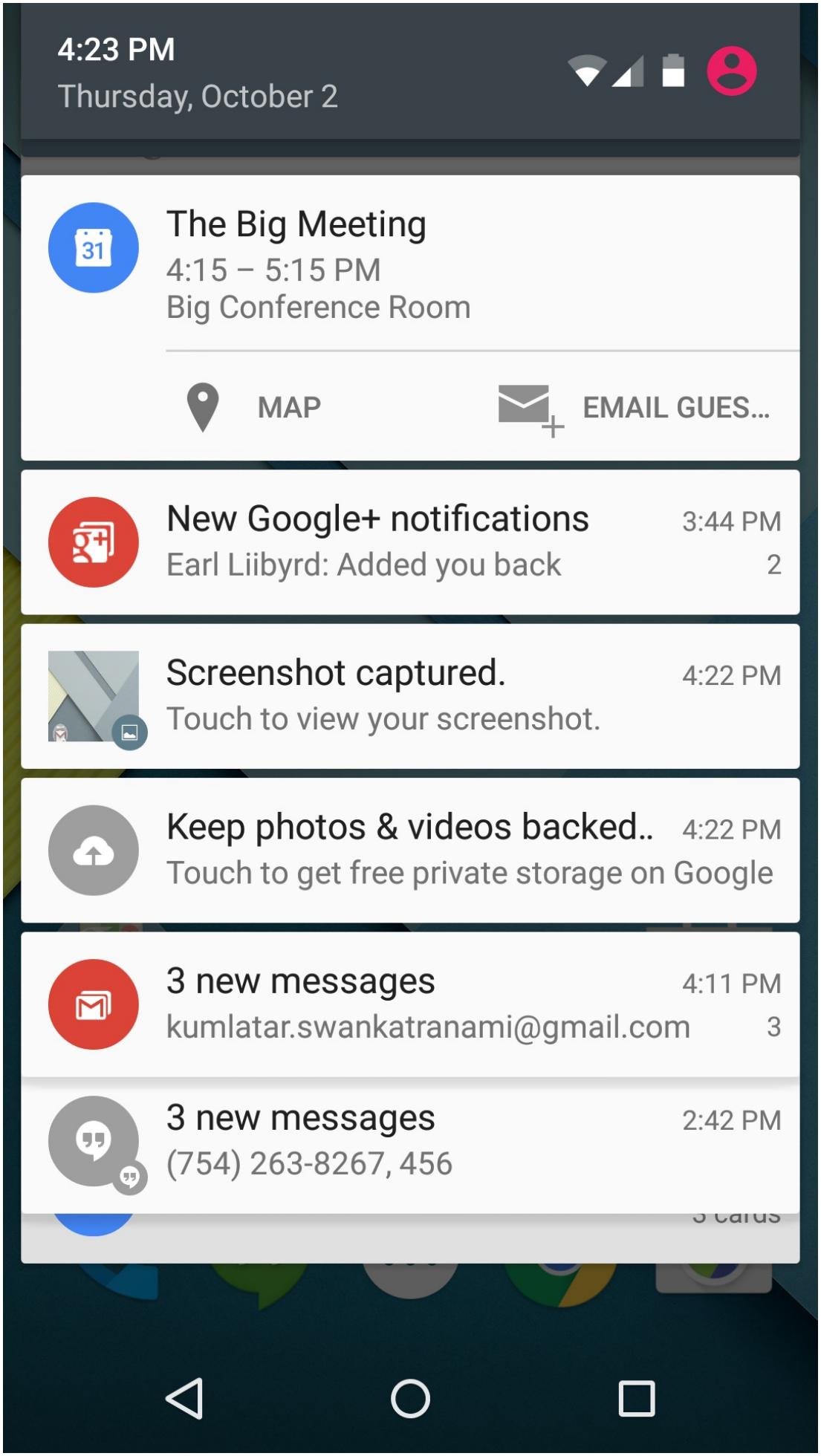
Dalam bab ini, Anda akan mempelajari cara membuat, mengirim, dan menggunakan kembali notifikasi, serta cara membuatnya kompatibel dengan versi Android yang berbeda.

Apa yang dimaksud dengan notifikasi?

Notifikasi adalah pesan yang ditampilkan aplikasi kepada pengguna di luar UI normal aplikasi. Bila Anda memberi tahu sistem untuk mengeluarkan notifikasi, notifikasi terlebih dahulu akan muncul kepada pengguna berupa ikon di *area notifikasi*, di sebelah kiri bilah status.



Untuk melihat detail notifikasi, pengguna membuka *panel samping notifikasi*, atau menampilkan notifikasi pada layar kunci jika perangkat terkunci. Area notifikasi, layar kunci, dan panel samping notifikasi merupakan area yang dikontrol sistem yang bisa ditampilkan pengguna kapan saja.


4:23 PM
Thursday, October 2
The Big Meeting
4:15 – 5:15 PM
Big Conference Room
MAP
EMAIL GUES...
New Google+ notifications
3:44 PM
Earl Liibyrd: Added you back
2
Screenshot captured.
4:22 PM
Touch to view your screenshot.
Keep photos & videos backed..
4:22 PM
Touch to get free private storage on Google
3 new messages
4:11 PM
kumlatar.swankatranami@gmail.com
3
3 new messages
2:42 PM
(754) 263-8267, 456

Tangkapan layar menampilkan panel samping notifikasi yang "terbuka". Bilah status tidak terlihat, karena panel samping notifikasi terbuka.

Proses ini dijelaskan di bawah ini.

Membuat notifikasi

Anda membuat notifikasi menggunakan kelas `NotificationCompat.Builder` . (Gunakan `NotificationCompat` untuk kompatibilitas mundur terbaik. Untuk informasi selengkapnya, lihat Kompatibilitas notifikasi.) Kelas pembangun menyederhanakan pembuatan objek yang kompleks.

Untuk membuat `NotificationCompat.Builder` , teruskan konteks aplikasi ke konstruktor:

```
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this);
```

Menyetel komponen notifikasi

Saat menggunakan NotificationCompat.Builder, Anda harus menetapkan ikon kecil, teks untuk judul, dan pesan notifikasi. Anda harus mempertahankan pesan notifikasi kurang dari 40 karakter dan tidak mengulangi apa yang ada di judul. Misalnya:

```
NotificationCompat.Builder mBuilder =
    new NotificationCompat.Builder(this)
    .setSmallIcon(R.drawable.notification_icon)
    .setContentTitle("Dinner is ready!")
    .setContentText("Lentil soup, rice pilaf, and cake for dessert.");
```

Anda juga perlu menyetel `Intent` yang menentukan apa yang terjadi bila pengguna mengeklik notifikasi. Biasanya Intent ini mengakibatkan aplikasi meluncurkan Aktivitas.

Untuk memastikan sistem mengirimkan `Intent` bahkan bila aplikasi sedang tidak dijalankan bila pengguna mengeklik notifikasi, bungkus `Intent` dalam objek `PendingIntent` , yang memungkinkan sistem mengirimkan `Intent` apa pun keadaan aplikasi.

Untuk membuat instance `PendingIntent` , gunakan salah satu metode berikut, bergantung pada bagaimana Anda ingin `Intent` yang dimuat akan dikirim:

- Untuk meluncurkan Aktivitas bila pengguna mengeklik notifikasi, gunakan `PendingIntent.getActivity()` , dengan meneruskan `Intent` eksplisit untuk Aktivitas yang ingin diluncurkan. Metode `getActivity()` sesuai dengan `Intent` yang dikirim menggunakan `startActivity()` .
- Untuk `Intent` yang diteruskan ke dalam `startService()` (misalnya layanan untuk mengunduh file), gunakan `PendingIntent.getService()` .
- Untuk `Intent` siaran yang dikirim bersama `sendBroadcast()` , gunakan `PendingIntent.getBroadcast()` .

Setiap metode `PendingIntent` ini menggunakan argumen berikut:

- Konteks aplikasi.
- Kode permintaan, yang merupakan ID integer konstanta untuk `PendingIntent` .
- Objek `Intent` yang akan dikirim.
- Flag `PendingIntent` yang menentukan cara sistem menangani beberapa objek `PendingIntent` dari aplikasi yang sama.

Misalnya:

```
Intent contentIntent = new Intent(this, ExampleActivity.class);
PendingIntent pendingContentIntent = PendingIntent.getActivity(this, 0,
    contentIntent, PendingIntent.FLAG_UPDATE_CURRENT);
mBuilder.setContentIntent(pendingContentIntent);
```

Untuk mengetahui selengkapnya tentang `PendingIntent`, lihat `PendingIntent documentation`.

Komponen opsional

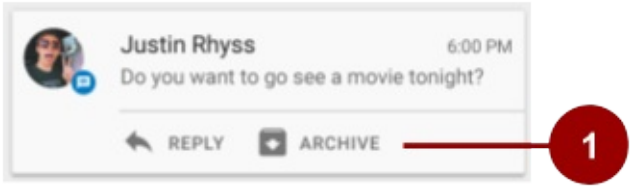
Anda bisa menggunakan beragam opsi bersama notifikasi, termasuk:

- Tindakan notifikasi
- Prioritas
- Layout yang diluaskan
- Notifikasi yang berlangsung

Untuk opsi lainnya yang bisa Anda gunakan bersama notifikasi, lihat referensi `NotificationCompat.Builder`.

Tindakan notifikasi

Aksi notifikasi adalah aksi yang bisa diambil pengguna pada notifikasi. Aksi tersedia melalui tombol aksi di notifikasi. Seperti `Intent` yang menentukan apa yang terjadi bila pengguna mengeklik notifikasi, aksi notifikasi menggunakan `PendingIntent` untuk melakukan aksi. Sistem Android biasanya menampilkan aksi notifikasi berupa tombol yang berdekatan dengan isi notifikasi. Mulai dengan Android 4.1 (API level 16), notifikasi mendukung ikon yang disematkan di bawah teks isi, seperti yang ditampilkan dalam tangkapan layar di bawah ini.



1. Notifikasi ini memiliki dua tindakan yang bisa diambil pengguna, "Reply" atau "Archive". Masing-masing memiliki ikon.

Untuk menambahkan aksi notifikasi, gunakan metode `addAction()` bersama objek `NotificationCompat.Builder.` Teruskan di ikon, string judul dan `PendingIntent` untuk dipicu bila pengguna mengetuk aksi. Misalnya:

```
mBuilder.addAction(R.drawable.car, "Get Directions", mapPendingIntent);
```

Untuk memastikan fungsionalitas tombol aksi selalu tersedia, ikuti petunjuk di bagian Kompatibilitas notifikasi, di bawah ini.

Prioritas notifikasi

Android memungkinkan Anda menetapkan level prioritas ke setiap notifikasi untuk memengaruhi cara sistem Android akan mengirimkannya. Notifikasi memiliki prioritas antara `MIN` (`-2`) dan `MAX` (`2`) yang sesuai dengan kepentingannya. Tabel berikut menampilkan konstanta prioritas yang tersedia yang didefinisikan di kelas `Notification`.

Konstanta Prioritas	Penggunaan
PRIORITY_MAX	Untuk notifikasi mendesak dan urgen yang memperingatkan pengguna terhadap kondisi yang didesak-waktu atau perlu diatasi sebelum bisa melanjutkan dengan tugas yang didesak-waktu.
PRIORITY_HIGH	Terutama untuk komunikasi penting, seperti pesan atau chat.
PRIORITY_DEFAULT	Untuk semua notifikasi yang tidak termasuk dalam salah satu prioritas lain yang dijelaskan di sini.
PRIORITY_LOW	Untuk informasi dan kejadian yang berharga atau relevan secara kontekstual, namun tidak urgen atau didesak-waktu.
PRIORITY_MIN	Untuk informasi latar belakang yang perlu diketahui. Misalnya, cuaca atau tempat menarik terdekat.

Untuk mengubah prioritas notifikasi, gunakan metode `setPriority()` pada objek `NotificationCompat.Builder` , dengan meneruskan di salah satu konstanta di atas.

```
mBuilder.setPriority(Notification.PRIORITY_HIGH);
```

Notifikasi bisa jadi mengganggu. Menggunakan prioritas notifikasi dengan benar adalah langkah pertama untuk memastikan pengguna tidak mencopot pemasangan aplikasi Anda karena terlalu mengganggu.

Mengintip

Notifikasi dengan prioritas `HIGH` atau `MAX` bisa *mengintip*, yang artinya bergeser sedikit ke dalam tampilan di layar pengguna saat ini, aplikasi apa pun yang digunakan pengguna. Perhatikan, di perangkat yang menjalankan Android 6.0 dan yang lebih tinggi, pengguna bisa memblokir pengintipan dengan mengubah setelan "App notification" perangkat. Artinya Anda tidak bisa mengandalkan pengintipan notifikasi, sekalipun Anda telah mempersiapkannya sedemikian rupa.

Untuk membuat notifikasi yang bisa mengintip:

1. Setel prioritas ke `HIGH` atau `MAX` .
2. Setel suara atau pola lampu menggunakan metode `setDefaults()` pada pembangun, yang meneruskan konstanta `DEFAULTS_ALL` . Tindakan ini akan memberikan suara, pola lampu, dan getaran default pada notifikasi.

```
NotificationCompat.Builder mBuilder =
    new NotificationCompat.Builder(this)
    .setSmallIcon(R.drawable.notification_icon)
    .setContentTitle("My notification")
    .setContentText("Hello World!")
    .setPriority(PRIORITY_HIGH)
    .setDefaults(DEFAULTS_ALL);
```

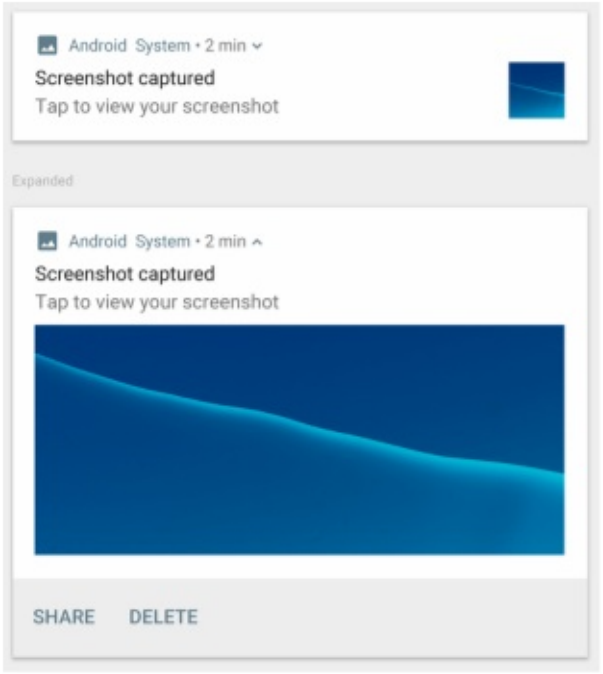
Layout tampilan yang diluaskan

Notifikasi di panel samping notifikasi muncul dalam dua layout utama, *tampilan normal* (yang merupakan default) dan *tampilan yang diluaskan*. Notifikasi tampilan yang diluaskan diperkenalkan dalam Android 4.1. Gunakan dengan hemat, karena tampilan tersebut menggunakan lebih banyak ruang dan perhatian daripada layout tampilan normal.

Untuk membuat notifikasi muncul di layout yang diluaskan, gunakan salah satu kelas helper ini:

- Penggunaan `NotificationCompat.BigTextStyle` untuk notifikasi berformat besar yang menyertakan banyak teks.
- Penggunaan `NotificationCompat.InboxStyle` untuk notifikasi berformat besar yang menyertakan daftar berisi hingga lima string.

- Penggunaan `Notification.MediaStyle` untuk notifikasi pemutaran media. Saat ini tidak ada versi `NotificationCompat` untuk gaya ini, sehingga hanya bisa digunakan pada perangkat Android 4.1 atau di atasnya. Lihat bagian Kompatibilitas notifikasi untuk informasi selengkapnya.
- Penggunaan `NotificationCompat.BigPictureStyle`, yang ditampilkan di tangkapan layar di bawah ini, untuk notifikasi berformat besar yang menyertakan lampiran gambar besar.



Misalnya, inilah cara menyetel `BigPictureStyle` di notifikasi:

```
NotificationCompat notif = new NotificationCompat.Builder(mContext)
    .setContentTitle("New photo from " + sender.toString())
    .setContentText(subject)
    .setSmallIcon(R.drawable.new_post)
    .setLargeIcon(aBitmap)
    .setStyle(new NotificationCompat.BigPictureStyle()
        .bigPicture(aBigBitmap)
        .setBigContentTitle("Large Notification Title"))
    .build();
```

Untuk mengetahui selengkapnya tentang mengimplementasikan gaya yang diluaskan, lihat dokumentasi NotificationCompat.Style.

Notifikasi yang berlangsung

Notifikasi yang berlangsung adalah notifikasi yang tidak bisa ditutup oleh pengguna. Aplikasi harus membatalkan notifikasi tersebut secara eksplisit dengan memanggil `cancel()` atau `cancelAll()`. Membuat beberapa notifikasi yang berlangsung akan mengganggu pengguna karena mereka tidak bisa membatalkan notifikasi tersebut. Gunakan notifikasi yang berlangsung dengan hemat.

Untuk membuat notifikasi yang berlangsung, setel `setOngoing()` ke `true`. Gunakan notifikasi yang berlangsung untuk menunjukkan tugas latar belakang yang aktif berinteraksi dengan pengguna (seperti memainkan musik) atau tugas yang sedang berlangsung di perangkat (misalnya pengunduhan file, operasi sinkronisasi, dan koneksi jaringan aktif).

Mengirim notifikasi

Gunakan kelas `NotificationManager` untuk mengirim notifikasi:

1. Panggil `getSystemService()`, dengan meneruskan di konstanta `NOTIFICATION_SERVICE`, untuk membuat instance `NotificationManager`.
2. Panggil `notify()` untuk mengirimkan notifikasi. Dalam metode `notify()`, teruskan kedua nilai ini:
   - ID notifikasi, yang digunakan untuk memperbarui atau membatalkan notifikasi.
   - Objek `NotificationCompat` yang Anda buat menggunakan objek `NotificationCompat.Builder`.

Contoh berikut membuat instance `NotificationManager`, kemudian membangun dan mengirimkan notifikasi:

```
mNotifyManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);

//Builds the notification with all the parameters
NotificationCompat.Builder notifyBuilder = new NotificationCompat.Builder(this)
       .setContentTitle(getString(R.string.notification_title))
       .setContentText(getString(R.string.notification_text))
       .setSmallIcon(R.drawable.ic_android)
       .setContentIntent(notificationPendingIntent)
       .setPriority(NotificationCompat.PRIORITY_HIGH)
       .setDefaults(NotificationCompat.DEFAULT_ALL);

//Delivers the notification
mNotifyManager.notify(NOTIFICATION_ID, notifyBuilder.build());
```

Menggunakan kembali notifikasi

Bila Anda perlu mengeluarkan notifikasi beberapa kali untuk tipe kejadian yang sama, Anda bisa memperbarui notifikasi sebelumnya dengan mengubah beberapa nilainya, menambahkan ke notifikasi, atau keduanya.

Untuk menggunakan kembali notifikasi yang ada:

1. Perbarui objek `NotificationCompat.Builder` dan bangun objek `Notification` dari sana, seperti saat Anda pertama kali membuat dan membangun notifikasi.
2. Kirim notifikasi bersama ID yang sama dengan yang digunakan sebelumnya.
   Penting: Jika notifikasi sebelumnya masih terlihat, sistem akan memperbaruinya dari isi objek `Notification`. Jika notifikasi sebelumnya sudah ditutup, notifikasi baru akan dibuat.

Mengosongkan notifikasi

Notifikasi tetap terlihat hingga salah satu hal berikut terjadi:

- Jika bisa dikosongkan, notifikasi akan menghilang bila pengguna menutupnya atau dengan menggunakan "Clear All".
- Jika Anda memanggil `setAutoCancel()` saat membuat notifikasi, notifikasi akan menghilang bila pengguna mengekliknya.
- Jika Anda memanggil `cancel()` untuk ID notifikasi tertentu, notifikasi akan menghilang.
- Jika Anda memanggil `cancelAll()`, semua notifikasi yang telah dikeluarkan akan menghilang.

Karena notifikasi yang berlangsung tidak bisa ditutup oleh pengguna, aplikasi harus membatalkannya dengan memanggil `cancel()` atau `cancelAll()`.

Kompatibilitas notifikasi

Untuk memastikan kompatibilitas terbaik, buat notifikasi dengan `NotificationCompat` dan subkelasnya, terutama `NotificationCompat.Builder`.

Ingatlah bahwa tidak semua fitur notifikasi tersedia untuk setiap versi Android, meskipun metode untuk menyetelnya ada di kelas pustaka dukungan `NotificationCompat.Builder`. Misalnya, layout tampilan yang diluaskan untuk notifikasi hanya tersedia di Android 4.1 dan yang lebih tinggi, namun tombol aksi bergantung pada layout tampilan yang diluaskan. Artinya

jika Anda menggunakan tombol aksi notifikasi, tombol itu tidak akan ditampilkan di perangkat yang menjalankan apa saja sebelum Android 4.1.

Untuk mengatasinya:

- Jangan mengandalkan tombol aksi notifikasi untuk melakukan aksi notifikasi; sebagai gantinya buatlah aksi yang tersedia di Aktivitas. Anda mungin perlu menambahkan Aktivitas baru untuk melakukannya.

  Misalnya, jika Anda menyetel aksi notifikasi yang menyediakan kontrol untuk menghentikan dan memulai pemutaran media, pertama-tama implementasikan kontrol ini dalam Aktivitas di aplikasi Anda.

- Mulailah Aktivitas bila pengguna mengeklik notifikasi. Caranya:

  1. Buat `PendingIntent` untuk Aktivitas.
  2. Panggil `setContentIntent()` untuk menambahkan `PendingIntent` ke notifikasi.

- Penggunaan `addAction()` untuk menambahkan fitur ke notifikasi jika perlu. Ingatlah bahwa fungsionalitas apa pun yang ditambahkan juga harus tersedia di Aktivitas yang dimulai bila pengguna mengeklik notifikasi.

Panduan desain notifikasi

Notifikasi selalu mengganggu pengguna. Karena itu, notifikasi harus singkat, tepat waktu, dan yang terpenting, relevan.

- **Relevan:** Tanyakan pada diri sendiri apakah informasi ini sangat penting untuk pengguna. Apa yang terjadi jika mereka tidak mendapatkan notifikasi? Misalnya, acara kalender yang dijadwalkan kemungkinan adalah relevan.
- **Tepat Waktu:** Notifikasi perlu muncul ketika berguna. Misalnya, memberi tahu pengguna bila tiba waktu untuk berangkat ke janji temu adalah hal yang berguna.
- **Singkat:** Gunakan kata sesedikit mungkin. Sekarang, tantanglah diri sendiri untuk mengatakannya dengan lebih sedikit.

Berikan kemampuan untuk memilih kepada pengguna:

- Sediakan setelan di aplikasi Anda yang memungkinkan pengguna memilih jenis notifikasi yang ingin diterima dan cara menerimanya.

Selain prinsip dasar ini, notifikasi memiliki panduan desain sendiri:

- Untuk mengetahui cara mendesain notifikasi dan interaksinya, lihat dokumentasi pola notifikasi Desain Material.
- Untuk mempelajari cara mendesain notifikasi dan interaksinya bagi versi Android yang lebih lama, lihat Notifikasi, Android 4.4 dan yang lebih rendah.
- Untuk detail penting tentang perubahan Desain Material yang diperkenalkan di Android 5.0 API (level 21), lihat pelatihan Desain Material.

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Notifikasi

Ketahui selengkapnya

Panduan

- Notifikasi
- Panduan desain notifikasi

Referensi

- Referensi NotificationCompat.Builder
- Referensi NotificationCompat.Style

8.2: Menjadwalkan Alarm

Materi:

- Pengantar
- Tipe alarm
- Praktik terbaik alarm
- Menjadwalkan alarm
- Memeriksa alarm yang ada
- Membatalkan alarm
- Alarm yang terlihat pengguna ("jam alarm")
- Praktik terkait
- Ketahui selengkapnya

Anda sudah mengetahui cara menggunakan penerima siaran untuk membuat aplikasi merespons kejadian sistem bahkan bila aplikasi sedang tidak dijalankan. Dalam bab ini, Anda akan mempelajari cara menggunakan alarm untuk menjadwalkan tugas selama waktu tertentu, baik aplikasi dijalankan pada saat alarm disetel untuk aktif maupun tidak. Alarm bisa untuk sekali penggunaan atau berulang. Misalnya, Anda bisa menggunakan alarm berulang untuk menjadwalkan pengunduhan setiap hari pada waktu yang sama.

Untuk membuat alarm, gunakan kelas `AlarmManager` . Alarm di Android memiliki karakteristik berikut:

- Alarm memungkinkan Anda mengirim maksud pada waktu atau interval yang disetel. Anda bisa menggunakan alarm bersama penerima siaran untuk memulai layanan dan menjalankan operasi lainnya.
- Alarm beroperasi di luar aplikasi, sehingga Anda bisa menggunakannya untuk memicu kejadian atau tindakan bahkan saat aplikasi sedang tidak dijalankan, dan bahkan jika perangkat nonaktif.
- Bila digunakan dengan benar, alarm bisa membantu Anda meminimalkan kebutuhan sumber daya aplikasi. Misalnya, Anda bisa menjadwalkan operasi tanpa mengandalkan timer atau terus menjalankan layanan latar belakang.

Bila *tidak* menggunakan alarm:

- Untuk kejadian pengaturan waktu seperti tick dan waktu tunggu, dan untuk operasi berwaktu yang dipastikan terjadi selama masa aplikasi Anda, gunakan kelas `Handler` bersama `Timer` dan `Thread` . Pendekatan ini memberi Android kontrol yang lebih baik atas sumber daya sistem daripada jika Anda menggunakan alarm.
- Untuk operasi sinkronisasi server, gunakan `SyncAdapter` bersama Google Cloud Messaging Service.
- Untuk tugas yang bisa menunggu hingga kondisi memungkinkan, seperti bila perangkat terhubung ke WiFi dan sedang mengisi daya (misalnya, memperbarui informasi cuaca atau kabar berita), Anda mungkin tidak ingin menggunakan alarm. Untuk tugas ini di perangkat API 21+, pertimbangkan penggunaan `JobScheduler` , yang akan Anda pelajari di pelajaran mendatang.

Tipe alarm

Ada dua tipe alarm umum di Android: *alarm waktu tempuh riil* serta alarm *jam real-time (RTC)*, dan keduanya menggunakan objek `PendingIntent` .

Alarm waktu tempuh riil

Alarm waktu tempuh riil menggunakan waktu, dalam milidetik, sejak perangkat melakukan booting. Alarm waktu tempuh riil tidak terpengaruh oleh zona waktu, sehingga bekerja dengan baik untuk alarm berdasarkan waktu yang telah ditempuh. Misalnya, gunakan alarm waktu tempuh riil untuk alarm yang aktif setiap setengah jam.

Kelas AlarmManager menyediakan dua tipe alarm waktu tempuh riil:

- `ELAPSED_REALTIME` : Akan memicu `PendingIntent` berdasarkan pada jumlah waktu sejak perangkat booting, namun

tidak membangunkan perangkat. Waktu tempuh menyertakan waktu selama perangkat sedang tidur. Semua alarm berulang akan terpicu bila perangkat bangun pada waktu berikutnya.

- `ELAPSED_REALTIME_WAKEUP` : Akan memicu `PendingIntent` setelah jangka waktu yang ditetapkan berlalu sejak perangkat booting, yang membangunkan CPU perangkat jika layar mati. Gunakan alarm ini sebagai ganti `ELAPSED_REALTIME` jika aplikasi Anda memiliki dependensi waktu, misalnya jika memiliki jendela terbatas selama menjalankan operasi.

Alarm jam real-time (RTC)

Alarm jam real-time (RTC) adalah alarm berbasis jam yang menggunakan Coordinated Universal Time (UTC). Hanya pilih alarm RTC dalam tipe situasi ini:

- Anda perlu alarm yang dipicu pada waktu tertentu dalam sehari.
- Waktu alarm bergantung pada lokal saat ini.

Aplikasi dengan alarm berbasis jam mungkin tidak berfungsi dengan baik di seluruh lokal, karena mungkin akan terpicu pada waktu yang salah. Dan jika pengguna mengubah setelan waktu perangkat, maka hal itu bisa menyebabkan perilaku yang tidak diharapkan di aplikasi Anda.

Kelas `AlarmManager` menyediakan dua tipe alarm RTC:

- `RTC` : Akan memicu maksud yang menunggu pada waktu yang ditetapkan, namun tidak membangunkan perangkat. Semua alarm berulang akan terpicu bila perangkat bangun pada waktu berikutnya.
- `RTC_WAKEUP` : Akan memicu maksud yang menunggu pada waktu yang ditetapkan, yang membangunkan CPU perangkat jika layar mati.

Praktik terbaik alarm

Alarm memengaruhi cara aplikasi Anda menggunakan (atau menyalahgunakan) sumber daya sistem. Misalnya, bayangkan aplikasi populer yang menyinkronkan dengan server. Jika operasi sinkronisasi berdasarkan pada waktu jam dan setiap instance aplikasi menghubungkan ke server pada waktu yang sama, beban pada server bisa mengakibatkan waktu respons tertunda atau bahkan kondisi "denial of service".

Untuk menghindari masalah ini dan masalah lainya, ikuti praktik terbaik ini:

- Tambahkan keacakan (jitter) ke permintaan jaringan yang terpicu akibat alarm berulang. Inilah satu cara untuk melakukannya:
  - Jadwalkan alarm pasti yang menjalankan suatu pekerjaan lokal. "Pekerjaan lokal" berarti sesuatu yang tidak menghubungi server melalui jaringan atau memerlukan data dari server.
  - Jadwalkan alarm terpisah yang berisi permintaan jaringan, dan setel alarm ini agar terpicu setelah periode waktu acak. Biasanya alarm kedua ini disetel oleh komponen apa pun yang menerima `PendingIntent` dari alarm pertama. (Anda juga bisa menyetel alarm ini pada waktu yang sama seperti alarm pertama.)
- Pertahankan frekuensi alarm Anda tetap minimum.
- Jangan bangunkan perangkat jika tidak diperlukan.
- Gunakan pengaturan waktu yang paling tidak akurat untuk memungkinkan `AlarmManager` menjadi yang paling efisien. Misalnya, bila Anda menjadwalkan alarm berulang, gunakan `setInexactRepeating()` sebagai ganti `setRepeating()` . Untuk detailnya, lihat Menjadwalkan alarm berulang, di bawah ini.
- Hindari mendasarkan alarm pada waktu jam dan gunakan `ELAPSED_REALTIME` untuk alarm berulang bila memungkinkan. Alarm berulang yang berdasarkan pada waktu pemicu akurat tidak akan menskalakan dengan baik.

Menjadwalkan alarm

Kelas `AlarmManager` memberi Anda akses ke layanan alarm sistem Android. `AlarmManager` memungkinkan Anda menyiarkan `Intent` pada waktu yang dijadwalkan, atau setelah interval tertentu.

Untuk menjadwalkan alarm:

1. Panggil `getSystemService(ALARM_SERVICE)` untuk mendapatkan instance kelas `AlarmManager` .
2. Gunakan salah satu metode `set...()` yang tersedia di `AlarmManager` (sebagaimana dijelaskan di bawah ini). Metode yang Anda gunakan bergantung pada apakah alarm telah menempuh waktu yang riil, atau RTC.

Semua metode `AlarmManager.set...()` menyertakan dua argumen:

- Argumen `type` , yakni cara Anda menetapkan tipe alarm:
  - `ELAPSED_REALTIME` atau `ELAPSED_REALTIME_WAKEUP` , yang dijelaskan di Alarm waktu tempuh riil di atas.
  - `RTC` atau `RTC_WAKEUP` , yang dijelaskan di Alarm jam real-time (RTC) di atas.
- Objek `PendingIntent` , yakni cara Anda menetapkan tugas yang akan dijalankan pada waktu yang diberikan.

Menjadwalkan alarm sekali-pakai

Untuk menjadwalkan satu alarm, gunakan salah satu metode berikut ini di instance `AlarmManager` :

- `set()` : Untuk perangkat yang menjalankan API 19+, metode ini menjadwalkan satu alarm berwaktu tidak pasti, maksudnya, sistem akan menggeser alarm untuk meminimalkan membangunkan dan penggunaan baterai. Untuk perangkat yang menjalankan versi API lebih rendah, metode ini menjadwalkan alarm yang berwaktu pasti.
- `setWindow()` : Untuk perangkat yang menjalankan API 19+, gunakan metode ini untuk menyetel jangka waktu untuk memicu alarm.
- `setExact()` : Untuk perangkat yang menjalankan API 19+, metode ini memicu alarm pada waktu yang pasti. Gunakan metode ini hanya untuk alarm yang harus dipicu pada waktu yang pasti, misalnya jam alarm yang berdering pada waktu yang diminta. Alarm pasti akan mengurangi kemampuan OS untuk meminimalkan penggunaan baterai, jadi jangan menggunakannya jika tidak diperlukan.

Inilah contoh penggunaan `set()` untuk menjadwalkan alarm sekali-pakai:

```
alarmMgr.set(AlarmManager.ELAPSED_REALTIME,
             SystemClock.elapsedRealtime() + 1000*300,
             alarmIntent);
```

Dalam contoh ini:

- Kelas `type` adalah `ELAPSED_REALTIME` , yang berarti bahwa ini adalah alarm waktu tempuh riil. Jika perangkat sedang tidak digunakan saat alarm dikirim, alarm tidak akan membangunkan perangkat.
- Alarm dikirim 5 menit (300.000 milidetik) setelah metode dikembalikan.
- `alarmIntent` adalah siaran `PendingIntent` berisi aksi yang akan dijalankan ketika alarm dikirim.
  Catatan: Untuk operasi pengaturan waktu seperti tick dan waktu tunggu, serta kejadian yang lebih dari sekali dalam satu menit, akan lebih mudah dan efisien menggunakan Penangan daripada alarm.

Istirahatkan dan Aplikasi Siaga

Perangkat API 23+ kadang-kadang masuk ke mode Istirahatkan atau Aplikasi Siaga untuk menghemat daya:

- Mode *Istirahatkan* dipicu bila pengguna meninggalkan perangkat yang stekernya tidak terhubung dan tidak bergerak selama periode waktu tertentu, dengan layar dimatikan. Selama "masa pemeliharaan" singkat, sistem akan keluar dari Istirahatkan untuk memungkinkan aplikasi menyelesaikan aktivitas yang ditangguhkan, termasuk memicu alarm standar, kemudian kembali ke Istirahatkan. Mode Istirahatkan berakhir bila pengguna kembali ke perangkat mereka.
- Mode *Aplikasi Siaga* dipicu pada aplikasi diam yang tidak digunakan baru-baru ini. Mode Aplikasi Siaga berakhir bila pengguna kembali ke aplikasi atau menghubungkan steker di perangkat.

Untuk menggunakan alarm dengan Istirahatkan dan Aplikasi Siaga:

- Jika Anda memerlukan alarm yang terpicu saat perangkat berada dalam mode Istrahatkan atau Aplikasi Siaga tanpa menunggu masa pemeliharaan, gunakan `setAndAllowWhileIdle()` untuk alarm tidak pasti dan `setExactAndAllowWhileIdle()` untuk alarm pasti, atau setel alarm yang terlihat pengguna (API 21+).
- Beberapa alarm bisa menunggu masa pemeliharaan, atau hingga perangkat keluar dari mode Istirahatkan atau Aplikasi Siaga. Untuk alarm ini, gunakan metode `set()` dan `setExact()` standar untuk mengoptimalkan daya tahan

baterai.

Menjadwalkan alarm berulang

Anda juga bisa menggunakan `AlarmManager` untuk menjadwalkan alarm berulang, dengan menggunakan salah satu metode berikut:

- `setRepeating()` : Sebelum Android 4.4 (API Level 19), metode ini akan membuat alarm berulang dengan waktu yang pasti. Pada perangkat yang menjalankan API 19 dan yang lebih tinggi, perilaku `setRepeating()` benar-benar seperti `setInexactRepeating()` .
- `setInexactRepeating()` : Metode ini membuat alarm tidak pasti berulang yang memungkinkan batch. Bila Anda menggunakan `setInexactRepeating()` , Android akan menyinkronkan alarm berulang untuk beberapa aplikasi dan memicunya pada waktu yang sama. Hal ini mengurangi jumlah total waktu yang digunakan sistem untuk membangunkan perangkat, sehingga mengurangi konsumsi daya baterai. Seperti pada API 19, semua alarm berulang adalah alarm tidak pasti.

Untuk mengurangi kemungkinan konsumsi daya baterai:

- Jadwalkan alarm berulang menjadi sejarang mungkin.
- Gunakan pengaturan waktu tidak pasti, yang memungkinkan sistem untuk menyatukan alarm dari aplikasi yang berbeda.

Catatan: meskipun `setInexactRepeating()` merupakan perbaikan atas `setRepeating()` , tetap saja bisa membebani server jika setiap instance aplikasi menghubungkan ke server pada waktu yang kurang lebih sama. Oleh karena itu, untuk permintaan jaringan, tambahkan beberapa keacakan pada alarm Anda, seperti yang dijelaskan dalam Praktik terbaik alarm.

Jika Anda benar-benar memerlukan alarm pasti yang berulang pada API 19+, setel alarm sekali-pakai dengan `setExact()` dan setel alarm berikutnya setelah alarm itu dipicu. Alarm kedua ini disetel oleh komponen apa pun yang menerima `PendingIntent` —biasanya layanan atau penerima siaran.

Inilah contoh penggunaan `setInexactRepeating()` untuk menjadwalkan alarm berulang:

```
alarmMgr.setInexactRepeating(AlarmManager.RTC_WAKEUP,
               calendar.getTimeInMillis(),
               AlarmManager.INTERVAL_FIFTEEN_MINUTES,
               alarmIntent);
```

Dalam contoh ini:

- Dalam hal ini, `type` adalah `RTC_WAKEUP` , berarti alarm ini adalah alarm berbasis jam yang membangunkan perangkat bila alarm telah dikirim.
- Kekerapan alarm pertama dikirim segera, karena `calendar.getTimeInMillis()` mengembalikan waktu saat ini sebagai UTC milidetik.
- Setelah kekerapan pertama, alarm dikirim kurang lebih setiap 15 menit.

  Jika metodenya adalah `setRepeating()` sebagai ganti `setInexactRepeating()` , dan jika perangkat menjalankan versi API yang lebih rendah dari 19, alarm *pasti* dikirim setiap 15 menit.

  Nilai-nilai yang memungkinkan untuk argumen ini adalah `INTERVAL_DAY` , `INTERVAL_FIFTEEN_MINUTES` , `INTERVAL_HALF_DAY` , `INTERVAL_HALF_HOUR` , `INTERVAL_HOUR` .

- `alarmIntent` adalah `PendingIntent` yang berisi aksi untuk dijalankan bila alarm telah dikirim. Maksud ini biasanya berasal dari `IntentSender.getBroadcast()` .

Memeriksa alarm yang ada

Sering kali ada gunanya memeriksa apakah alarm sudah disetel. Misalnya, Anda mungkin ingin menonaktifkan kemampuan untuk menyetel alarm lain jika sudah ada alarm yang disetel.

Untuk memeriksa alarm yang ada:

1. Buat `PendingIntent` yang berisi `Intent` serupa yang digunakan untuk menyetel alarm, namun kali ini menggunakan flag `FLAG_NO_CREATE` .

   Dengan `FLAG_NO_CREATE` , `PendingIntent` hanya dibuat jika sudah ada yang berisi `Intent` yang sama. Jika tidak maka permintaan akan mengembalikan `nol` .

2. Periksa apakah `PendingIntent` adalah `null` :
   - Jika ini `null` , berarti alarm belum disetel.
   - Jika bukan `null` , maka `PendingIntent` sudah ada, berarti alarm telah disetel.

Misalnya, kode berikut akan mengembalikan `true` jika alarm yang dimuat dalam `alarmIntent` sudah ada:

```
boolean alarmExists =
 (PendingIntent.getBroadcast(this, 0,
       alarmIntent,
       PendingIntent.FLAG_NO_CREATE) != null);
```

Membatalkan alarm

Untuk membatalkan alarm, gunakan `cancel()` dan teruskan di `PendingIntent` . Misalnya:

```
alarmManager.cancel(alarmIntent);
```

Alarm yang terlihat pengguna ("jam alarm")

Untuk perangkat API 21+, Anda bisa menyetel jam alarm yang terlihat pengguna dengan memanggil `setAlarmClock()` . Aplikasi bisa mengambil jam alarm yang terlihat pengguna berikutnya yang disetel agar terpicu dengan memanggil `getNextAlarmClock()` .

Jam alarm yang disetel dengan `setAlarmClock()` akan berfungsi, bahkan bila perangkat atau aplikasi dalam keadaan diam (serupa dengan `setExactAndAllowWhileIdle()`), yang membuat Anda sedekat mungkin dengan panggilan membangunkan yang pasti.

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Alarm Manager

Ketahui selengkapnya

- Panduan Menjadwalkan Alarm Berulang
- Referensi AlarmManager
- Entri Blog Memilih Alarm
- Menjadwalkan Presentasi Alarm

8.3: Mentransfer Data Secara Efisien

Materi:

- Keadaan radio nirkabel
- Membundel transfer jaringan
- Pemuatan dini
- Memantau keadaan konektivitas
- Memantau keadaan baterai
- JobScheduler
- Praktik terkait
- Ketahui selengkapnya

Mentransfer data merupakan bagian penting dari sebagian besar aplikasi Android, namun hal ini bisa berpengaruh negatif pada daya tahan baterai dan meningkatkan biaya penggunaan data. Penggunaan radio nirkabel untuk mentransfer data berpotensi menjadi salah satu sumber paling signifikan penguras baterai aplikasi Anda.

Pengguna peduli terhadap konsumsi daya baterai karena lebih suka menggunakan perangkat seluler tanpa menghubungkannya ke pengisi daya. Pengguna juga peduli terhadap penggunaan data, karena setiap bit data yang ditransfer bisa menghabiskan biaya.

Dalam bab ini, Anda mempelajari cara aktivitas jaringan di aplikasi memengaruhi perangkat keras radio di perangkat sehingga Anda bisa meminimalkan konsumsi daya baterai yang terkait dengan aktivitas jaringan. Anda juga akan mempelajari cara menunggu kondisi yang tepat untuk menyelesaikan tugas yang banyak menggunakan sumber daya.

Keadaan radio nirkabel

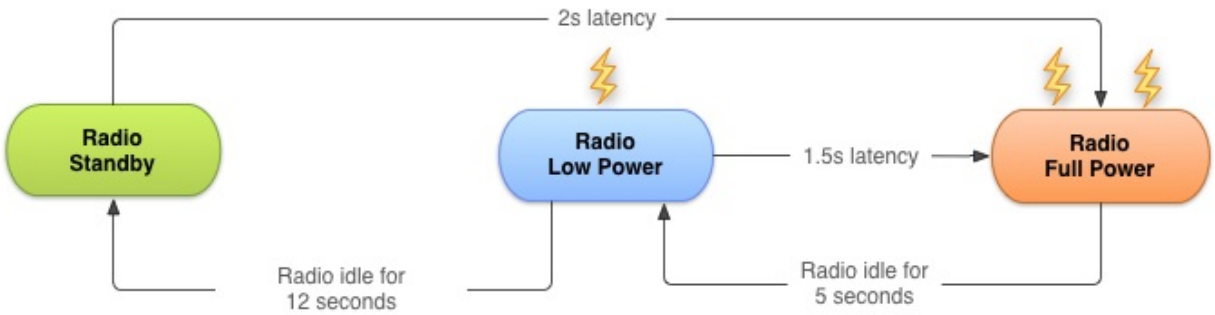
Radio nirkabel yang sepenuhnya aktif mengonsumsi daya yang signifikan. Untuk menghemat daya saat tidak digunakan, radio akan beralih di antara beberapa keadaan energi. Akan tetapi, ada konsekuensi antara penghematan daya dan waktu yang dibutuhkan untuk menyalakan bila diperlukan.

Untuk jaringan 3G pada umumnya, radio memiliki tiga keadaan energi:

1. **Daya penuh:** Digunakan bila koneksi aktif. Memungkinkan perangkat mentransfer data dengan kecepatan tertinggi.
2. **Daya rendah:** Status menengah yang menggunakan baterai kurang dari 50%.
3. **Siaga:** Keadaan energi minimum, selama tidak ada koneksi jaringan yang aktif atau dibutuhkan.

Walaupun keadaan rendah dan siaga menggunakan baterai lebih sedikit, keadaan tersebut juga menimbulkan latensi pada permintaan jaringan. Kembali ke daya penuh dari keadaan rendah memerlukan waktu sekitar 1,5 detik, sedangkan beralih dari siaga ke penuh bisa memerlukan waktu lebih dari 2 detik.

Android menggunakan mesin keadaan untuk menentukan cara transisi antar keadaan. Untuk meminimalkan latensi, mesin keadaan akan menunggu sebentar sebelum transisi ke keadaan energi yang lebih rendah.



Mesin keadaan radio pada setiap perangkat, khususnya penundaan transisi terkait ("waktu ekor") dan latensi startup, bervariasi berdasarkan teknologi radio nirkabel yang digunakan (2G, 3G, LTE, dll.) dan didefinisikan serta dikonfigurasi oleh jaringan operator yang digunakan perangkat untuk beroperasi.

Bab ini menjelaskan mesin keadaan representatif untuk radio nirkabel 3G pada umumnya, berdasarkan data yang disediakan oleh AT&T. Akan tetapi, prinsip umum dan praktik terbaik yang dihasilkan berlaku untuk semua implementasi radio nirkabel.

Seperti halnya praktik terbaik, ada konsekuensi yang perlu dipertimbangkan untuk development aplikasi Anda sendiri.

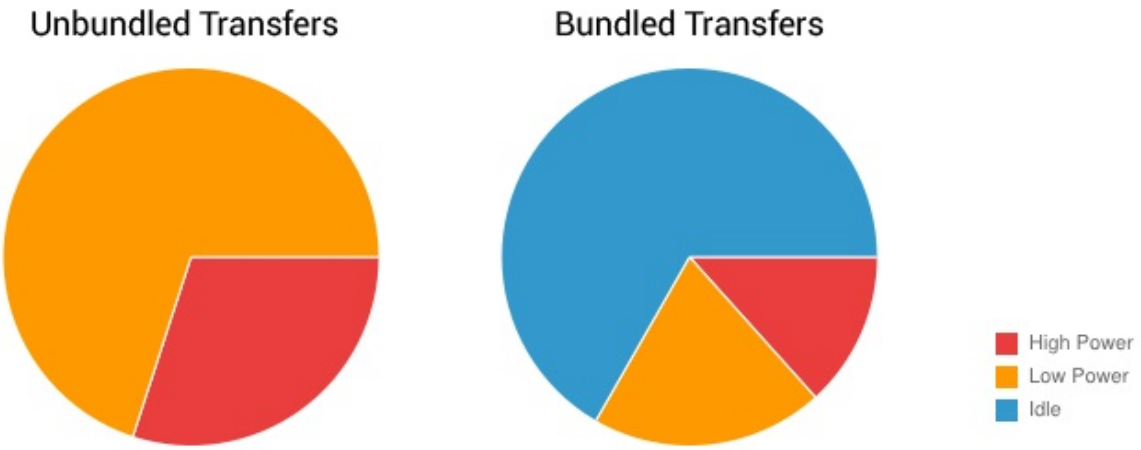
Membundel transfer jaringan

Setiap kali Anda membuat koneksi jaringan baru, radio akan bertransisi ke keadaan daya penuh. Dalam kasus mesin keadaan radio 3G yang dijelaskan di atas, keadaan akan tetap pada daya penuh selama transfer, diikuti dengan 5 detik waktu ekor, diikuti dengan 12 detik pada keadaan energi rendah sebelum dimatikan. Jadi, untuk perangkat 3G umum, setiap sesi transfer data menyebabkan radio mengonsumsi daya selama hampir 20 detik.

Maksudnya dalam praktik:

- Aplikasi yang mentransfer data yang tidak dibundel selama 1 detik setiap 18 detik akan membuat radio nirkabel selalu aktif.
- Sebagai perbandingan, aplikasi serupa yang membundel transfer selama 3 detik setiap menit akan membuat radio dalam keadaan daya tinggi hanya selama 8 detik, dan dalam keadaan daya rendah selama 12 detik lagi.

Contoh kedua memungkinkan radio dalam keadaan diam selama 40 detik setiap menitnya, yang menghasilkan pengurangan besar dalam konsumsi daya baterai.



Membundel dan mengantre transfer data Anda adalah hal penting. Anda bisa membundel transfer yang dijadwalkan terjadi dalam suatu jangka waktu dan membuat semuanya terjadi secara bersamaan, sehingga memastikan radio mengonsumsi daya dalam waktu sesedikit mungkin.

Pemuatan dini

Pemuatan dini untuk data berarti aplikasi Anda akan memperkirakan isi atau data berikutnya yang diinginkan pengguna, dan memuatnya lebih dini. Misalnya, bila pengguna memperhatikan bagian pertama artikel, perkiraan yang baik adalah memuat dini bagian berikutnya. Atau, jika pengguna sedang menonton video, memuat dini menit video berikutnya juga merupakan perkiraan yang baik.

Pemuatan dini atas data adalah cara efektif untuk mengurangi jumlah sesi transfer data independen. Pemuatan dini memungkinkan Anda mengunduh semua data yang mungkin diperlukan untuk jangka waktu tertentu dalam rentetan tunggal, melalui koneksi tunggal, pada kapasitas penuh. Hal ini mengurangi jumlah aktivasi radio yang diperlukan untuk

mengunduh data. Akibatnya, Anda tidak hanya menghemat daya tahan baterai, melainkan juga meningkatkan latensi bagi pengguna, mengurangi bandwidth yang diperlukan, dan mengurangi waktu pengunduhan.

Pemuatan dini memiliki konsekuensi. Jika mengunduh terlalu banyak atau data yang salah, Anda bisa menambah konsumsi daya baterai. Dan jika mengunduh pada waktu yang salah, bisa membuat pengguna menunggu. Mengoptimalkan data pemuatan dini merupakan topik tingkat lanjutan yang tidak dibahas dalam kursus ini, namun panduan berikut akan membahas situasi umum.

Seberapa agresif pemuatan dini akan bergantung pada ukuran data yang diunduh dan kemungkinan akan digunakannya. Sebagai panduan kasar, berdasarkan pada mesin keadaan yang dijelaskan di atas, untuk data yang memiliki peluang akan digunakan sebesar 50% dalam sesi pengguna saat ini, Anda biasanya bisa memuat dini selama sekitar 6 detik (sekitar 1-2 Mb) sebelum potensi biaya mengunduh data yang tidak digunakan cocok dengan potensi penghematan dari tidak memulai pengunduhan data itu.

Secara umum, inilah praktik yang baik untuk memuat dini data sehingga Anda hanya perlu memulai pengunduhan lain setiap 2 hingga 5 menit, dan secara berurutan 1 hingga 5 megabyte.

Dengan mengikuti prinsip ini, pengunduhan besar—seperti file video—harus diunduh berupa potongan kecil dengan interval teratur (setiap 2 hingga 5 menit), sehingga pemuatan dini secara efektif hanya untuk data video yang mungkin akan ditampilkan dalam beberapa menit berikutnya.

Contoh pemuatan dini

Banyak aplikasi berita yang berupaya mengurangi bandwidth dengan mengunduh berita utama hanya setelah kategori dipilih, artikel lengkap hanya bila pengguna ingin membacanya, dan gambar kecil sama seperti mereka menggulir ke tampilan.

Dengan menggunakan pendekatan ini, radio dipaksa untuk tetap aktif bagi sebagian besar sesi pembacaan berita saat pengguna menggulir berita utama, mengubah kategori, dan membaca artikel. Tidak hanya itu, namun peralihan terus-menerus di antara keadaan energi mengakibatkan latensi yang signifikan saat beralih kategori atau membaca artikel.

Inilah pendekatan yang lebih baik:

1. Memuat dini sejumlah data yang wajar pada startup, dimulai dengan serangkaian pertama judul berita dan gambar kecil. Proses ini memastikan waktu startup yang cepat.
2. Lanjutkan dengan judul berita selebihnya, gambar kecil selebihnya, dan teks artikel untuk setiap artikel dari rangkaian judul berita pertama.

Memantau keadaan konektivitas

Perangkat bisa mengakses jaringan menggunakan tipe perangkat keras yang berbeda:

- *Radio nirkabel* menggunakan beragam jumlah daya baterai yang bergantung pada teknologi, dan semakin besar bandwidth semakin banyak konsumsi energi. Bandwidth lebih tinggi berarti Anda bisa memuat dini secara lebih agresif, sehingga mengunduh data lebih banyak dalam waktu yang sama. Akan tetapi, mungkin kurang intuitif, karena biaya baterai waktu ekor relatif lebih tinggi, juga lebih efisien untuk membuat radio tetap aktif dalam jangka waktu yang lebih lama untuk setiap sesi transfer guna mengurangi frekuensi pembaruan.
- *Radio WiFi* menggunakan daya baterai jauh lebih sedikit daripada nirkabel dan menawarkan bandwidth yang lebih besar.

Bila memungkinkan, lakukan transfer data saat terhubung melalui Wi-Fi.

Anda bisa menggunakan `ConnectivityManager` untuk menentukan radio nirkabel yang aktif dan memodifikasi rutinitas pemuatan dini dengan bergantung pada tipe jaringan:

```
ConnectivityManager cm =
   (ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);
TelephonyManager tm =
    (TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);

NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
int PrefetchCacheSize = DEFAULT_PREFETCH_CACHE;

switch (activeNetwork.getType()) {
    case (ConnectivityManager.TYPE_WIFI):
        PrefetchCacheSize = MAX_PREFETCH_CACHE; break;
    case (ConnectivityManager.TYPE_MOBILE): {
        switch (tm.getNetworkType()) {
           case (TelephonyManager.NETWORK_TYPE_LTE |
                  TelephonyManager.NETWORK_TYPE_HSPAP):
                PrefetchCacheSize *= 4;
                break;
            case (TelephonyManager.NETWORK_TYPE_EDGE |
                  TelephonyManager.NETWORK_TYPE_GPRS):
               PrefetchCacheSize /= 2;
               break;
            default: break;
        }
        break;
      }
  default: break;
  }
```

Sistem akan mengirimkan maksud siaran bila keadaan konektivitas berubah, sehingga Anda bisa mendengarkan perubahan ini dengan menggunakan `BroadcastReceiver`.

Memantau keadaan baterai

Untuk meminimalkan konsumsi daya baterai, pantau keadaan baterai dan tunggu kondisi tertentu sebelum memulai operasi yang banyak menggunakan daya baterai.

Dalam hal ini, `BatteryManager` akan menyiarkan semua detail baterai dan pengisian daya dalam `Intent` siaran yang menyertakan status pengisian daya.

Untuk memeriksa status baterai saat ini, ujilah maksud siaran:

```
IntentFilter ifilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
Intent batteryStatus = context.registerReceiver(null, ifilter);
// Are we charging / charged?
int status = batteryStatus.getIntExtra(BatteryManager.EXTRA_STATUS, -1);
boolean isCharging = status == BatteryManager.BATTERY_STATUS_CHARGING ||
                     status == BatteryManager.BATTERY_STATUS_FULL;

// How are we charging?
int chargePlug = batteryStatus.getIntExtra(BatteryManager.EXTRA_PLUGGED, -1);
boolean usbCharge = chargePlug == BatteryManager.BATTERY_PLUGGED_USB;
boolean acCharge = chargePlug == BatteryManager.BATTERY_PLUGGED_AC;
```

Jika Anda ingin bereaksi terhadap perubahan keadaan pengisian baterai, gunakan `BroadcastReceiver` yang didaftarkan untuk tindakan status baterai:

```
<receiver android:name=".PowerConnectionReceiver">
  <intent-filter>
    <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
    <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
  </intent-filter>
</receiver>
```

Maksud siaran juga dikirim bila level baterai berubah dengan cara signifikan:

```
"android.intent.action.BATTERY_LOW"
"android.intent.action.BATTERY_OKAY"
```

JobScheduler

Terus-menerus memantau konektivitas dan status baterai perangkat bisa menjadi suatu tantangan, dan itu mengharuskan penggunaan komponen seperti penerima siaran, yang bisa mengonsumsi sumber daya sistem bahkan bila aplikasi sedang tidak dijalankan. Karena mentransfer data secara efisien merupakan tugas umum, Android SDK menyediakan kelas yang memudahkannya: `JobScheduler`.

Diperkenalkan dalam API level 21, `JobScheduler` memungkinkan Anda menjadwalkan tugas dengan ketentuan tertentu (bukan waktu tertentu seperti pada `AlarmManager`).

`JobScheduler` memiliki tiga komponen:

1. `JobInfo` menggunakan pola builder untuk menyetel ketentuan tugas.
2. `JobService` adalah wrapper kelas `Service`, tempat tugas benar-benar diselesaikan.
3. `JobScheduler` menjadwalkan dan membatalkan tugas.
   Catatan: `JobScheduler` hanya tersedia mulai API 21+. Tidak ada versi kompabilitas mundur untuk sebelum API rilis. Jika aplikasi menargetkan perangkat dengan API level sebelumnya, Anda mungkin akan merasakan manfaat FirebaseJobDispatcher alternatif.

1. JobInfo

Setel ketentuan tugas dengan membuat objek `JobInfo` menggunakan kelas `JobInfo.Builder`. Kelas `JobInfo.Builder` adalah instance yang dibuat dari konstruktor yang membutuhkan dua argumen: ID tugas (yang bisa digunakan untuk membatalkan tugas), dan `ComponentName` `JobService` yang berisi tugas. `JobInfo.Builder` harus menyetel setidaknya satu, ketentuan non-default untuk tugas. Misalnya:

```
JobScheduler scheduler = (JobScheduler) getSystemService(JOB_SCHEDULER_SERVICE);
ComponentName serviceName = new ComponentName(getPackageName(),
NotificationJobService.class.getName());
JobInfo.Builder builder = new JobInfo.Builder(JOB_ID, serviceName);
builder.setRequiredNetworkType(NETWORK_TYPE_UNMETERED);
JobInfo jobInfo = builder.build();
```

Catatan: Lihat praktik terkait untuk contoh lengkap.
Kelas JobInfo.Builder memiliki banyak metode `set()` yang memungkinkan Anda untuk menentukan ketentuan tugas. Di bawah ini adalah daftar batasan yang tersedia bersama masing-masing metode `set()` dan kontanta kelasnya:

- **Kebijakan Backoff/Coba Ulang:** Menentukan waktu dan cara menjadwalkan ulang tugas jika gagal. Setel ketentuan ini menggunakan metode `setBackoffCriteria()`, yang menggunakan dua argumen: waktu awal untuk menunggu setelah tugas gagal, dan strategi backoff. Argumen strategi backoff bisa berupa satu dari dua konstanta: `BACKOFF_POLICY_LINEAR` atau `BACKOFF_POLICY_EXPONENTIAL`. Default-nya adalah {30 seconds, Exponential}.
- **Latensi Minimum:** Jumlah waktu tunggu minimum sebelum menyelesaikan tugas. Setel ketentuan ini menggunakan metode `setMinimumLatency()` yang memerlukan argumen tunggal: jumlah waktu tunggu dalam milidetik.
- **Ganti Batas Waktu:** Waktu tunggu maksimum sebelum menjalankan tugas, bahkan jika ketentuan lain tidak terpenuhi. Setel ketentuan ini menggunakan metode `setOverrideDeadline()` yang merupakan waktu tunggu maksimum dalam milidetik.
- **Periodik:** Mengulangi tugas setelah jumlah waktu tertentu. Setel ketentuan ini menggunakan metode `setPeriodic()` dengan meneruskan interval pengulangan. Ketentuan ini saling eksklusif dengan latensi minimum dan menggantikan ketentuan batas waktu: menyetel `setPeriodic()` dengan salah satu ketentuan ini akan mengakibatkan kesalahan.
- **Bertahan:** Menyetel apakah tugas dipertahankan saat boot ulang sistem. Agar ketentuan ini bekerja, aplikasi Anda harus memiliki izin `RECEIVE_BOOT_COMPLETED`. Setel ketentuan ini menggunakan metode `setPersisted()` dengan

meneruskan boolean yang menunjukkan apakah akan mempertahankan tugas atau tidak.

- **Tipe Jaringan yang Diperlukan:** Tipe jaringan yang diperlukan tugas Anda. Jika jaringan tidak diperlukan, Anda tidak perlu memanggil fungsi ini, karena default-nya adalah `NETWORK_TYPE_NONE`. Setel ketentuan ini menggunakan metode `setRequiredNetworkType()` dengan meneruskan salah satu konstanta berikut: `NETWORK_TYPE_NONE`, `NETWORK_TYPE_ANY`, `NETWORK_TYPE_NOT_ROAMING`, `NETWORK_TYPE_UNMETERED`.
- **Keadaan Pengisian Daya yang Diperlukan:** Apakah steker perangkat perlu dihubungkan atau tidak untuk menjalankan tugas ini. Setel ketentuan ini menggunakan metode `setRequiresCharging()` dengan meneruskan boolean. Default-nya adalah `false`.
- **Mengharuskan Perangkat Diam:** Apakah perangkat harus dalam mode diam untuk menjalankan tugas ini. "Mode diam" berarti perangkat tidak digunakan dan belum digunakan selama beberapa waktu, seperti yang didefinisikan secara bebas oleh sistem. Bila perangkat dalam mode diam, inilah waktu yang tepat untuk menjalankan tugas yang banyak menggunakan sumber daya. Setel ketentuan ini menggunakan metode `setRequiresDeviceIdle()` dengan meneruskan boolean. Default-nya adalah `false`.

2. JobService

Setelah ketentuan tugas terpenuhi, kerangka kerja akan meluncurkan subkelas `JobService`, yang merupakan tempat Anda mengimplementasikan tugas itu sendiri. `JobService` berjalan pada thread UI, sehingga Anda perlu memindahkan operasi pemblokiran ke thread pekerja.

Deklarasikan subkelas `JobService` di Manifes Android, dan sertakan izin `BIND_JOB_SERVICE`:

```
<service android:name="MyJobService"
             android:permission="android.permission.BIND_JOB_SERVICE" />
```

Di subkelas `JobService` Anda, gantikan kedua metode, `onStartJob()` dan `onStopJob()`.

onStartJob()

Sistem akan memanggil `onStartJob()` dan secara otomatis meneruskan sebuah objek `JobParameters`, yang dibuat sistem dengan informasi tentang tugas Anda. Jika tugas Anda berisi operasi yang berjalan lama, pindahkan pekerjaan ke thread terpisah. Metode `onStartJob()` mengembalikan boolean: `true` jika tugas sudah dipindahkan ke thread terpisah (artinya mungkin belum selesai) dan `false` jika tidak ada tugas lain yang harus diselesaikan.

Gunakan metode `jobFinished()` dari thread apa pun untuk memberi tahu sistem bahwa tugas Anda selesai. Metode ini memerlukan dua parameter: objek `JobParameters` yang berisi informasi tentang tugas, dan boolean yang menunjukkan apakah tugas perlu dijadwal ulang, sesuai dengan kebijakan backoff yang didefinisikan.

onStopJob()

Sistem akan memanggil `onStopJob()` jika ini menentukan bahwa Anda harus menghentikan eksekusi tugas bahkan sebelum Anda memanggil `jobFinished()`. Ini terjadi jika persyaratan yang Anda tetapkan saat menjadwalkan tugas tidak lagi dipenuhi.

Contoh:

- Jika Anda meminta WiFi dengan `setRequiredNetworkType()` namun pengguna menonaktifkan WiFi sewaktu tugas dieksekusi, maka sistem akan memanggil `onStopJob()`.
- Jika Anda menetapkan `setRequiresDeviceIdle()` namun pengguna mulai berinteraksi dengan perangkat sewaktu tugas dieksekusi, maka sistem akan memanggil `onStopJob()`.

Anda bertanggung jawab atas perilaku aplikasi saat menerima `onStopJob()`, jadi jangan abaikan. Metode ini mengembalikan boolean, yang menunjukkan apakah Anda ingin menjadwal ulang tugas berdasarkan kebijakan backoff yang didefinisikan, atau melepaskan tugas.

3. JobScheduler

Bagian akhir penjadwalan tugas adalah menggunakan kelas `JobScheduler` untuk menjadwalkan tugas. Untuk mendapatkan instance kelas ini, panggil `getSystemService(JOB_SCHEDULER_SERVICE)`. Kemudian jadwalkan tugas menggunakan metode `schedule()`, dengan meneruskan objek `JobInfo` yang Anda buat dengan `JobInfo.Builder`. Misalnya:

```
mScheduler.schedule(myJobInfo);
```

Kerangka kerja tahu tentang kapan Anda menerima callback, dan mencoba untuk menggabung dan menangguhkannya sebanyak mungkin. Biasanya, jika Anda tidak menetapkan batas waktu tugas, sistem bisa menjalankannya kapan saja, bergantung pada keadaan saat ini dari antrean internal objek `JobScheduler`; akan tetapi, mungkin akan ditangguhkan hingga saat berikutnya perangkat dihubungkan ke sumber daya.

Untuk membatalkan tugas, panggil `cancel()`, dengan meneruskan ID tugas dari objek `JobInfo.Builder`, atau panggil `cancelAll()`. Misalnya:

```
mScheduler.cancelAll();
```

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- JobScheduler

Ketahui selengkapnya

- Panduan Mentransfer Data Tanpa Menguras Baterai
- Panduan Mengoptimalkan Pengunduhan Agar Akses Jaringan Efisien
- Panduan Memodifikasi Pola Pengunduhan Anda Berdasarkan Tipe Konektivitas
- Referensi JobScheduler
- Referensi JobService
- Referensi JobInfo
- Referensi JobInfo.Builder
- Referensi JobParameters
- Presentasi tentang Menjadwalkan Tugas
- Apa perbedaan jaringan dan kecepatan?

9.0: Menyimpan Data

Materi:

- Preferensi bersama
- File
- Database SQLite
- Opsi storage lain
- Koneksi jaringan
- Mencadangkan data aplikasi
- Firebase
- Ketahui selengkapnya

Android menyediakan sejumlah opsi bagi Anda untuk menyimpan data aplikasi yang persisten. Solusi yang dipilih bergantung pada kebutuhan khusus Anda, misalnya apakah data harus bersifat privat untuk aplikasi Anda atau bisa diakses oleh aplikasi lainnya (dan pengguna) serta berapa banyak ruang yang diperlukan data.

Opsi storage data Anda adalah seperti berikut:

- Preferensi bersama—Menyimpan data primitif pribadi dalam pasangan nilai-kunci Hal ini akan dibahas di bab berikutnya.
- Penyimpanan internal—Menyimpan data privat pada memori perangkat.
- Penyimpanan eksternal—Menyimpan data publik pada penyimpanan eksternal bersama.
- Database SQLite—Menyimpan data terstruktur dalam database privat.
- Koneksi jaringan—Menyimpan data di web dengan server jaringan sendiri.
- Pencadangan Awan—Membuat cadangan data pengguna dan aplikasi di awan.
- Penyedia materi—Menyimpan data secara privat dan membuatnya tersedia secara publik. Hal ini akan dibahas dalam bab berikutnya.
- Database realtime Firebase—Menyimpan dan menyinkronkan data dengan database awan NoSQL. Data disinkronkan dengan semua klien yang terhubung secara realtime, dan tetap tersedia bila aplikasi Anda offline.

Preferensi bersama

Menggunakan preferensi bersama adalah cara untuk membaca dan menulis pasangan nilai-kunci informasi secara persisten ke dan dari suatu file.

Catatan: Secara default, pasangan nilai-kunci ini tidak digunakan bersama, bukan pula preferensi, jadi jangan mencampuradukkannya dengan Preference API.
Preferensi Bersama dibahas dalam ">bab tersendiri.

File

Android menggunakan sistem file yang serupa dengan sistem file berbasis disk pada platform lainnya seperti Linux. Operasi berbasis disk tentunya familier bagi setiap orang yang telah menggunakan I/O file Linux atau paket java.io.

Semua perangkat Android memiliki dua area file-storage: storage “internal” dan “eksternal”. Nama-nama ini berasal dari masa awal Android, saat kebanyakan perangkat menawarkan memori bawaan non-volatil (penyimpanan internal), plus media storage lepas-pasang seperti kartu micro-SD (penyimpanan eksternal).

Saat ini, beberapa perangkat membagi ruang storage permanen menjadi partisi "internal" dan "eksternal", sehingga walaupun tanpa media storage lepas-pasang, selalu ada dua ruang storage dan perilaku API sama, ada atau tidak ada penyimpanan eksternal. Daftar berikut merangkum fakta tentang setiap ruang storage.

Penyimpanan internal	Penyimpanan eksternal
Selalu tersedia.	Tidak selalu tersedia, karena pengguna bisa memasang penyimpanan eksternal sebagai storage USB, dan dalam beberapa kasus, bisa melepasnya dari perangkat.
Hanya aplikasi Anda yang bisa mengakses file. Secara khusus, direktori penyimpanan internal aplikasi ditetapkan oleh nama paket aplikasi Anda di lokasi khusus sistem file Android. Aplikasi lain tidak bisa menjelajah direktori internal Anda dan tidak memiliki akses baca atau tulis kecuali jika Anda secara eksplisit menyetel file itu agar bisa dibaca atau bisa ditulis.	Bisa dibaca semua orang. Setiap aplikasi bisa membacanya.
Bila pengguna mencopot pemasangan aplikasi, sistem akan membuang semua file aplikasi Anda dari penyimpanan internal.	Bila pengguna mencopot pemasangan aplikasi, sistem akan membuang file aplikasi dari sini hanya jika Anda menyimpannya dalam direktori dari getExternalFilesDir().
Penyimpanan internal adalah yang terbaik bila ingin memastikan bahwa pengguna maupun aplikasi lain tidak bisa mengakses file Anda.	Penyimpanan eksternal adalah tempat terbaik untuk file yang tidak memerlukan pembatasan akses dan untuk file yang ingin Anda gunakan bersama aplikasi lain atau memungkinkan pengguna mengaksesnya dengan komputer.

Penyimpanan internal

Anda tidak memerlukan izin apa pun untuk menyimpan file pada penyimpanan internal. Aplikasi Anda selalu memiliki izin untuk membaca dan menulis file dalam direktori penyimpanan internalnya.

Anda bisa membuat file dalam dua direktori berbeda:

- Storage permanen: `getFilesDir()`
- Storage sementara: `getCacheDir()`. Disarankan untuk file kecil dan sementara dengan total kurang dari 1 MB. Perhatikan, sistem mungkin menghapus file sementara jika kehabisan memori.

Untuk membuat file baru di salah satu direktori ini, Anda bisa menggunakan konstruktor File(), meneruskan File yang disediakan oleh salah satu metode di atas yang menetapkan direktori penyimpanan internal Anda. Misalnya:

```
File file = new File(context.getFilesDir(), filename);
```

Atau, Anda bisa memanggil openFileOutput() untuk mendapatkan FileOutputStream yang menulis ke file dalam direktori internal. Misalnya, inilah cara menulis sejumlah teks ke file:

```
String filename = "myfile";
String string = "Hello world!";
FileOutputStream outputStream;

try {
  outputStream = openFileOutput(filename, Context.MODE_PRIVATE);
  outputStream.write(string.getBytes());
  outputStream.close();
} catch (Exception e) {
  e.printStackTrace();
}
```

Atau, jika Anda perlu menyimpan beberapa file ke cache, lebih baik gunakan createTempFile(). Misalnya, metode berikut akan mengekstrak nama file dari URL dan membuat file dengan nama itu dalam direktori cache internal aplikasi Anda:

```
public File getTempFile(Context context, String url) {
    File file;
    try {
        String fileName = Uri.parse(url).getLastPathSegment();
        file = File.createTempFile(fileName, null, context.getCacheDir());
    } catch (IOException e) {
        // Error while creating file
    }
    return file;
}
```

Penyimpanan eksternal

Gunakan penyimpanan eksternal untuk file yang harus disimpan secara permanen, bahkan jika aplikasi Anda dicopot pemasangannya, dan tersedia bebas untuk pengguna dan aplikasi lainnya, seperti foto, gambar, atau dokumen yang dibuat oleh aplikasi.

Beberapa file privat yang tidak memiliki nilai bagi aplikasi lain juga bisa disimpan pada penyimpanan eksternal. File semacam itu bisa berupa sumber daya aplikasi tambahan yang telah diunduh, atau file media sementara. Pastikan menghapusnya bila aplikasi Anda telah dicopot pemasangannya.

Memperoleh izin untuk penyimpanan eksternal

Untuk menulis ke penyimpanan eksternal, Anda harus meminta izin WRITE_EXTERNAL_STORAGE dalam file manifes. Hal ini secara implisit menyertakan izin untuk membaca.

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

Jika aplikasi Anda perlu membaca penyimpanan eksternal (namun bukan menulisnya), maka Anda perlu mendeklarasikan izin READ_EXTERNAL_STORAGE.

```
<manifest ...>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    ...
</manifest>
```

Selalu periksa apakah penyimpanan eksternal telah dipasang

Karena penyimpanan eksternal mungkin tidak tersedia—seperti saat pengguna telah memasang storage ke PC atau telah melepas kartu SD yang menyediakan penyimpanan eksternal—Anda harus selalu memverifikasi apakah volume tersedia sebelum mengaksesnya. Anda bisa membuat kueri status penyimpanan eksternal dengan memanggil getExternalStorageState(). Jika keadaan yang dikembalikan sama dengan MEDIA_MOUNTED, maka Anda bisa membaca dan menulis file. Misalnya, metode berikut ini berguna untuk menentukan ketersediaan storage:

```
/* Checks if external storage is available for read and write */
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

Penyimpanan eksternal privat dan publik

Penyimpanan eksternal berstruktur sangat spesifik dan digunakan oleh sistem Android. Ada beberapa direktori publik dan direktori privat yang khusus untuk aplikasi Anda. Setiap pohon file ini memiliki tiga direktori yang diidentifikasi melalui konstanta sistem.

Misalnya, setiap file yang Anda simpan ke dalam direktori nada dering publik DIRECTORY_RINGTONES akan tersedia untuk semua aplikasi nada dering lainnya.

Sebaliknya, setiap file yang Anda simpan dalam direktori nada dering privat DIRECTORY_RINGTONES, secara default, hanya bisa dilihat oleh aplikasi Anda dan akan dihapus bersama aplikasi.

Lihat daftar direktori publik untuk daftar selengkapnya.

Mendapatkan deskriptor file

Untuk mengakses direktori penyimpanan eksternal publik, dapatkan sebuah jalur dan buat file yang memanggil getExternalStoragePublicDirectory().

```
File path = Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES);
File file = new File(path, "DemoPicture.jpg");
```

Untuk mengakses direktori penyimpanan eksternal pribadi, dapatkan sebuah jalur dan file yang memanggil getExternalFilesDir().

```
File file = new File(getExternalFilesDir(null), "DemoFile.jpg");
```

Membuat kueri ruang storage

Jika sudah mengetahui jumlah data yang disimpan, Anda bisa mengetahui apakah tersedia ruang yang cukup tanpa menyebabkan IOException dengan memanggil getFreeSpace() atau getTotalSpace(). Setiap metode ini memberitahukan ruang yang tersedia saat ini dan total ruang di volume storage.

Anda tidak harus memeriksa jumlah ruang yang tersedia sebelum menyimpan file. Sebagai gantinya, Anda bisa langsung mencoba menulis file, kemudian menangkap IOException jika terjadi. Anda mungkin perlu melakukannya jika tidak mengetahui secara persis jumlah ruang yang diperlukan.

Menghapus file

Anda harus selalu menghapus file yang tidak lagi diperlukan. Cara paling langsung untuk menghapus file adalah membuat referensi file yang telah dibuka memanggil delete()pada dirinya sendiri.

```
myFile.delete();
```

Jika file disimpan pada penyimpanan internal, Anda juga bisa meminta Context untuk menemukan dan menghapus file dengan memanggil deleteFile():

```
myContext.deleteFile(fileName);
```

Sebagai warga yang baik, Anda juga seharusnya secara teratur menghapus file cache yang dibuat dengan getCacheDir().

Berinteraksi dengan rangkuman file

Setelah Anda memiliki deskriptor file, gunakan operator atau aliran file java.io standar untuk berinteraksi dengan file. Hal ini tidak khusus untuk Android dan tidak dibahas di sini.

Database SQLite

Menyimpan data ke database cocok untuk data terstruktur atau berulang, misalnya informasi kontak. Android menyediakan database seperti-SQL untuk keperluan ini.

Bab dan praktik berikut akan mengajarkan secara mendalam cara menggunakan database SQLite bersama aplikasi Android Anda:

- SQLite Primer
- Pengantar Database SQLite
- Praktik Storage Data SQLite
- Menelusuri Praktik Database SQLite

Opsi storage lain

Android menyediakan opsi storage tambahan di luar cakupan kursus pengantar ini. Jika Anda ingin mendalaminya, lihatlah sumber daya di bawah ini.

Koneksi jaringan

Anda bisa menggunakan jaringan (bila tersedia) untuk menyimpan dan mengambil data pada layanan berbasis web sendiri. Untuk melakukan operasi jaringan, gunakan kelas-kelas dalam paket berikut:

- java.net.*
- android.net.*

Mencadangkan data aplikasi

Pengguna seringkali menghabiskan banyak waktu serta tenaga untuk membuat data dan menyetel preferensi dalam aplikasi. Mempertahankan data itu untuk pengguna jika mereka mengganti perangkat yang rusak atau memutakhirkan ke perangkat baru merupakan bagian penting untuk memastikan pengalaman pengguna yang menyenangkan.

Auto Backup untuk Android 6.0 (API level 23) dan yang lebih tinggi

Untuk aplikasi dengan versi SDK target berupa Android 6.0 (API level 23) dan yang lebih tinggi, perangkat yang menjalankan Android 6.0 dan yang lebih tinggi secara otomatis membuat cadangan data aplikasi ke awan. Sistem melakukan pencadangan otomatis ini bagi hampir semua data aplikasi secara default, dan melakukannya tanpa harus menulis kode aplikasi tambahan.

Bila pengguna memasang aplikasi pada perangkat baru, atau memasang ulang aplikasi di satu perangkat (misalnya setelah dikembalikan ke setelan pabrik), sistem secara otomatis memulihkan data aplikasi dari awan. Fitur pencadangan otomatis mempertahankan data aplikasi yang dibuat pada perangkat pengguna dengan mengunggahnya ke akun Google Drive pengguna dan mengenkripsinya. Anda atau pengguna tidak akan dikenakan biaya storage data, dan data yang disimpan tidak dihitung terhadap kuota Google Drive pribadi pengguna. Setiap aplikasi bisa menyimpan hingga 25 MB. Setelah data yang dicadangkan mencapai 25 MB, aplikasi tidak akan lagi mengirim data ke awan. Jika melakukan pemulihan data, sistem akan menggunakan cuplikan data terakhir yang dikirim aplikasi ke awan.

Pencadangan otomatis terjadi bila ketentuan berikut terpenuhi:

- Perangkat sedang tidak digunakan.
- Perangkat sedang diisi dayanya.
- Perangkat terhubung ke jaringan Wi-Fi.
- Paling tidak 24 jam telah berlalu sejak pencadangan terakhir.

Anda bisa menyesuaikan dan mengonfigurasi pencadangan otomatis untuk aplikasi. Lihat Mengonfigurasi Auto Backup for Apps.

Pencadangan untuk Android 5.1 (API level 22) dan yang lebih rendah

Untuk pengguna Android versi sebelumnya, Anda perlu menggunakan Backup API untuk mengimplementasikan pencadangan data. Singkatnya, hal ini mengharuskan Anda:

1. Mendaftar Android Backup Service untuk mendapatkan Backup Service Key.
2. Mengonfigurasi Manifes untuk menggunakan Backup Service.
3. Membuat agen pencadangan dengan memperluas kelas BackupAgentHelper.
4. Meminta pencadangan bila data berubah.

Informasi selengkapnya dan kode contoh:

- Menggunakan Backup API
- Pencadangan Data

Firebase

Firebase adalah platform seluler yang membantu Anda mengembangkan aplikasi, menumbuhkan basis pengguna, dan menghasilkan uang lebih banyak. Firebase terdiri dari beberapa fitur pelengkap yang bisa dipadupadankan sesuai dengan kebutuhan Anda.

Beberapa fitur tersebut adalah Analytics, Perpesanan Awan, Notifikasi, dan Test Lab.

Untuk pengelolaan data, Firebase menawarkan Realtime Database.

- Simpan dan sinkronkan data dengan database awan NoSQL.
- Aplikasi yang terhubung akan berbagi data
- Ditampung di awan
- Data disimpan sebagai JSON
- Data disimpan sebagai JSON dan disinkronkan secara real-time dengan setiap klien yang terhubung.
- Data tetap tersedia saat aplikasi Anda sedang offline

Lihat beranda Firebase untuk informasi selengkapnya.

Ketahui selengkapnya

File

- Menyimpan File
- Dokumentasi dan contoh kode getExternalFilesDir()
- Dokumentasi dan contoh kode getExternalStoragePublicDirectory()
- Kelas java.io.File
- Tutorial I/O Oracle Java

Pencadangan

- Mengonfigurasi Auto Backup for Apps
- Menggunakan Backup API
- Pencadangan Data

Preferensi Bersama

- Menyimpan Rangkaian Nilai-Kunci
- Panduan Menggunakan Preferensi Bersama
- Referensi Preferensi Bersama

Firebase

- Beranda Firebase
- Firebase Realtime Database
- Tambahkan Firebase ke Proyek Android Anda

9.1: Preferensi Bersama

Materi:

- Preferensi bersama vs. keadaan instance tersimpan
- Membuat preferensi bersama file
- Menyimpan preferensi bersama
- Memulihkan preferensi bersama
- Mengosongkan preferensi bersama
- Mendengarkan perubahan preferensi
- Praktik terkait
- Ketahui selengkapnya

Preferensi bersama memungkinkan Anda membaca dan menulis data primitif dalam jumlah kecil sebagai pasangan kunci/nilai ke file pada storage perangkat. Kelas SharedPreference menyediakan API untuk mendapatkan handle ke file preferensi serta untuk membaca, menulis, dan mengelola data ini. File preferensi bersama ini sendiri dikelola oleh kerangka kerja dan bisa diakses (digunakan bersama) oleh semua komponen aplikasi Anda. Data tersebut tidak digunakan bersama atau bisa diakses oleh aplikasi lainnya.

Untuk mengelola data dalam jumlah besar, gunakan database SQLite atau opsi storage lainnya yang cocok, yang akan dibahas dalam bab berikutnya.

Preferensi bersama vs. keadaan instance tersimpan

Dalam bab sebelumnya, Anda telah mempelajari tentang mempertahankan keadaan dengan menggunakan keadaan instance tersimpan. Inilah perbandingan di antara keduanya.

Preferensi Bersama	**Keadaan instance tersimpan**
Bertahan di semua sesi pengguna, meskipun aplikasi Anda ditutup dan dimulai ulang, atau perangkat di-boot ulang.	Mempertahankan data keadaan di seluruh instance aktivitas dalam sesi pengguna yang sama.
Data yang harus diingat di semua sesi, misalnya setelan pilihan pengguna atau skor game mereka.	Data yang tidak boleh diingat di semua sesi, misalnya tab yang dipilih, atau keadaan aktivitas saat ini.
Sejumlah kecil pasangan kunci/nilai	Sejumlah kecil pasangan kunci/nilai
Data bersifat privat untuk aplikasi.	Data bersifat privat untuk aplikasi.
Penggunaan umumnya adalah untuk menyimpan preferensi pengguna.	Kita biasanya membuat ulang keadaan setelah perangkat diputar.

Catatan: SharedPreference API juga berbeda dari Preference API. Preference API bisa digunakan untuk membangun antarmuka pengguna untuk laman setelan, dan tidak menggunakan preferensi bersama bagi implementasi dasarnya. Lihat Setelan untuk informasi selengkapnya mengenai setelan dan Preference API.

Membuat file preferensi bersama file

Anda hanya memerlukan satu file preferensi bersama untuk aplikasi, dan biasanya diberi nama dengan nama paket aplikasi. Hal ini membuat namanya unik dan mudah dikaitkan dengan aplikasi Anda.

Anda membuat file preferensi bersama dalam metode onCreate() aktivitas utama dan menyimpannya dalam sebuah variabel anggota.

```
private String sharedPrefFile = "com.example.android.hellosharedprefs";
mPreferences = getSharedPreferences(sharedPrefFile, MODE_PRIVATE);
```

Diperlukan argumen mode, karena Android versi lama memiliki mode lain yang memungkinkan Anda membuat file preferensi bersama yang bisa dibaca dan ditulis oleh umum. Mode ini tidak digunakan lagi di API 17, dan sekarang **sangat tidak dianjurkan** karena alasan keamanan. Jika Anda perlu berbagi data dengan aplikasi lainnya, gunakan layanan atau penyedia materi.

Menyimpan preferensi bersama

Anda menyimpan preferensi dalam keadaan onPause() daur hidup aktivitas dengan menggunakan antarmuka SharedPreferences.Editor.

1. Dapatkan SharedPreferences.Editor. Editor menangani semua operasi file untuk Anda. Bila dua editor memodifikasi preferensi pada waktu yang sama, maka yang terakhir dipanggil akan menang.
2. Tambahkan pasangan kunci/nilai ke editor dengan menggunakan metode put yang sesuai untuk tipe data tersebut. Metode put akan menimpa nilai yang ada sebelumnya pada kunci yang ada.
3. Panggil apply() untuk menuliskan perubahan Anda. Metode apply() menyimpan preferensi secara asinkron, di luar thread UI. Editor preferensi bersama juga memiliki metode commit() untuk menyimpan preferensi secara sinkron. Metode commit() tidak disarankan karena bisa memblokir operasi lain. Karena instance SharedPreferences adalah singleton dalam suatu proses, maka aman untuk mengganti instance commit() dengan apply() jika Anda sudah mengabaikan nilai kembalian.

   Anda tidak perlu mengkhawatirkan daur hidup komponen Android dan interaksinya dengan penulisan apply() ke disk. Kerangka kerja memastikan penulisan disk yang sedang diakses dari apply() selesai sebelum mengubah keadaan.

```
@Override
protected void onPause() {
   super.onPause();
   SharedPreferences.Editor preferencesEditor = mPreferences.edit();
   preferencesEditor.putInt("count", mCount);
   preferencesEditor.putInt("color", mCurrentColor);
   preferencesEditor.apply();
}
```

Memulihkan preferensi bersama

Anda memulihkan preferensi bersama dalam metode onCreate() aktivitas. Metode get() mengambil dua argumen— satu untuk kunci dan satu lagi untuk nilai default jika kunci tidak bisa ditemukan. Dengan menggunakan argumen default, Anda tidak perlu menguji apakah preferensi ada dalam file.

```
mPreferences = getSharedPreferences(sharedPrefFile, MODE_PRIVATE);
if (savedInstanceState != null) {
    mCount = mPreferences.getInt("count", 1);
    mShowCount.setText(String.format("%s", mCount));

    mCurrentColor = mPreferences.getInt("color", mCurrentColor);
    mShowCount.setBackgroundColor(mCurrentColor);
} else { … }
```

Mengosongkan preferensi bersama

Untuk mengosongkan semua nilai dalam file preferensi bersama, panggil metode clear() pada editor preferensi bersama dan terapkan perubahan.

```
SharedPreferences.Editor preferencesEditor = mPreferences.edit();
preferencesEditor.putInt("number", 42);
preferencesEditor.clear();
preferencesEditor.apply();
```

Anda bisa mengombinasikan panggilan untuk put dan clear. Akan tetapi, saat menerapkan preferensi, clear selalu dilakukan terlebih dahulu, tanpa menghiraukan apakah Anda memanggil metode clear sebelum atau setelah metode put pada editor ini.

Mendengarkan perubahan preferensi

Ada sejumlah alasan yang membuat Anda perlu diberi tahu segera setelah pengguna mengubah salah satu preferensi. Untuk menerima callback saat perubahan terjadi pada salah satu preferensi, implementasikan antarmuka SharedPreference.OnSharedPreferenceChangeListener dan daftarkan listener untuk objek SharedPreferences dengan memanggil registerOnSharedPreferenceChangeListener().

Antarmuka tersebut hanya memiliki satu metode callback, onSharedPreferenceChanged(), dan Anda bisa mengimplementasikan antarmuka sebagai bagian dari aktivitas.

```
public class SettingsActivity extends PreferenceActivity
                              implements OnSharedPreferenceChangeListener {
    public static final String KEY_PREF_SYNC_CONN = "pref_syncConnectionType";
    ...

    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
        String key) {
        if (key.equals(KEY_PREF_SYNC_CONN)) {
            Preference connectionPref = findPreference(key);
            // Set summary to be the user-description for the selected value
            connectionPref.setSummary(sharedPreferences.getString(key, ""));
        }
    }
}
```

Dalam contoh ini, metode akan memeriksa apakah setelan yang diubah adalah untuk kunci preferensi yang diketahui. Ini akan memanggil findPreference() untuk mendapatkan objek Preference yang diubah agar bisa memodifikasi rangkuman item menjadi keterangan pilihan pengguna.

Untuk pengelolaan daur hidup yang tepat dalam aktivitas, daftarkan dan cabut pendaftaran SharedPreferences.OnSharedPreferenceChangeListener Anda, masing-masing selama callback onResume() dan onPause():

```
@Override
protected void onResume() {
    super.onResume();
    getPreferenceScreen().getSharedPreferences()
            .registerOnSharedPreferenceChangeListener(this);
}

@Override
protected void onPause() {
    super.onPause();
    getPreferenceScreen().getSharedPreferences()
            .unregisterOnSharedPreferenceChangeListener(this);
}
```

Menahan referensi ke listener

Bila Anda memanggil registerOnSharedPreferenceChangeListener(), pengelola preferensi saat ini tidak akan langsung menyimpan referensi ke listener. Anda harus menahan referensi ke listener, atau referensi akan rentan terhadap pengumpulan sampah. Pertahankan referensi ke listener dalam data instance suatu objek yang akan ada selama Anda memerlukan listener.

```
SharedPreferences.OnSharedPreferenceChangeListener listener =
    new SharedPreferences.OnSharedPreferenceChangeListener() {
  public void onSharedPreferenceChanged(SharedPreferences prefs, String key) {
    // listener implementation
  }
};
prefs.registerOnSharedPreferenceChangeListener(listener);
```

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Preferensi Bersama

Ketahui selengkapnya

- Menyimpan Data
- Opsi Storage
- Menyimpan Rangkaian Nilai-Kunci
- SharedPreferences
- SharedPreferences.Editor

Stackoverflow

- Cara menggunakan SharedPreferences di Android untuk menyimpan, mengambil dan mengedit nilai-nilai
- onSavedInstanceState vs. SharedPreferences

9.2: Setelan Aplikasi

Materi:

- Menentukan kontrol setelan yang sesuai
- Menyediakan navigasi untuk Setelan
- UI Setelan
- Menampilkan setelan
- Menyetel nilai-nilai default untuk setelan
- Membaca nilai-nilai setelan
- Mendengarkan perubahan setelan
- Menggunakan template Settings Activity
- Praktik terkait
- Ketahui selengkapnya

Bab ini menjelaskan setelan aplikasi yang memungkinkan pengguna untuk menunjukkan preferensi mereka tentang bagaimana seharusnya aplikasi atau layanan berperilaku.

Menentukan kontrol setelan yang sesuai

Aplikasi sering kali menyertakan setelan yang memungkinkan pengguna memodifikasi fitur dan perilaku aplikasi. Misalnya, beberapa aplikasi memungkinkan pengguna menetapkan apakah notifikasi diaktifkan atau menetapkan seberapa sering aplikasi menyinkronkan data dengan awan.

Kontrol yang ada dalam setelan aplikasi seharusnya menangkap preferensi pengguna yang memengaruhi sebagian besar pengguna atau menyediakan dukungan penting bagi sebagian kecil pengguna. Misalnya, setelan notifikasi memengaruhi semua pengguna, sementara setelan mata uang untuk pasar asing menyediakan dukungan penting bagi pengguna dalam pasar itu.

Setelan biasanya tidak sering diakses, karena setelah pengguna mengubah setelan, mereka jarang kembali dan mengubahnya lagi. Jika kontrol atau preferensi yang Anda sediakan untuk pengguna adalah sesuatu yang perlu sering diakses, pertimbangkan untuk memindahkannya ke menu opsi bilah aplikasi, atau ke menu navigasi samping seperti panel samping navigasi.

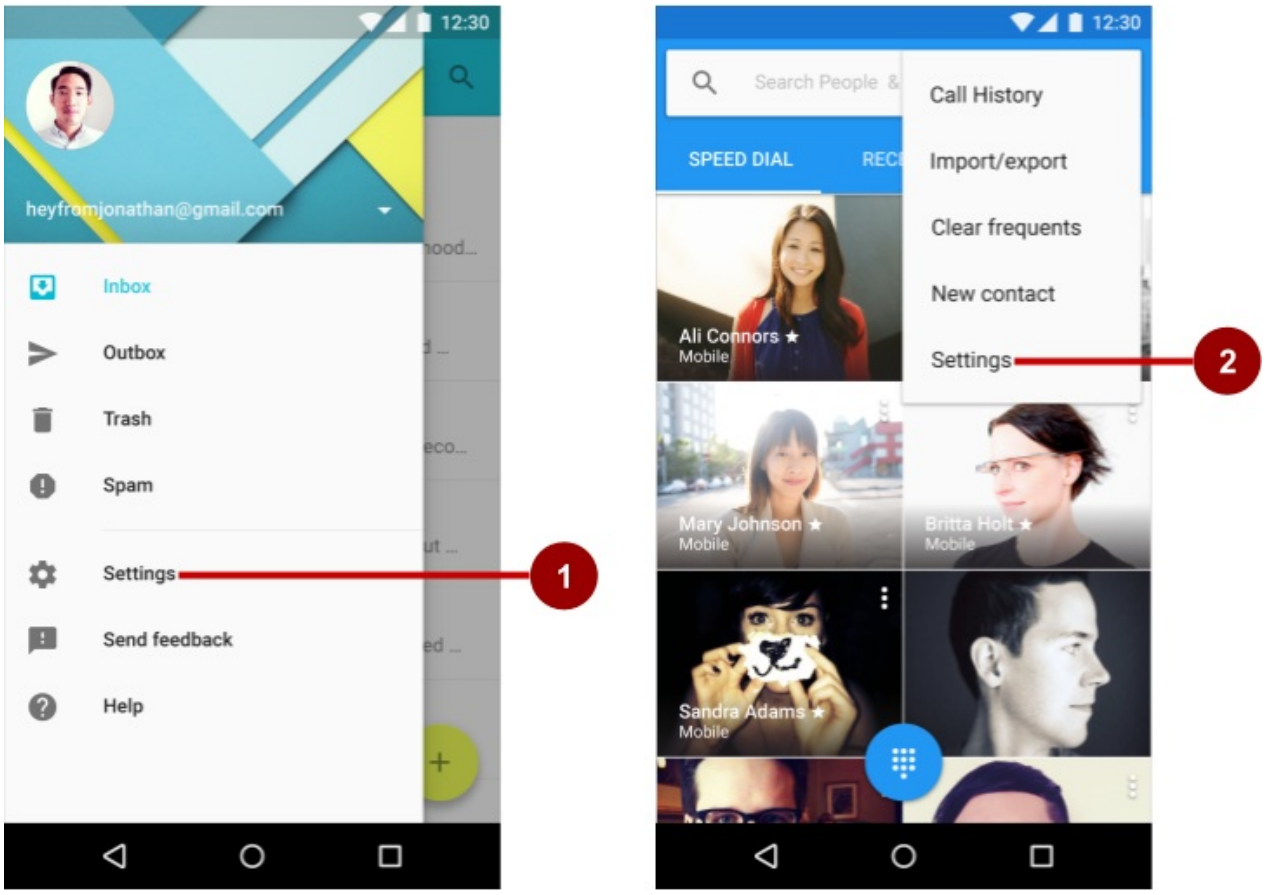
Setel default untuk kontrol setelan Anda yang familier bagi pengguna dan perbaiki pengalaman aplikasi. Nilai default awal untuk setelan harus:

- Menyatakan nilai yang akan dipilih kebanyakan pengguna, seperti **Semua kontak** untuk "Kontak yang akan ditampilkan" dalam aplikasi Kontak.
- Menggunakan daya baterai lebih sedikit. Misalnya, dalam aplikasi Android Settings, Bluetooth dinonaktifkan hingga pengguna mengaktifkannya.
- Menimbulkan risiko terkecil untuk keamanan dan kehilangan data. Misalnya, setelan default untuk aksi default aplikasi Gmail adalah mengarsipkan, bukan menghapus.
- Melakukan interupsi bila penting saja. Misalnya, setelan default bila panggilan dan notifikasi masuk adalah hanya menyela bila penting.

Tip: Jika setelan berisi informasi tentang aplikasi, seperti nomor versi atau informasi lisensi, pindahkan setelan ini ke layar Bantuan yang diakses terpisah.

Menyediakan navigasi ke Setelan

Pengguna harus bisa masuk ke setelan aplikasi dengan mengetuk **Settings** di navigasi samping, misalnya panel samping navigasi, seperti yang ditampilkan yang di samping kiri gambar di bawah ini, atau di menu opsi di bilah aplikasi, seperti yang ditampilkan di sebelah kanan gambar di bawah ini.



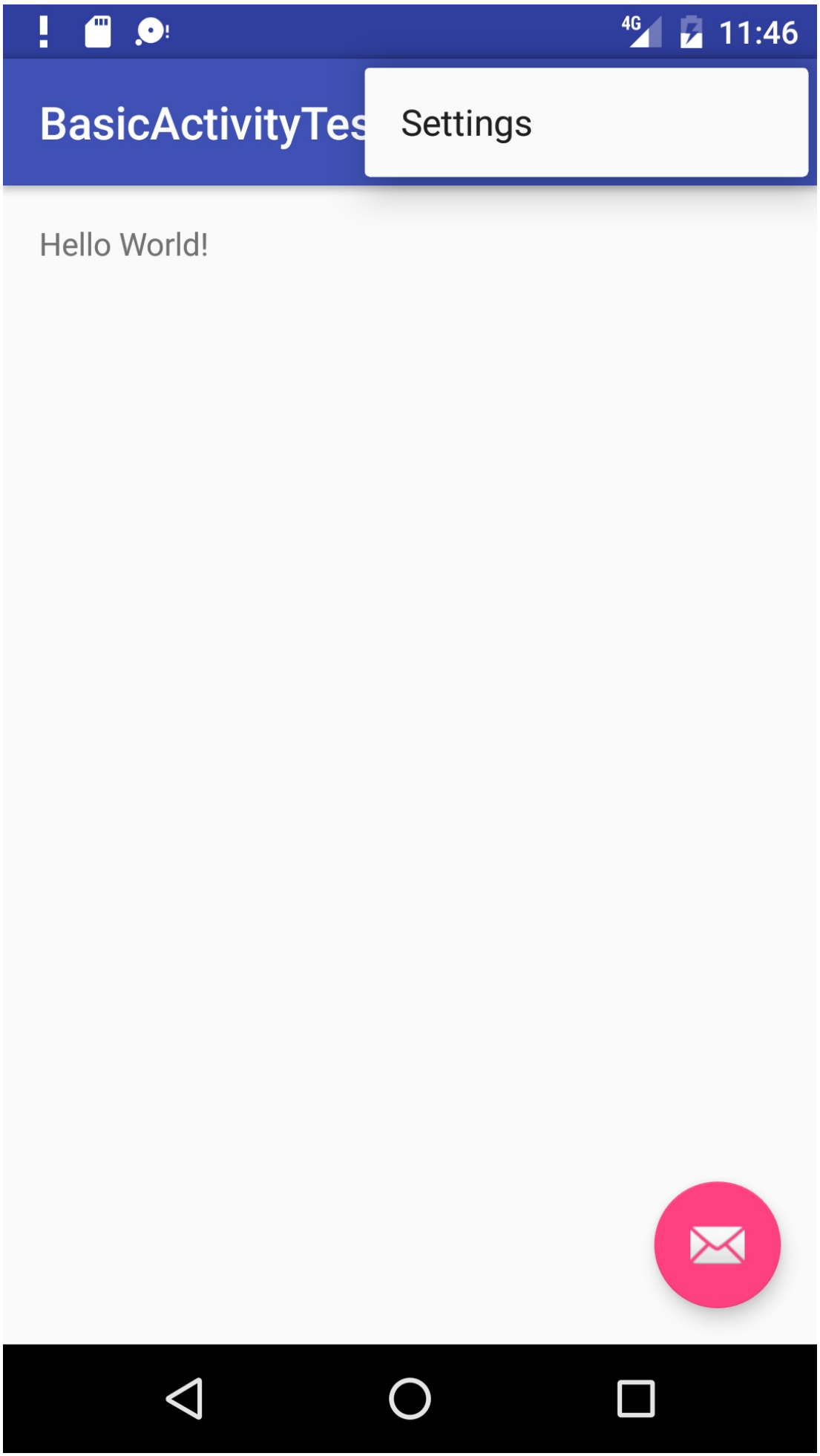
Dalam gambar di atas:

1. Setelan di navigasi samping (panel samping navigasi)
2. Setelan di menu opsi bilah aplikasi

Ikuti panduan desain ini untuk mengarahkan ke setelan:

- Jika aplikasi Anda menawarkan navigasi samping seperti panel samping navigasi, sertakan **Settings** di bawah semua item lainnya (kecuali **Help** dan **Send**).
- Jika aplikasi Anda tidak menawarkan navigasi samping, tempatkan **Settings** dalam menu opsi bilah aplikasi di bawah semua item lainnya (kecuali **Help** dan **Send feedback**).
  Catatan: Gunakan kata **Settings** dalam navigasi aplikasi untuk mengakses setelan. Jangan gunakan sinonim seperti "Options" atau "Preferences".

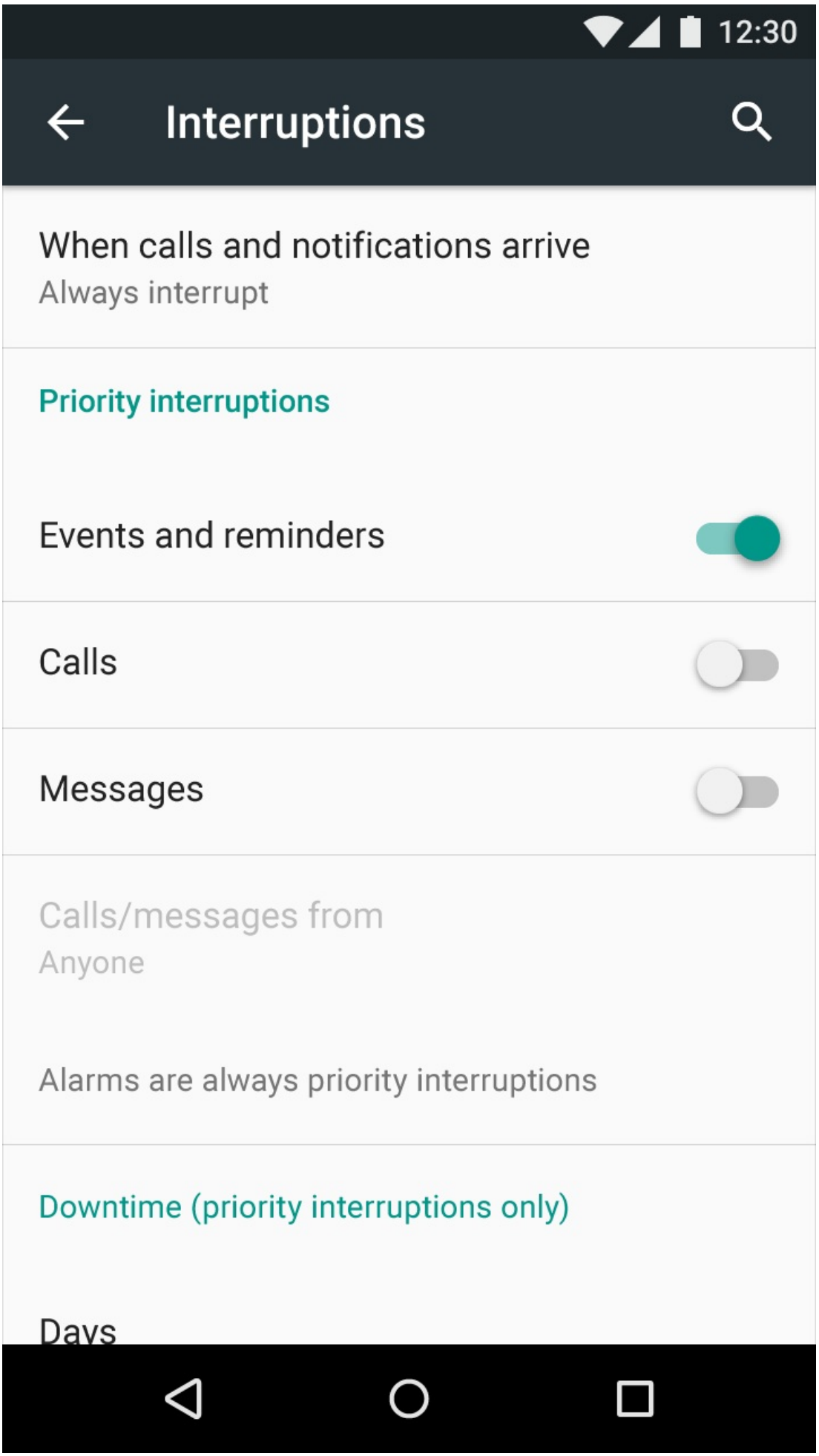
Tip: Android Studio menyediakan pintasan untuk mempersiapkan menu opsi bersama **Settings**. Jika Anda memulai proyek Android Studio untuk ponsel cerdas atau tablet dengan menggunakan template Basic Activity, aplikasi baru akan menyertakan **Settings** seperti yang ditampilkan di bawah ini:


11:46
BasicActivityTes
Settings
Hello World!

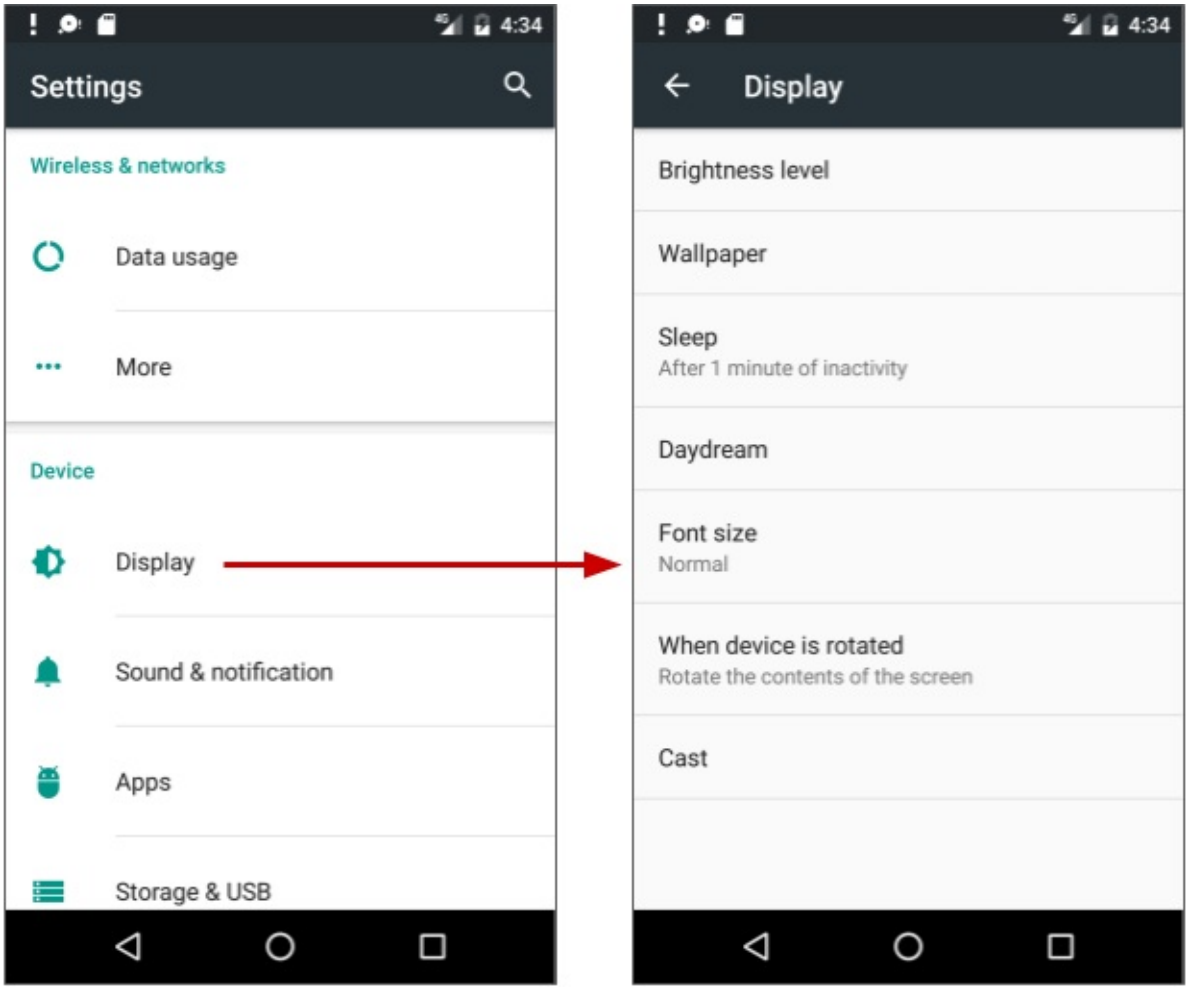
UI Setelan

Setelan seharusnya tesusun rapi, bisa diprediksi, dan berisi jumlah pilihan yang bisa dikelola. Pengguna harus bisa memahami dengan cepat semua setelan yang tersedia dan nilainya saat ini. Ikuti panduan desain ini:

- **7 setelan atau kurang**: Susun setelan sesuai dengan prioritas, yang paling penting berada di bagian atas.
- **7-15 setelan**: Kelompokkan setelan terkait di bawah pembatas bagian. Misalnya, dalam gambar di bawah, "Priority interruptions" dan "Downtime (priority interruptions only)" adalah pembatas bagian.


12:30
Interruptions
When calls and notifications arrive
Always interrupt
Priority interruptions
Events and reminders
Calls
Messages
Calls/messages from
Anyone
Alarms are always priority interruptions
Downtime (priority interruptions only)
Days

- **16 setelan atau lebih**: Kelompokkan setelan terkait ke dalam sub-layar tersendiri. Gunakan judul, seperti **Tampilan** pada layar Setelan utama (seperti yang ditampilkan pada sisi kiri gambar di bawah) untuk memungkinkan pengguna mengarahkan ke setelan tampilan (ditampilkan pada sisi kanan gambar di bawah):



Membangun setelan

Bangun sebuah setelan aplikasi menggunakan beragam subkelas dari kelas Preference daripada menggunakan objek View. Kelas ini menyediakan Tampilan yang ditampilkan untuk setiap setelan, dan mengaitkannya dengan antarmuka SharedPreferences untuk menyimpan/mengambil data preferensi.

Setiap Preference muncul sebagai suatu item dalam daftar. Subkelas langsung menyediakan kontainer untuk layout yang melibatkan beberapa setelan. Misalnya:

- PreferenceGroup: Menyatakan sekelompok setelan (objek Preference).
- PreferenceCategory Menyediakan judul nonaktif di atas grup sebagai pembatas bagian
- PreferenceScreen Menyatakan Preference tingkat atas yang merupakan akar hierarki Preference. Gunakan PreferenceScreen dalam layout di bagian atas setiap layar setelan.

Misalnya, untuk menyediakan pembatas dengan judul di antara grup setelan (seperti yang ditampilkan dalam gambar sebelumnya untuk 7-15 setelan), tempatkan masing-masing grup objek Preference di dalam PreferenceCategory. Untuk menggunakan layar tersendiri bagi grup, tempatkan setiap grup Preference di dalam PreferenceScreen.

Subkelas Preference untuk setelan lainnya menyediakan UI yang sesuai bagi pengguna untuk mengubah setelan. Misalnya:

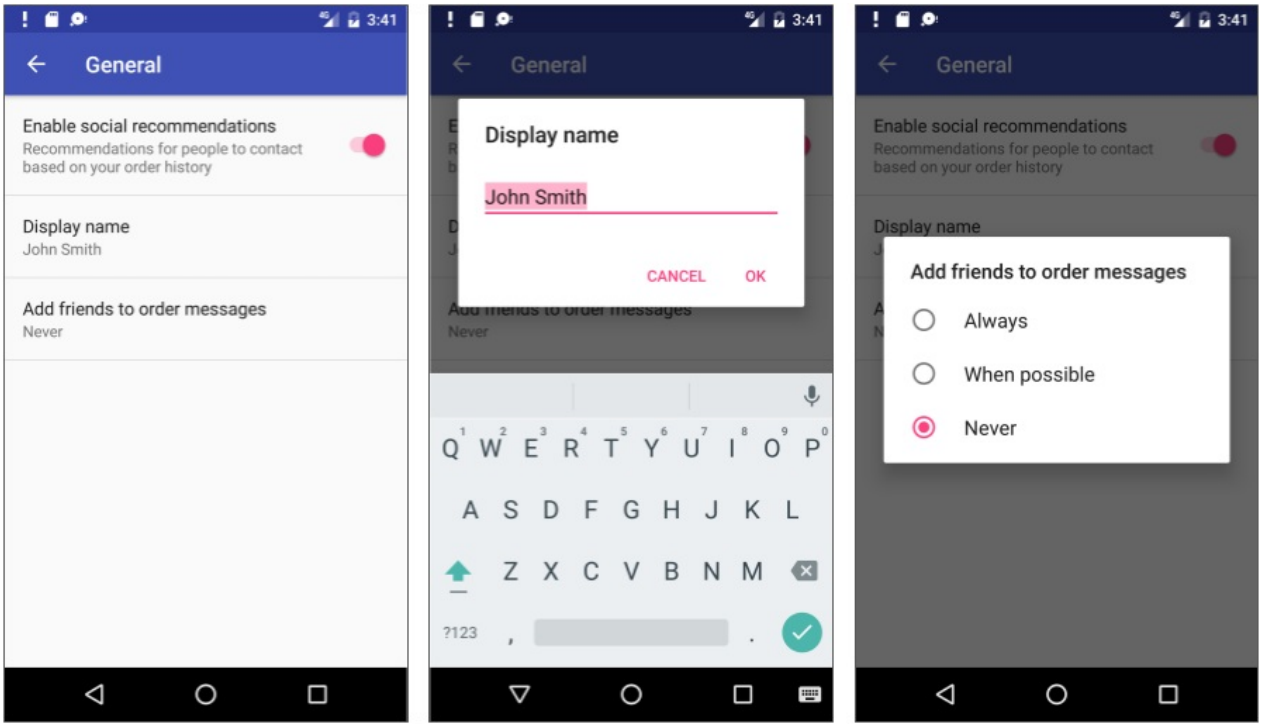
- CheckBoxPreference: Membuat item daftar yang menampilkan kotak centang untuk setelan yang diaktifkan atau dinonaktifkan. Nilai tersimpan adalah boolean (`true` jika dicentang).
- ListPreference: Membuat item yang membuka dialog berisi daftar tombol radio.

- SwitchPreference: Membuat opsi dua keadaan yang bisa beralih (misalnya aktif/nonaktif atau true/false).
- EditTextPreference: Membuat item yang membuka dialog berisi widget EditText. Nilai tersimpan adalah String.
- RingtonePreference: Memungkinkan pengguna memilih nada dering yang tersedia pada perangkat.

Definisikan daftar setelan Anda dalam XML, dengan menyediakan struktur yang mudah dibaca dan mudah diperbarui. Setiap subkelas Preference bisa dideklarasikan bersama elemen XML yang cocok dengan nama kelas, misalnya `<CheckBoxPreference>` .

Atribut XML untuk setelan

Contoh berikut dari template Settings Activity mendefinisikan sebuah layar dengan tiga setelan seperti yang ditampilkan dalam gambar di bawah ini: tombol beralih (di bagian atas layar pada sisi kiri), bidang entri teks (tengah), dan daftar tombol radio (kanan):

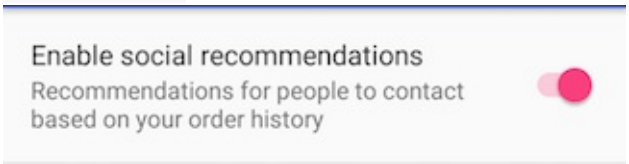


Akar hierarki setelan adalah layout `PreferenceScreen` :

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

Di dalam setelan ini ada tiga setelan:

- `SwitchPreference` : Menampilkan tombol beralih untuk menonaktifkan atau mengaktifkan opsi.

  Setelan ini memiliki atribut berikut:

  - `android:defaultValue` : Opsi telah diaktifkan (disetel ke `true`) secara default.
  - `android:summary` : Rangkuman teks muncul di bawah setelan. Untuk beberapa setelan, rangkuman tersebut akan berubah untuk menampilkan apakah opsi telah diaktifkan atau dinonaktifkan.
  - `android:title` : Judul setelan. Untuk `SwitchPreference` , judul muncul di sebelah kiri tombol beralih.
  - android:key: Kunci yang digunakan untuk menyimpan nilai setelan. Setiap setelan (objek Preference) memiliki pasangan nilai-kunci yang sesuai yang digunakan sistem untuk menyimpan setelan dalam file SharedPreferences

default untuk setelan aplikasi Anda.

```
<SwitchPreference
     android:defaultValue="true"
     android:key="example_switch"
     android:summary="@string/pref_description_social_recommendations"
     android:title="@string/pref_title_social_recommendations" />
```

- `EditTextPreference` : Menampilkan bidang teks tempat pengguna memasukkan teks.

  - Gunakan atribut EditText seperti `android:capitalize` dan `android:maxLines` untuk mendefinisikan penampilan bidang teks dan kontrol masukan.
  - Setelan default adalah sumber daya string `pref_default_display_name` .

```
<EditTextPreference
   android:capitalize="words"
   android:defaultValue="@string/pref_default_display_name"
   android:inputType="textCapWords"
   android:key="example_text"
   android:maxLines="1"
   android:selectAllOnFocus="true"
   android:singleLine="true"
   android:title="@string/pref_title_display_name" />
```

- `ListPreference` : Menampilkan dialog dengan tombol radio agar pengguna membuat satu pilihan.

  - Nilai default-nya disetel ke `-1` untuk tanpa pilihan.
  - Teks untuk tombol radio (Always, When possible, dan Never) didefinisikan dalam larik `pref_example_list_titles` dan ditetapkan oleh atribut `android:entries` .
  - Nilai untuk pilihan tombol radio didefinisikan dalam larik `pref_example_list_values` dan ditetapkan oleh atribut `android:entryValues` .
  - Tombol radio ditampilkan dalam dialog, yang biasanya memiliki tombol positif (**OK** atau **Accept**) dan negatif (**Cancel**). Akan tetapi, dialog setelan tidak memerlukan tombol ini, karena pengguna bisa menyentuh bagian luar dialog untuk menghilangkannya. Untuk menyembunyikan tombol ini, setel atribut `android:positiveButtonText` dan `android:negativeButtonText` ke `"@null"` .

```
<ListPreference
   android:defaultValue="-1"
   android:entries="@array/pref_example_list_titles"
   android:entryValues="@array/pref_example_list_values"
   android:key="example_list"
   android:negativeButtonText="@null"
   android:positiveButtonText="@null"
   android:title="@string/pref_title_add_friends_to_messages" />
```

Simpan file XML dalam direktori **res/xml/**. Walaupun bisa memberi nama file sesuka Anda, biasanya diberi nama **preferences.xml**.

Jika Anda menggunakan pustaka v7 appcompat dukungan dan memperluas Settings Activity dengan AppCompatActivity dan fragmen dengan PreferenceFragmentCompat, seperti yang ditampilkan di bagian berikutnya, ubah atribut XML setelan untuk menggunakan versi pustaka appcompat v7 dukungan. Misalnya, untuk setelan SwitchPreference, ubah `<SwitchPreference` dalam kode menjadi:

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
   <android.support.v7.preference.SwitchPreferenceCompat
   ... />
</PreferenceScreen>
```

Menampilkan setelan

Gunakan subkelas Activity atau Fragment yang khusus untuk menampilkan daftar setelan.

- Untuk aplikasi yang mendukung Android 3.0 dan versi yang lebih baru, praktik terbaik untuk setelan adalah menggunakan Settings Activity dan fragmen untuk setiap file XML preferensi:
  - Tambahkan kelas Settings Activity yang memperluas Activity dan menjadi host fragmen yang memperluas PreferenceFragment.
  - Agar tetap kompatibel dengan pustaka appcompat v7, perluas Settings Activity dengan AppCompatActivity, perluas fragmen dengan PreferenceFragmentCompat.
- Jika aplikasi Anda mendukung versi Android yang lebih lama dari 3.0 (API level 10 dan yang lebih rendah), bangun aktivitas setelan khusus sebagai ekstensi kelas PreferenceActivity.

Fragmen seperti PreferenceFragment menyediakan arsitektur yang lebih fleksibel untuk aplikasi Anda, dibandingkan menggunakan aktivitas saja. Fragmen mirip dengan bagian modular sebuah aktivitas—fragmen memiliki daur hidup sendiri dan menerima kejadian masukan sendiri, dan Anda bisa menambahkan atau membuang fragmen saat aktivitas sedang berjalan. Gunakan PreferenceFragment untuk mengontrol tampilan setelan Anda sebagai ganti PreferenceActivity bila memungkinkan.

Akan tetapi, untuk membuat layout dua-panel bagi layar besar bila Anda memiliki beberapa grup setelan, Anda bisa menggunakan aktivitas yang memperluas PreferenceActivity dan juga PreferenceFragment untuk menampilkan setiap daftar setelan. Anda akan melihat pola ini dengan template Settings Activity seperti yang nanti dijelaskan dalam bab ini di "Menggunakan template Settings Activity".

Contoh berikut menampilkan bagaimana agar tetap kompatibel dengan pustaka appcompat v7 dengan memperluas Settings Activity menggunakan AppCompatActivity, dan memperluas fragmen dengan PreferenceFragmentCompat. Untuk menggunakan pustaka ini dan versi `PreferenceFragmentCompat` dari PreferenceFragment, Anda juga harus menambahkan pustaka android.support:preference-v7 ke bagian `dependencies` file **build.gradle (Module: app)**:

```
dependencies {
   ...
   compile 'com.android.support:preference-v7:25.0.1'
}
```

Anda juga perlu menambahkan deklarasi `preferenceTheme` berikut ke `AppTheme` dalam file **styles.xml**:

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
   ...
   <item name="preferenceTheme">@style/PreferenceThemeOverlay</item>
</style>
```

Menggunakan PreferenceFragment

Yang berikut ini menampilkan cara menggunakan PreferenceFragment untuk menampilkan daftar setelan, dan cara menambahkan PreferenceFragment ke sebuah aktivitas untuk setelan. Agar tetap kompatibel dengan pustaka appcompat v7, perluas Settings Activity menggunakan AppCompatActivity, dan perluas fragmen menggunakan PreferenceFragmentCompat untuk setiap file XML preferensi.

Ganti metode `onCreate()` yang secara otomatis dihasilkan dengan metode onCreatePreferences() untuk memuat file preferensi dengan `setPreferencesFromResource()` :

```
public static class SettingsFragment extends PreferenceFragment {
    @Override
    public void onCreatePreferences(Bundle savedInstanceState,
                                            String rootKey) {
        setPreferencesFromResource(R.xml.preferences, rootKey);
    }
}
```

Seperti yang ditampilkan dalam kode di atas, Anda mengaitkan layout XML setelan dengan fragmen selama callback `onCreatePreferences()` dengan memanggil setPreferencesFromResource() dengan dua argumen:

- `R.xml.` dan nama file XML (`preferences`).
- dalam hal ini `rootKey` mengidentifikasi akar preferensi dalam `PreferenceScreen` .

  ```
  setPreferencesFromResource(R.xml.preferences, rootKey);
  ```

Anda kemudian bisa membuat Activity untuk setelan (bernama `SettingsActivity`) yang memperluas AppCompatActivity, dan menambahkan fragmen setelan:

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

        // Display the fragment as the main content.
        getSupportFragmentManager().beginTransaction()
                .replace(android.R.id.content, new SettingsFragment())
                .commit();
        ...
    }
}
```

Kode di atas adalah pola umum yang digunakan untuk menambahkan fragmen ke sebuah aktivitas sehingga fragmen muncul sebagai materi utama aktivitas. Anda menggunakan:

- `getFragmentManager()` jika kelas memperluas `Activity` dan fragmen memperluas `PreferenceFragment` .
- `getSupportFragmentManager()` jika kelas memperluas `AppCompatActivity` dan fragmen memperluas `PreferenceFragmentCompat` .

Untuk informasi selengkapnya tentang fragmen, lihat Fragmen.

Untuk mempersiapkan navigasi bagi aktivitas setelan, pastikan mendeklarasikan induk Settings Activity menjadi MainActivity dalam file AndroidManifest.xml.

Memanggil aktivitas setelan

Jika Anda mengimplementasikan menu opsi dengan item **Settings**, gunakan maksud berikut untuk memanggil Settings Activity dengan metode `onOptionsItemSelected()` bila pengguna mengetuk **Settings** (menggunakan `action_settings` untuk ID sumber daya menu **Settings**):

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   int id = item.getItemId();
   // ... Handle other options menu items
   if (id == R.id.action_settings) {
      Intent intent = new Intent(this, SettingsActivity.class);
      startActivity(intent);
      return true;
      }
   return super.onOptionsItemSelected(item);
}
```

Jika Anda mengimplementasikan panel samping navigasi dengan item **Settings**, gunakan maksud berikut memanggil Settings Activity dengan metode `onNavigationItemSelected()` bila pengguna mengetuk **Settings** (menggunakan `action_settings` untuk ID sumber daya menu **Settings**):

```
@Override
public boolean onNavigationItemSelected(MenuItem item) {
   int id = item.getItemId();
   if (id == R.id.action_settings) {
      Intent intent = new Intent(this, SettingsActivity.class);
      startActivity(intent);
   } else if ...
   // ... Handle other navigation drawer items
   return true;
}
```

Menyetel nilai default untuk setelan

Jika pengguna mengubah setelan, sistem akan menyimpan perubahan ke file SharedPreferences. Seperti yang telah Anda pelajari dalam pelajaran lain, preferensi bersama memungkinkan Anda membaca dan menulis data primitif dalam jumlah kecil sebagai pasangan kunci/nilai ke file pada storage perangkat.

Aplikasi harus melakukan inisialisasi file SharedPreferences dengan nilai default untuk masing-masing setelan saat pengguna membuka aplikasi untuk pertama kali. Ikuti langkah-langkah ini:

1. Pastikan untuk menetapkan nilai default untuk setiap setelan dalam file XML dengan menggunakan atribut `android:defaultValue` :

   ```
   ...
   <SwitchPreference
           android:defaultValue="true"
           ... />
   ...
   ```

2. Dari metode onCreate() di aktivitas utama aplikasi—dan di aktivitas lainnya yang digunakan pengguna untuk memasuki aplikasi Anda untuk pertama kali—panggil setDefaultValues():

   ```
   ...
     PreferenceManager.setDefaultValues(this,
                                 R.xml.preferences, false);
   ...
   ```

Langkah 2 memastikan bahwa aplikasi diinisialisasi dengan benar menggunakan setelan default. Metode `setDefaultValues()` menggunakan tiga argumen:

- Konteks aplikasi, seperti `this` .

- ID sumber daya (`preferences`) untuk file XML layout setelan menyertakan nilai default yang disetel oleh Langkah 1 di atas.
- Boolean yang menunjukkan apakah nilai default harus disetel lebih dari satu kali. Bila `false` , sistem akan menyetel nilai default hanya jika metode ini belum pernah dipanggil sebelumnya (atau KEY_HAS_SET_DEFAULT_VALUES dalam file SharedPreferences nilai default adalah false). Asalkan Anda menyetel argumen ketiga ini ke `false` , Anda bisa dengan aman memanggil metode ini setiap kali aktivitas dimulai tanpa menggantikan nilai setelan yang disimpan pengguna dengan menyetel ulang ke nilai defaultnya. Akan tetapi, jika Anda menyetelnya ke `true` , metode akan menggantikan nilai-nilai sebelumnya dengan default.

Membaca nilai setelan

Setiap Preference yang Anda tambahkan memiliki pasangan nilai-kunci yang digunakan sistem untuk menyimpan setelan dalam file SharedPreferences default untuk setelan aplikasi Anda. Bila pengguna mengubah setelan, sistem akan memperbarui nilai yang bersangkutan dalam file SharedPreferences untuk Anda. Satu-satunya saat Anda harus berinteraksi langsung dengan file SharedPreferences terkait adalah bila perlu membaca nilai untuk menentukan perilaku aplikasi berdasarkan setelan pengguna.

Semua preferensi aplikasi disimpan secara default ke file yang bisa diakses dari mana saja dalam aplikasi dengan memanggil metode statis PreferenceManager.getDefaultSharedPreferences(). Metode ini membutuhkan konteks dan mengembalikan objek SharedPreferences berisi semua pasangan nilai-kunci yang dikaitkan dengan objek Preference.

Misalnya, cuplikan kode berikut menampilkan membaca salah satu nilai preferensi dari metode `onCreate()` aktivitas utama:

```
...
SharedPreferences sharedPref =
    PreferenceManager.getDefaultSharedPreferences(this);
Boolean switchPref = sharedPref
    .getBoolean("example_switch", false);
...
```

Cuplikan kode di atas menggunakan `PreferenceManager.getDefaultSharedPreferences(this)` untuk mendapatkan setelan sebagai objek SharedPreferences (`sharedPref`).

Kemudian menggunakan `getBoolean()` untuk mendapatkan nilai boolean preferensi yang menggunakan kunci `"example_switch"` . Jika tidak ada nilai untuk kunci, metode `getBoolean()` akan menyetel nilai ke `false` .

Mendengarkan perubahan setelan

Ada sejumlah alasan mengapa Anda ingin mempersiapkan listener untuk setelan tertentu:

- Jika perubahan nilai setelan juga memerlukan perubahan rangkuman setelan, Anda bisa mendengarkan perubahan, kemudian mengubah rangkuman dengan nilai setelan yang baru.
- Jika setelan memerlukan sejumlah opsi lagi, Anda bisa mendengarkan perubahan dan langsung merespons dengan menampilkan opsi.
- Jika setelan membuat setelan lain tidak terpakai atau tidak layak, Anda bisa mendengarkan perubahan dan langsung merespons dengan menonaktifkan setelan lainnya.

Untuk mendengarkan setelan, gunakan antarmuka Preference.OnPreferenceChangeListener, yang menyertakan metode onPreferenceChange() yang mengembalikan nilai setelan baru.

Dalam contoh berikut, listener mengambil nilai baru setelah setelan diubah, dan mengubah rangkuman setelan (yang muncul pada setelan di UI) untuk menampilkan nilai baru. Ikuti langkah-langkah ini:

1. Gunakan file preferensi bersama, seperti yang dijelaskan dalam bab sebelumnya, untuk menyimpan nilai preferensi (setelan). Deklarasikan variabel-variabel berikut dalam definisi kelas SettingsFragment:

```
public class SettingsFragment extends PreferenceFragment {
   private SharedPreferences mPreferences;
   private String sharedPrefFile = "com.example.android.settingstest";
   ...
}
```

2. Tambahkan yang berikut ini ke metode `onCreate()` pada SettingsFragment untuk mendapatkan preferensi yang didefinisikan oleh kunci `example_switch`, dan untuk menyetel teks awal (sumber daya string `option_on`) untuk ringkasan:

```
@Override
public void onCreate(Bundle savedInstanceState) {
   ...
   mPreferences =
            this.getActivity()
            .getSharedPreferences(sharedPrefFile, MODE_PRIVATE);
   Preference preference = this.findPreference("example_switch");
   preference.setSummary(mPreferences.getString("summary",
                                getString(R.string.option_on)));
   ...
}
```

3. Tambahkan kode berikut ke `onCreate()` setelah kode di langkah sebelumnya:

```
...
preference.setOnPreferenceChangeListener(new
                           Preference.OnPreferenceChangeListener() {
   @Override
   public boolean onPreferenceChange(Preference preference,
                           Object newValue) {
      if ((Boolean) newValue == true) {
         preference.setSummary(R.string.option_on);
         SharedPreferences.Editor preferencesEditor =
                                          mPreferences.edit();
         preferencesEditor.putString("summary",
                           getString(R.string.option_on)).apply();
      } else {
         preference.setSummary(R.string.option_off);
         SharedPreferences.Editor preferencesEditor =
                                          mPreferences.edit();
         preferencesEditor.putString("summary",
                           getString(R.string.option_off)).apply();
      }
      return true;
   }
});
...
```

   Kode tersebut melakukan hal berikut:

   i. Mendengarkan perubahan setelan switch dengan menggunakan onPreferenceChange(), dan mengembalikan `true`:

   ```
   @Override
   public boolean onPreferenceChange(Preference preference,
                     Object newValue) {
    ...
    return true;
   }
   ```

   ii. Menentukan nilai boolean baru (`newValue`) untuk setelan setelah perubahan (`true` atau `false`):

```
if ((Boolean) newValue == true) {
 ...
} else {
 ...
}
```

iii. Mengedit file Shared Preferences (seperti yang dijelaskan dalam praktik sebelumnya) menggunakan SharedPreferences.Editor:

```
...
preference.setSummary(R.string.option_on);
SharedPreferences.Editor preferencesEditor =
                                  mPreferences.edit();
...
```

iv. Memasukkan nilai baru sebagai string dalam rangkuman dengan menggunakan putString() dan menerapkan perubahan menggunakan apply():

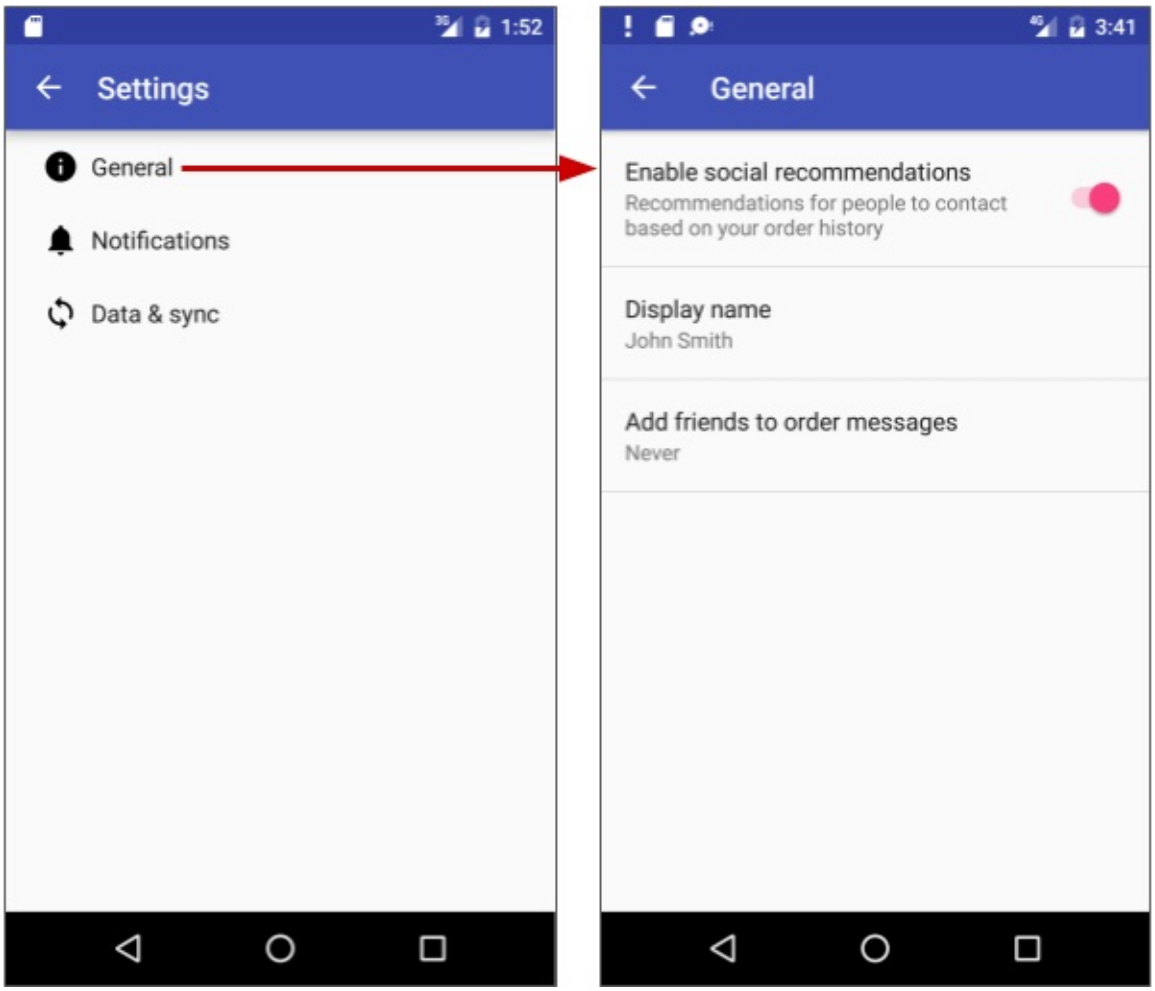
```
...
preferencesEditor.putString("summary",
                   getString(R.string.option_on)).apply();
...
```

Menggunakan template Settings Activity

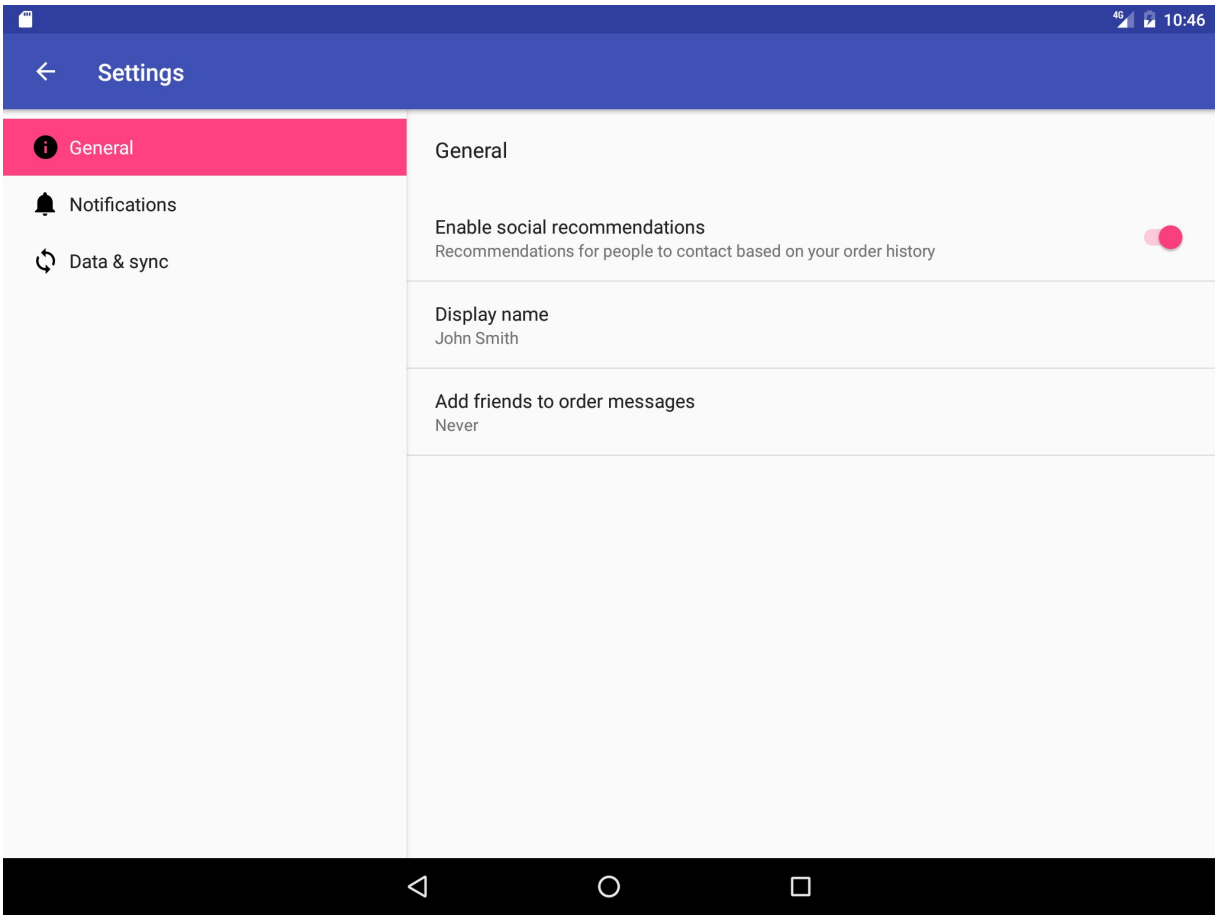
Jika Anda perlu membangun sejumlah sublayar setelan dan ingin memanfaatkan layar berukuran tablet, serta mempertahankan kompatibilitas dengan versi lama Android untuk tablet, Android Studio menyediakan sebuah pintasan: template Settings Activity.

Template Settings Activity telah terisi dengan setelan yang bisa Anda sesuaikan untuk aplikasi, dan menyediakan layout yang berbeda untuk ponsel cerdas dan tablet:

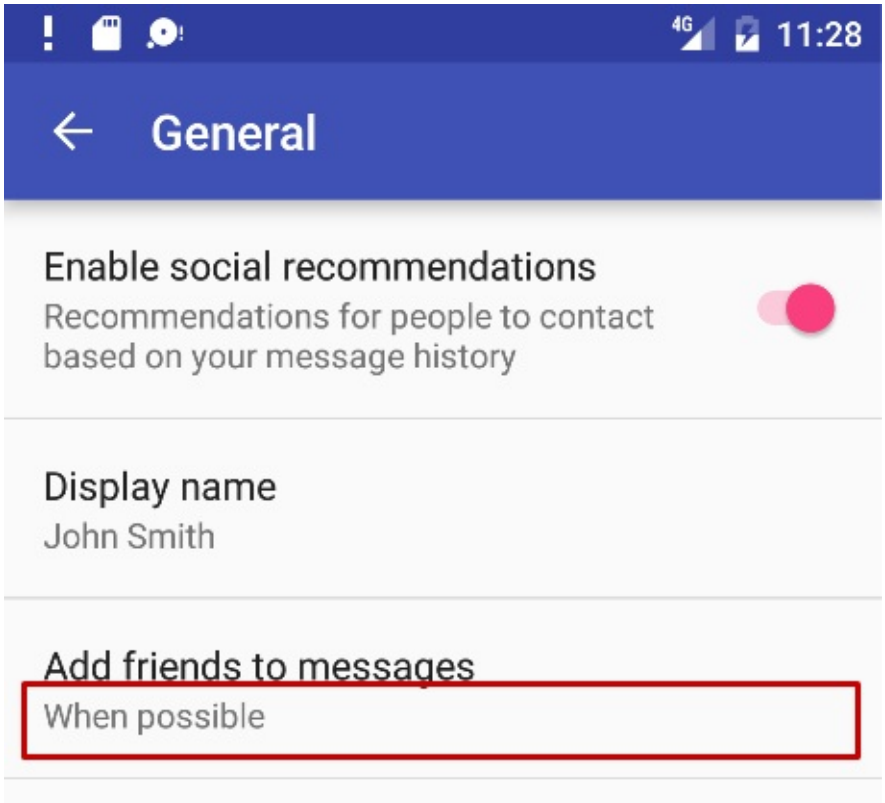
- *Ponsel cerdas*: Layar Settings utama dengan tautan header untuk setiap grup setelan, misalnya General untuk setelan umum, seperti yang ditampilkan di bawah ini.


1:52
Settings
General
Notifications
Data & sync
3:41
General
Enable social recommendations
Recommendations for people to contact based on your order history
Display name
John Smith
Add friends to order messages
Never

- *Tablet*: Layout layar detail/master dengan tautan header untuk setiap grup di sebelah kiri (master), dan grup setelan di sebelah kanan (detail), seperti yang ditampilkan dalam gambar di bawah ini.



Template Settings Activity juga menyediakan fungsi untuk mendengarkan perubahan setelan dan mengubah rangkuman untuk merefleksikan perubahan setelan. Misalnya, jika Anda mengubah setelan "Add friends to messages" (opsinya adalah **Always**, **When possible**, atau **Never**), opsi yang dipilih akan muncul di rangkuman di bawah setelan:



Secara umum, Anda tidak perlu mengubah kode template Settings Activity untuk menyesuaikan aktivitas bagi setelan yang diinginkan di aplikasi. Anda bisa menyesuaikan judul, rangkuman, nilai yang memungkinkan, nilai default setelan tanpa mengubah kode template, dan bahkan menambahkan setelan lain ke grup yang disediakan. Untuk menyesuaikan setelan, edit sumber daya larik string dan string di file strings.xml dan atribut layout untuk setiap setelan dalam file di direktori **xml**.

Anda menggunakan kode template Settings Activity apa adanya. Agar berfungsi untuk aplikasi Anda, tambahkan kode ke Main Activity untuk menyetel nilai-nilai setelan default, dan untuk *membaca* serta *menggunakan* nilai setelan tersebut, seperti yang ditampilkan dalam bab ini nanti.

Menyertakan template Settings Activity dalam proyek Anda

Untuk menyertakan template Settings Activity dalam proyek aplikasi Anda di Android Studio, ikuti langkah-langkah ini:

1. Pilih **New > Activity > Settings Activity**.
2. Dalam dialog yang muncul, terima Activity Name (**SettingsActivity** adalah nama yang disarankan) dan Title (**Settings**).
3. Klik tiga titik di ujung bidang Hierarchical Parent dan pilih aktivitas induk (biasanya **MainActivity**), sehingga navigasi Naik di Settings Activity akan mengembalikan pengguna ke MainActivity. Memilih aktivitas induk secara otomatis akan memperbarui file AndroidManifest.xml untuk mendukung navigasi Naik.

Template Settings Activity membuat file XML dalam direktori **res > xml**, tempat Anda bisa menambahkan atau menyesuaikan setelan yang diinginkan:

- pref_data_sync.xml: Layout PreferenceScreen untuk setelan "Data & Sync".
- pref_general.xml: Layout PreferenceScreen untuk setelan "General".
- pref_headers.xml: Layout header untuk layar utama Settings.
- pref_notification.xml: Layout PreferenceScreen untuk setelan "Notifications".

  Layout XML di atas menggunakan beragam subkelas dari kelas Preference, bukan objek View, dan subkelas langsung menyediakan kontainer untuk layout yang melibatkan beberapa setelan. Misalnya PreferenceScreen menyatakan Preference tingkat atas, yaitu akar hierarki Preference. File di atas menggunakan PreferenceScreen di bagian atas setiap layar setelan. Subkelas Preference untuk setelan lainnya menyediakan UI yang sesuai bagi pengguna untuk mengubah setelan. Misalnya:

  - CheckBoxPreference: Kotak centang untuk setelan yang diaktifkan atau dinonaktifkan.
  - ListPreference: Dialog yang berisi daftar tombol radio.
  - SwitchPreference: Opsi untuk beralih dalam dua keadaan (misalnya aktif/nonaktif atau true/false).
  - EditTextPreference: Dialog dengan sebuah widget EditText.
  - RingtonePreference: Dialog dengan nada dering di perangkat.

Template Settings Activity juga menyediakan yang berikut ini:

- Sumber daya string di file strings.xml di direktori **res > values** yang bisa Anda sesuaikan untuk setelan yang diinginkan.

  Semua yang digunakan di Settings Activity, seperti judul untuk setelan, larik string untuk daftar, dan keterangan untuk setelan, didefinisikan sebagai sumber daya string di akhir file ini. String ini ditandai dengan komentar seperti `<!-- Strings related to Settings -->` dan `<!-- Example General settings -->`.

  Tip: Anda bisa mengedit string ini untuk menyesuaikan setelan yang diperlukan untuk aplikasi.

- SettingsActivity dalam direktori **java > com.example.android.*projectname***, yang bisa Anda gunakan apa adanya.

  Aktivitas yang menampilkan setelan. `SettingsActivity` memperluas `AppCompatPreferenceActivity` untuk mempertahankan kompatibilitas dengan Android versi lama.

- AppCompatPreferenceActivity di direktori **java > com.example.android.*projectname***, yang bisa Anda gunakan apa adanya.

Aktivitas ini adalah kelas helper yang digunakan oleh SettingsActivity untuk mempertahankan kompatibilitas mundur dengan Android versi sebelumnya.

Menggunakan header preferensi

Template Settings Activity menampilkan header preferensi pada layar utama yang memisahkan setelan ke dalam beberapa kategori (**General**, **Notifications**, dan **Data & sync**). Pengguna mengetuk heading untuk mengakses setelan yang ada pada heading itu. Pada tampilan tablet yang lebih besar (lihat gambar sebelumnya), header muncul di panel kiri dan setelan untuk setiap header muncul di panel kanan.

Untuk mengimplementasikan header, template menyediakan file pref_headers.xml:

```
<preference-headers xmlns:android="http://schemas.android.com/apk/res/android">
  <header
      android:fragment=
           "com.example.android.droidcafe.SettingsActivity$GeneralPreferenceFragment"
      android:icon="@drawable/ic_info_black_24dp"
      android:title="@string/pref_header_general" />

  <header
      android:fragment=
      "com.example.android.droidcafe.SettingsActivity$NotificationPreferenceFragment"
      android:icon="@drawable/ic_notifications_black_24dp"
      android:title="@string/pref_header_notifications" />

  <header
      android:fragment=
          "com.example.android.droidcafe.SettingsActivity$DataSyncPreferenceFragment"
      android:icon="@drawable/ic_sync_black_24dp"
      android:title="@string/pref_header_data_sync" />
</preference-headers>
```

File header XML mendaftarkan setiap kategori preferensi dan mendeklarasikan fragmen yang berisi preferensi terkait.

Untuk menampilkan header, template menggunakan metode `onBuildHeaders()` berikut:

```
@Override
@TargetApi(Build.VERSION_CODES.HONEYCOMB)
public void onBuildHeaders(List<Header> target) {
        loadHeadersFromResource(R.xml.pref_headers, target);
}
```

Cuplikan kode di atas menggunakan metode `loadHeadersFromResource()` kelas PreferenceActivity untuk memuat header dari sumber daya XML (pref_headers.xml). Anotasi TargetApi memberi tahu alat (bantu) pemindai kode Lint di Android Studio bahwa kelas atau metode menargetkan API level tertentu yang ditetapkan sebagai SDK level minimal dalam manifes. Sebaliknya lint akan menghasilkan kesalahan dan peringatan saat menggunakan fungsionalitas baru yang tidak tersedia di API level target.

Menggunakan PreferenceActivity bersama fragmen

Template Settings Activity menyediakan suatu aktivitas (SettingsActivity) yang memperluas PreferenceActivity untuk membuat layout dua-panel guna mendukung layar besar, *juga* menyertakan fragmen dalam aktivitas untuk menampilkan daftar setelan. Inilah pola yang berguna jika Anda memiliki beberapa grup setelan dan harus mendukung layar berukuran tablet serta ponsel cerdas.

Yang berikut ini menampilkan cara menggunakan aktivitas yang memperluas `PreferenceActivity` untuk menjadi host bagi satu atau beberapa fragmen (PreferenceFragment) yang menampilkan setelan aplikasi. Aktivitas bisa menjadi host beberapa fragmen, seperti `GeneralPreferenceFragment` dan `NotificationPreferenceFragment`, dan setiap definisi fragmen menggunakan `addPreferencesFromResource` untuk memuat setelan dari file preferensi XML:

```
public class SettingsActivity extends AppCompatPreferenceActivity {
   ...
   public static class GeneralPreferenceFragment extends
                                              PreferenceFragment {
   @Override
   public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            addPreferencesFromResource(R.xml.pref_general);

   ...
   }
   public static class NotificationPreferenceFragment extends
                                              PreferenceFragment {
   ...
}
```

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Menambahkan Setelan ke Aplikasi

Ketahui selengkapnya

- Dokumentasi Android Studio:
  - Panduan Pengguna Android Studio
- Panduan Android API, bagian "Kembangkan":
  - Setelan (pengkodean)
  - Kelas Preference
  - PreferenceFragment
  - Fragment
  - SharedPreferences
  - Menyimpan Rangkaian Nilai-Kunci
- Spesifikasi Desain Material:
  - Setelan (desain)

10.1: SQLite Primer

Materi:

- Database SQL
- SQLite
- Tabel contoh
- Transaksi
- Bahasa kueri
- Kueri untuk Android SQLite
- Kursor
- Ketahui selengkapnya

Kursus ini beranggapan bahwa Anda sudah familier dengan database secara umum, terutama database SQL, dan bahasa SQL yang digunakan untuk berinteraksi dengannya. Bab ini sebagai penyegar dan referensi singkat saja.

Database SQL

- Simpan data dalam baris dan kolom tabel.
- Perpotongan suatu baris dan kolom disebut bidang.
- Bidang berisi data, referensi ke bidang lain, atau referensi ke tabel lain.
- Baris diidentifikasi melalui ID unik.
- Kolom diidentifikasi melalui nama yang unik per tabel.

Anggaplah ini seperti spreadsheet dengan baris, kolom, dan sel, dalam hal ini sel bisa berisi data, referensi ke sel lain, dan tautan ke sheet lain.

SQLite

SQLite pustaka perangkat lunak yang mengimplementasikan mesin database SQL yang:

- mandiri (tidak memerlukan komponen lain)
- tanpa server (tidak memerlukan server-backend)
- konfigurasi-nol (tidak perlu dikonfigurasi untuk aplikasi Anda)
- transaksional (perubahan dalam satu transaksi di SQLite akan diterapkan pada semua bagian atau tidak diterapkan sama sekali)

SQLite adalah mesin database yang paling banyak diterapkan di seluruh dunia. Kode sumber untuk SQLite berada dalam domain publik.

Untuk detail database SQLite, lihat situs web SQLite.

Tabel contoh

SQLite menyimpan data dalam tabel.

Dengan anggapan berikut ini:

- Database DATABASE_NAME
- Tabel WORD_LIST_TABLE
- Kolom untuk _id, kata, dan keterangan

Setelah menyisipkan kata "alpha" dan "beta", dalam hal ini alpha memiliki dua definisi, tabelnya akan terlihat seperti ini:

DATABASE_NAME

WORD_LIST_TABLE		
_id	**word**	**definition**
1	"alpha"	"first letter"
2	"beta"	"second letter"
3	"alpha"	"particle"

Anda bisa menemukan apa yang ada dalam baris tertentu menggunakan _id, atau bisa mengambil baris dengan merumuskan kueri yang memilih baris dari tabel yang menetapkan pembatas. Gunakan bahasa kueri SQL yang dibahas di bawah ini untuk membuat kueri.

Transaksi

Transaksi adalah urutan operasi yang dijalankan sebagai satu unit kerja logis. Unit kerja logis harus menunjukkan empat properti, yaitu properti atomisitas, konsistensi, isolasi, dan ketahanan (ACID), untuk memenuhi syarat sebagai transaksi.

Semua perubahan dalam satu transaksi di SQLite diterapkan pada semua bagian atau tidak diterapkan sama sekali), bahkan jika tindakan menulis perubahan ke disk terputus karena

- program mogok,
- sistem operasi mogok, atau
- listrik mati.

Contoh transaksi:

- Mengirimkan uang dari rekening tabungan ke rekening giro.
- Memasukkan suatu istilah dan definisi ke dalam kamus.
- Mengikat changelist ke cabang utama.

ACID

- **Atomisitas.** Semua modifikasi data dijalankan, atau tidak ada yang dijalankan.
- **Konsistensi.** Bila selesai, suatu transaksi harus meninggalkan semua data dalam keadaan konsisten.
- **Isolasi.** Modifikasi yang dibuat oleh transaksi yang terjadi bersamaan harus diisolasi dari modifikasi yang dibuat oleh transaksi bersamaan lainnya. Suatu transaksi mengenali data baik dalam keadaan sebelum transaksi bersamaan lainnya memodifikasinya, maupun mengenali data setelah transaksi kedua selesai, namun tidak mengenali keadaan di antara keduanya.
- **Ketahanan.** Setelah transaksi selesai, efeknya diterapkan permanen dalam sistem. Modifikasi tetap ada bahkan seandainya listrik mati.

Selengkapnya mengenai transaksi.

Bahasa kueri

Anda menggunakan bahasa kueri SQL khusus untuk berinteraksi dengan database. Kueri bisa sangat kompleks, namun operasi dasarnya adalah

- menyisipkan baris
- menghapus baris
- memperbarui nilai dalam baris
- mengambil baris yang memenuhi kriteria tertentu

Di Android, objek database menyediakan metode praktis untuk menyisipkan, menghapus, dan memperbarui database. Anda hanya perlu memahami SQL untuk mengambil data.

Keterangan lengkap mengenai bahasa kueri.

Struktur kueri

Kueri SQL sangat terstruktur dan berisi bagian dasar berikut:

- SELECT kata, keterangan FROM WORD_LIST_TABLE WHERE kata="alpha"

Versi generik kueri contoh:

- SELECT kolom FROM tabel WHERE kolom="value"

Bagian:

- SELECT kolom—memilih kolom untuk dikembalikan. Gunakan * untuk mengembalikan semua kolom.
- FROM tabel—menetapkan tabel yang dipakai untuk mendapatkan hasil.
- WHERE—kata kunci untuk ketentuan yang harus terpenuhi.
- kolom="value"—ketentuan yang harus dipenuhi.
  - operator umum: =, LIKE, <, >
- AND, OR—menghubungkan beberapa ketentuan dengan operator logika.
- ORDER BY—mengabaikan urutan default, atau menetapkan ASC untuk menaik, DESC untuk menurun.
- LIMIT adalah kata kunci yang sangat berguna jika Anda ingin mendapatkan jumlah hasil yang terbatas.

Kueri contoh

1	SELECT * FROM WORD_LIST_TABLE	Mendapatkan seluruh tabel.
2	SELECT word, definition FROM WORD_LIST_TABLE WHERE _id > 2	Mengembalikan `[["alpha", "particle"]]`
3	SELECT _id FROM WORD_LIST_TABLE WHERE word="alpha" AND definition LIKE "%art%"	Mengembalikan id kata alpha dengan substring "art" dalam definisi. `[["3"]]`
4	SELECT * FROM WORD_LIST_TABLE ORDER BY word DESC LIMIT 1	Mengurutkan secara terbalik dan mengambil item pertama. Hal ini akan memberi Anda item terakhir untuk setiap urutan penyortiran. Pengurutan berdasarkan kolom pertama, dalam hal ini, _id. `[["3", "alpha", "particle"]]`
5	SELECT * FROM WORD_LIST_TABLE LIMIT 2,1	Mengembalikan 1 item mulai posisi 2. Penghitungan posisi dimulai dari 1 (bukan nol!). Mengembalikan `[["2", "beta", "second letter"]]`

Anda bisa berlatih dengan membuat dan menjalankan kueri database pada situs web Fiddle dan HeadFirst Labs.

Kueri untuk Android SQLite

Anda bisa mengirimkan kueri ke database SQLite sistem Android sebagai kueri mentah atau sebagai parameter.

- rawQuery(String sql, String[] selectionArgs) menjalankan SQL yang disediakan dan mengembalikan Cursor rangkaian hasil.

Tabel berikut menampilkan bagaimana dua kueri pertama dari hal di atas akan terlihat sebagai kueri mentah.

1	String query = "SELECT * FROM WORD_LIST_TABLE"; rawQuery(query, null);
2	query = "SELECT word, definition FROM WORD_LIST_TABLE WHERE _id> ? "; String[] selectionArgs = new String[]{"2"} rawQuery(query, selectionArgs) ;

- query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit) menjalankan kueri untuk tabel tertentu, yang mengembalikan Cursor melalui rangkaian hasil.

Inilah kueri yang menampilkan cara mengisi argumen:

```
SELECT * FROM WORD_LIST_TABLE
WHERE word="alpha"
ORDER BY word ASC
LIMIT 2,1;
```

Mengembalikan:

```
[["alpha", "particle"]]
String table = "WORD_LIST_TABLE"
String[] columns = new String[]{"*"};
String selection = "word = ?"
String[] selectionArgs = new String[]{"alpha"};
String groupBy = null;
String having = null;
String orderBy = "word ASC"
String limit = "2,1"

query(table, columns, selection, selectionArgs, groupBy, having, orderBy, limit);
```

Perhatikan, dalam kode sebenarnya, Anda tidak akan membuat variabel untuk nilai nol. Lihat dokumentasi SQLiteDatabase untuk versi metode ini dengan parameter berbeda.

Kursor

Kueri selalu mengembalikan objek Cursor. Cursor adalah antarmuka objek yang menyediakan akses baca-tulis acak ke rangkaian hasil yang dikembalikan oleh kueri database. Kursor menunjuk elemen pertama dalam hasil kueri.

Kursor adalah pointer ke dalam baris data terstruktur. Anda bisa menganggapnya sebagai pointer ke baris tabel.

Kelas Cursor menyediakan metode untuk menggerakkan kursor melalui struktur itu, dan metode untuk mendapatkan data dari kolom setiap baris.

Bila metode mengembalikan objek Cursor, Ulangi hasil, ekstrak data, lakukan sesuatu dengan data, dan terakhir tutuplah kursor untuk membebaskan memori.

Anda akan mengetahui selengkapnya tentang kursor dalam bab berikut.

Ketahui selengkapnya

- Situs web SQLite

- Keterangan lengkap bahasa kueri
- Kelas SQLiteDatabase
- Kelas Cursor

10.2: Database SQLite

Materi:

- Menggunakan database SQLite bersama Android
- Kursor
- ContentValues
- Mengimplementasikan database SQLite
- Operasi database
- Buat Instance Open Helper
- Bekerja dengan database
- Transaksi
- Mencadangkan database
- Mengirim database bersama APK Anda
- Praktik terkait
- Ketahui selengkapnya

Bab ini membahas SQLiteDatabase kerangka kerja Android dan kelas SQLiteOpenHelper. Bab ini bukanlah pengantar SQLite atau database SQL. Bab ini beranggapan bahwa Anda sudah familier dengan database SQL secara umum, dan pembangunan kueri SQL dasar. Periksa bab SQL Primer jika Anda memerlukan penyegaran.

Dari banyak opsi storage yang telah dibahas, menggunakan database SQLite adalah salah satu yang paling serbaguna dan praktis untuk diimplementasikan.

- Database SQLite adalah solusi storage yang baik jika Anda memiliki data terstruktur yang perlu diakses dan disimpan secara persisten serta sering ditelusuri dan diubah.
- Anda bisa menggunakan database sebagai storage utama untuk data aplikasi atau pengguna, atau Anda bisa menggunakannya untuk meng-cache serta menyediakan data yang diambil dari awan.
- Jika Anda bisa menyatakan data berupa baris dan kolom, pertimbangkan database SQLite.
- Penyedia materi, yang akan diperkenalkan dalam bab berikutnya, bekerja dengan bagus pada database SQLite.

Jika Anda menggunakan database SQLite, yang dinyatakan sebagai objek SQLiteDatabase, semua interaksi dengan database adalah melalui instance kelas SQLiteOpenHelper yang akan mengeksekusi permintaan dan mengelola database untuk Anda. Aplikasi Anda hanya boleh berinteraksi dengan SQLiteOpenHelper, yang akan dijelaskan di bawah ini.

Ada dua tipe data yang dikaitkan secara khusus dengan penggunaan database SQLite, Cursor dan ContentValues.

Cursor

SQLiteDatabase selalu menyajikan hasil berupa Cursor dalam format tabel yang menyerupai database SQL.

Anda bisa menganggap data sebagai larik baris. Kursor adalah pointer ke dalam satu baris data terstruktur. Kelas Cursor menyediakan metode untuk menggerakkan kursor melalui struktur data, dan metode untuk mendapatkan data dari bidang-bidang di setiap baris.

Kelas Cursor memiliki sejumlah subkelas yang mengimplementasikan kursor untuk tipe data tertentu.

- SQLiteCursor mengekspos hasil kueri dari sebuah SQLiteDatabase. SQLiteCursor tidak disinkronkan secara internal, sehingga kode yang menggunakan SQLiteCursor dari beberapa thread harus melakukan sinkronisasi sendiri saat menggunakan SQLiteCursor.
- MatrixCursor adalah implementasi kursor lengkap dan tidak tetap, yang didukung oleh larik objek yang secara otomatis meluaskan kapasitas internal bila diperlukan.

Beberapa operasi yang umum pada kursor adalah:

- getCount() mengembalikan jumlah baris dalam kursor.

- getColumnNames() mengembalikan larik string yang berisi nama semua kolom dalam rangkaian hasil dalam urutan pencantumannya dalam hasil.
- getPosition() mengembalikan posisi kursor saat ini dalam rangkaian baris.
- Getter tersedia untuk tipe data tertentu, seperti getString(int column) dan getInt(int column).
- Operasi seperti moveToFirst() dan moveToNext() akan menggerakkan kursor.
- close() membebaskan semua sumber daya dan membuat kursor menjadi tidak valid sama sekali. Ingat untuk menutup panggilan guna membebaskan sumber daya!

Memproses kursor

Jika panggilan metode mengembalikan ulangi pada hasil, ekstrak data, lakukan sesuatu dengan data, dan terakhir, harus menutup kursor untuk membebaskan memori. Jika tidak dilakukan, aplikasi Anda bisa mogok saat kehabisan memori.

Kursor dimulai sebelum baris hasil pertama, sehingga pada pengulangan pertama gerakkan kursor ke hasil pertama jika ada. Jika kursor kosong, atau baris terakhir sudah diproses, maka akan keluar dari loop. Jangan lupa menutup kursor bila Anda selesai menggunakannya. (Ini tidak boleh diulang terlalu sering.)

```
// Perform a query and store the result in a Cursor
Cursor cursor = db.rawQuery(...);
try {
    while (cursor.moveToNext()) {
        // Do something with the data
     }
} finally {
    cursor.close();
}
```

Jika menggunakan database SQL, Anda bisa mengimplementasikan kelas SQLiteOpenHelper untuk mengembalikan kursor ke aktivitas pemanggil atau adapter, atau Anda bisa mengonversi data ke format yang lebih cocok untuk adapter. Manfaat dari yang terakhir itu adalah pengelolaan kursor (dan penutupannya) ditangani oleh helper terbuka, dan antarmuka pengguna Anda tidak tergantung pada apa yang terjadi di backend. Lihat praktik Database SQLite untuk contoh implementasi.

ContentValues

Serupa dengan cara ekstra menyimpan data, instance ContentValues menyimpan data sebagai pasangan nilai-kunci, dalam ini kuncinya adalah nama kolom dan nilainya adalah nilai untuk sel. Satu instance ContentValues menyatakan satu baris tabel.

Metode insert() untuk database memerlukan nilai untuk mengisi baris yang diteruskan sebagai instance ContentValues.

```
ContentValues values = new ContentValues();
// Insert one row. Use a loop to insert multiple rows.
values.put(KEY_WORD, "Android");
values.put(KEY_DEFINITION, "Mobile operating system.");

db.insert(WORD_LIST_TABLE, null, values);
```

Mengimplementasikan database SQLite

Untuk mengimplementasikan database aplikasi Android, Anda perlu melakukan yang berikut ini.

1. (Disarankan) Buat model data.
2. Jadikan SQLiteOpenHelper sebagai subkelas
    i. Gunakan konstanta untuk nama tabel dan kueri pembuatan database

   ii. Implementasikan onCreate untuk membuat SQLiteDatabase bersama tabel untuk data Anda
   iii. Implementasikan onUpgrade()
   iv. Implementasikan metode opsional
3. Implementasikan metode query(), insert(), delete(), update(), count() dalam SQLiteOpenHelper.
4. Dalam MainActivity Anda, buat instance SQLiteOpenHelper.
5. Panggil metode SQLiteOpenHelper untuk digunakan bersama database Anda.

Keberatan:

- Bila Anda mengimplementasikan metode, selalu masukkan operasi database ke dalam blok try/catch.
- Aplikasi contoh tidak memvalidasi data pengguna. Bila menulis aplikasi untuk dipublikasikan, selalu pastikan data pengguna sesuai harapan untuk menghindari penyuntikan data buruk atau eksekusi perintah SQL yang berbahaya ke dalam database Anda.

Model data

Praktik yang baik adalah dengan membuat kelas yang menyatakan data Anda dengan getter dan setter.

Untuk database SQLite, instance kelas ini dapat menyatakan satu catatan, dan untuk database sederhana, satu baris dalam tabel.

```
public class WordItem {
    private int mId;
    private String mWord;
    private String mDefinition;
    // Getters and setters and more
}
```

Jadikan SQLiteOpenHelper sebagai subkelas

Open helper apa pun yang Anda buat harus memperluas SQLiteOpenHelper.

```
public class WordListOpenHelper extends SQLiteOpenHelper {

    public WordListOpenHelper(Context context) {
       super(context, DATABASE_NAME, null, DATABASE_VERSION);
       Log.d(TAG, "Construct WordListOpenHelper");
    }
}
```

Definisikan konstanta untuk nama tabel

Walaupun tidak diwajibkan, sudah biasa mendeklarasikan nama tabel, kolom, dan baris sebagai konstanta. Hal ini akan membuat kode lebih terbaca, lebih mudah mengubahi nama, dan kueri akan terlihat lebih mirip SQL. Anda bisa melakukannya dalam kelas open helper, atau dalam kelas publik tersendiri; Anda akan mengetahui selengkapnya tentang hal ini dalam bab tentang penyedia materi.

```
    private static final int DATABASE_VERSION = 1;
  // has to be 1 first time or app will crash
  private static final String WORD_LIST_TABLE = "word_entries";
  private static final String DATABASE_NAME = "wordlist";

  // Column names...
  public static final String KEY_ID = "_id";
  public static final String KEY_WORD = "word";

  // ... and a string array of columns.
  private static final String[] COLUMNS = {KEY_ID, KEY_WORD};
```

Definisikan kueri untuk membuat database

Anda memerlukan kueri yang membuat tabel untuk membuat database. Ini biasanya juga didefinisikan sebagai konstanta string. Contoh dasar ini membuat satu tabel dengan satu kolom untuk ID bertambah-otomatis dan kolom untuk menampung kata.

```
    private static final String WORD_LIST_TABLE_CREATE =
            "CREATE TABLE " + WORD_LIST_TABLE + " (" +
             KEY_ID + " INTEGER PRIMARY KEY, " +
             // will auto-increment if no value passed
             KEY_WORD + " TEXT );";
```

Implementasikan onCreate() dan buat database

Metode onCreate hanya dipanggil jika tidak ada database. Buat tabel Anda dalam metode, dan boleh menambahkan data awal.

```
@Override
public void onCreate(SQLiteDatabase db) { // Creates new database
   db.execSQL(WORD_LIST_TABLE_CREATE); // Create the tables
   fillDatabaseWithData(db); // Add initial data
   // Cannot initialize mWritableDB and mReadableDB here, because
   // this creates an infinite loop of on Create()
   // being repeatedly called.
}
```

Implementasikan onUpgrade()

Ini adalah metode yang diperlukan.

Jika database hanya berfungsi sebagai cache untuk data yang juga disimpan online, Anda bisa menghapus tabel dan membuat ulang setelah peningkatan versi selesai.

Catatan: Jika database adalah storage utama, Anda **harus** mengamankan data pengguna sebelum melakukannya karena operasi ini akan memusnahkan semua data. Lihat bab mengenai Menyimpan Data.

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // SAVE USER DATA FIRST!!!
        Log.w(WordListOpenHelper.class.getName(),
                "Upgrading database from version " + oldVersion + " to "
                        + newVersion + ", which will destroy all old data");
        db.execSQL("DROP TABLE IF EXISTS " + WORD_LIST_TABLE);
        onCreate(db);
}
```

Metode opsional

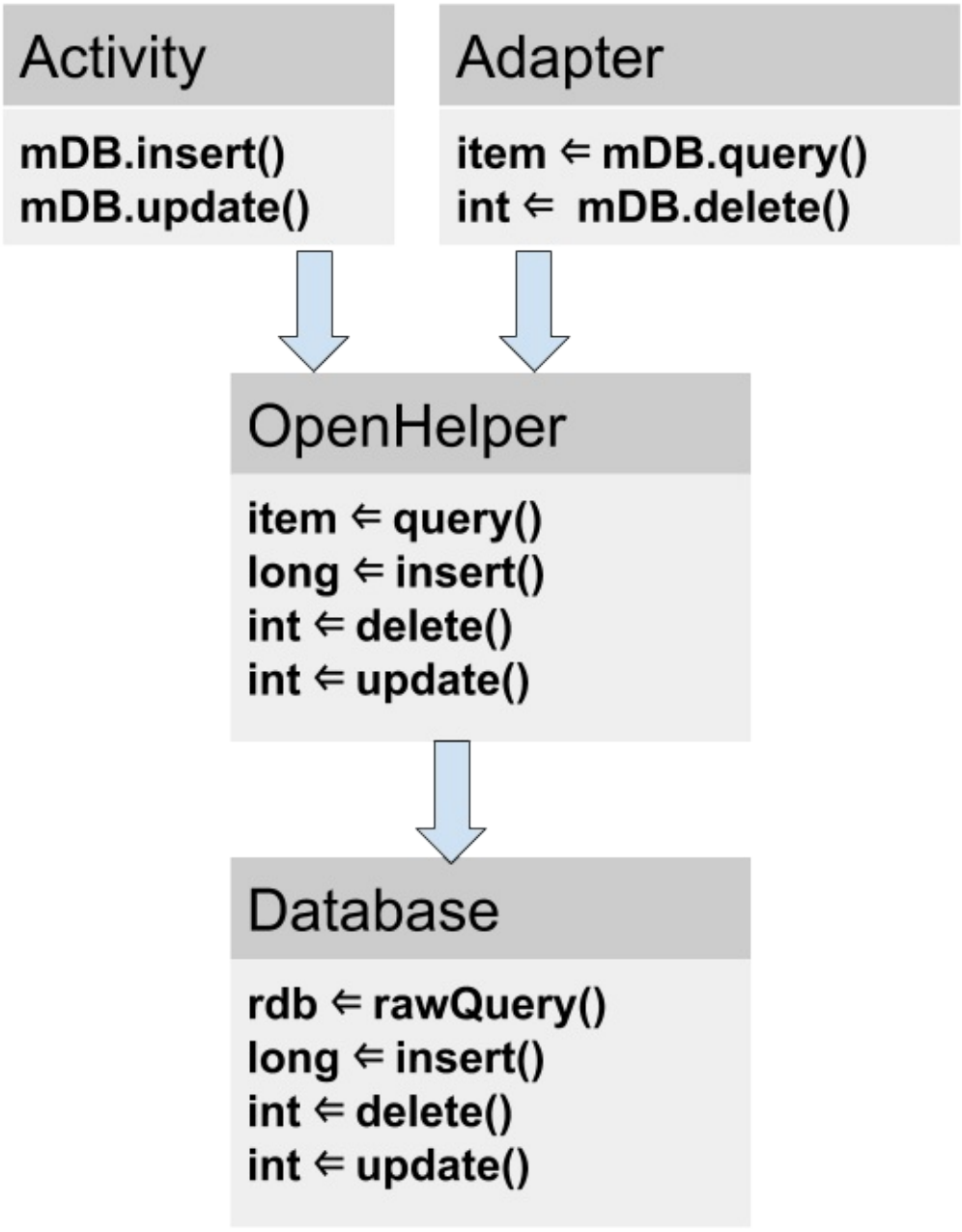
Kelas open helper menyediakan metode tambahan yang bisa Anda ganti bila diperlukan.

- onDowngrade()—Implementasi default menolak penurunan versi.
- onConfigure()—dipanggil sebelum onCreate. Gunakan ini hanya untuk memanggil metode yang mengonfigurasi parameter koneksi database.
- onOpen()—Pekerjaan apa pun selain konfigurasi yang harus dilakukan sebelum database dibuka.

Operasi database

Walaupun bisa memanggil metode dalam open helper yang diinginkan dan mengembalikan yang Anda pilih ke aktivitas pemanggil, lebih baik lanjutkan dengan metode query(), insert(), delete(), update(), count() standar yang sesuai dengan API database dan penyedia materi. Menggunakan format ini akan mempermudah penambahan penyedia materi atau pemuat di masa mendatang, dan memudahkan orang lain dalam memahami kode Anda.

Diagram berikut menampilkan cara mendesain API yang berbeda agar konsisten dan jelas.



query()

Metode kueri yang diimplementasikan dalam kelas open helper Anda bisa mengambil dan mengembalikan tipe data apa pun yang diperlukan antarmuka pengguna.

Karena open helper menyediakan metode praktir untuk menyisipkan, menghapus, dan memperbarui baris, metode kueri Anda tidak perlu generik dan mendukung operasi ini.

Secara umum, metode kueri Anda hanya boleh mengizinkan kueri yang diperlukan oleh aplikasi dan bukan untuk keperluan umum.

Database menyediakan dua metode untuk mengirimkan kueri: SQLiteDatabase.rawQuery() dan SQLiteDatabase.query(), bersama sejumlah opsi untuk argumen.

SQLiteDatabase.rawQuery().rawQuery()

Metode kueri open helper bisa membentuk kueri SQL dan mengirimkannya sebagai rawQuery ke database yang mengembalikan kursor. Jika data disediakan oleh aplikasi dan dikontrol penuh, Anda bisa menggunakan rawQuery().

```
rawQuery(String sql, String[] selectionArgs)
```

- Parameter pertama untuk db.rawquery() adalah string kueri SQLite.
- Parameter kedua berisi argumen.

```
cursor = mReadableDB.rawQuery(queryString, selectionArgs);
```

SQLiteDatabase.query()

Jika Anda memproses data yang disediakan pengguna, bahkan setelah validasi, lebih aman membentuk kueri dan menggunakan versi metode SQLiteDatabase.query() untuk database. Argumen adalah apa yang Anda harapkan dalam SQL dan didokumentasikan dalam dokumentasi SQLiteDatabase.

```
Cursor query (boolean distinct,  String table, String[] columns, String selection,
                String[] selectionArgs, String groupBy, String having,
                String orderBy,String limit)
```

Inilah contoh dasarnya:

```
String[] columns = new String[]{KEY_WORD};
String where =  KEY_WORD + " LIKE ?";
searchString = "%" + searchString + "%";
String[] whereArgs = new String[]{searchString};
cursor = mReadableDB.query(WORD_LIST_TABLE, columns, where, whereArgs, null, null, null);
```

Contoh lengkap open helper query()

```
public WordItem query(int position) {
   String query = "SELECT  * FROM " + WORD_LIST_TABLE +
           " ORDER BY " + KEY_WORD + " ASC " +
           "LIMIT " + position + ",1";

   Cursor cursor = null;
   WordItem entry = new WordItem();

   try {
       if (mReadableDB == null) {mReadableDB = getReadableDatabase();}
       cursor = mReadableDB.rawQuery(query, null);
       cursor.moveToFirst();
       entry.setId(cursor.getInt(cursor.getColumnIndex(KEY_ID)));
       entry.setWord(cursor.getString(cursor.getColumnIndex(KEY_WORD)));
   } catch (Exception e) {
       Log.d(TAG, "EXCEPTION! " + e);
   } finally {
       // Must close cursor and db now that we are done with it.
       cursor.close();
       return entry;
   }
}
```

insert()

Metode insert() open helper memanggil SQLiteDatabase.insert(), yaitu metode SQLiteDatabase praktis yang digunakan untuk menyisipkan baris ke dalam database. (Metode ini praktis, karena Anda tidak perlu menulis kueri SQL sendiri.)

Format

```
long insert(String table, String nullColumnHack, ContentValues values)
```

- Argumen pertama adalah nama tabel.
- Argumen kedua adalah sebuah `String nullColumnHack`. Ini adalah solusi yang memungkinkan Anda untuk menyisipkan baris kosong. Lihat dokumentasi untuk insert(). Gunakan null.
- Argumen ketiga harus berupa kontainer [ContentValues] bersama nilai-nilai untuk mengisi baris. Contoh ini hanya memiliki satu kolom; untuk tabel dengan banyak kolom, tambahkan
- Metode database mengembalikan ID item yang baru disisipkan, dan Anda harus meneruskannya ke aplikasi.

Contoh

```
newId = mWritableDB.insert(WORD_LIST_TABLE, null, values);
```

delete()

Metode delete dari open helper memanggil metode delete() database, yang praktis digunakan sehingga Anda tidak perlu menulis kueri SQL seluruhnya.

Format

```
int delete (String table, String whereClause, String[] whereArgs)
```

- Argumen pertama adalah nama tabel.
- Argumen kedua adalah sebuah klausa WHERE.
- Argumen ketiga adalah argumen untuk klausa WHERE.

Anda bisa menghapus menggunakan kriteria apa pun, dan metode akan mengembalikan jumlah item yang sebenarnya dihapus, yang juga harus dikembalikan oleh open helper.

Contoh

```
deleted = mWritableDB.delete(WORD_LIST_TABLE,
                    KEY_ID + " =? ", new String[]{String.valueOf(id)});
```

update()

Metode update dari open helper memanggil metode update() database, yang praktis digunakan sehingga Anda tidak perlu menulis kueri SQL seluruhnya. Argumen tersebut sudah familier dari metode sebelumnya, dan onUpdate mengembalikan jumlah baris yang diperbarui.

Format

```
int update(String table, ContentValues values,
    String whereClause, String[] whereArgs)
```

- Argumen pertama adalah nama tabel.
- Argumen kedua harus berupa ContentValues bersama nilai-nilai baru untuk baris tersebut.
- Argumen ketiga adalah klausa WHERE.
- Argumen keempat adalah argumen untuk klausa WHERE.

Contoh

```
ContentValues values = new ContentValues();
values.put(KEY_WORD, word);
mNumberOfRowsUpdated = mWritableDB.update(WORD_LIST_TABLE,
values, // new values to insert
KEY_ID + " = ?",
new String[]{String.valueOf(id)});
```

count()

Metode count() mengembalikan jumlah entri dalam database. Jika menggunakan RecyclerView.Adapter, Anda harus mengimplementasikan getItemCount(), yang perlu mendapatkan sejumlah baris dari open helper, yang perlu didapat dari database.

Di adapter

```
@Override
public int getItemCount() {
    return (int) mDB.count();
}
```

Di open helper

```
public long count(){
    if (mReadableDB == null) {mReadableDB = getReadableDatabase();}
    return DatabaseUtils.queryNumEntries(mReadableDB, WORD_LIST_TABLE);
}
```

queryNumEntries()) adalah metode di kelas DatabaseUtils publik, yang menyediakan banyak metode praktis untuk digunakan bersama kursor, database, dan juga penyedia materi.

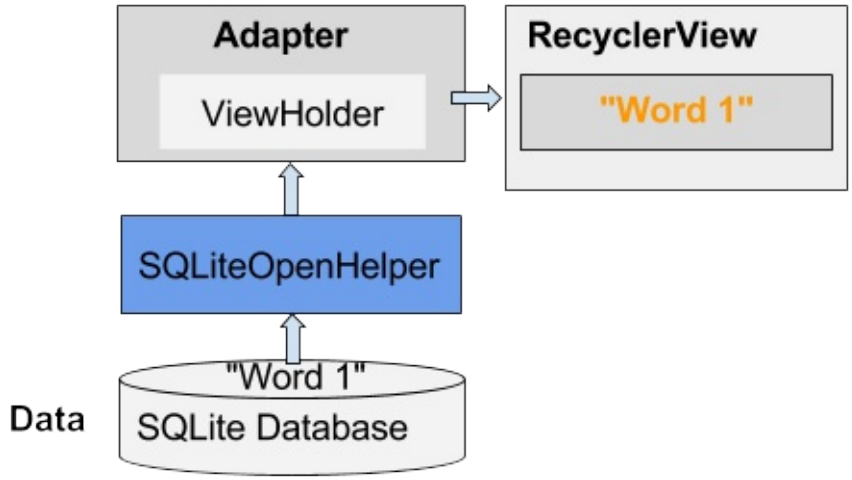
Buat Instance Open Helper

Untuk menangani database, dalam MainActivity, di onCreate, panggil:

```
mDB = new WordListOpenHelper(this);
```

Bekerja dengan database

Mengombinasikan backend SQLiteDatabase dengan RecyclerView untuk menampilkan data merupakan pola yang umum.



Misalnya:

- Menekan FAB bisa memulai aktivitas yang akan mendapatkan masukan dari pengguna dan menyimpannya ke dalam database sebagai item baru atau diperbarui.
- Menggesek suatu item bisa menghapusnya setelah pengguna mengonfirmasi penghapusan.

Transaksi

Gunakan transaksi

- saat melakukan beberapa operasi yang semuanya harus diselesaikan agar database konsisten, misalnya, memperbarui penetapan harga item terkait untuk kejadian penjualan.
- untuk batch beberapa operasi independen guna meningkatkan kinerja, seperti penyisipan massal.

Transaksi bisa disarangkan, dan kelas SQLiteDatabase menyediakan metode tambahan untuk mengelola transaksi tersarang. Lihat dokumentasi referensi SQLiteDatabase.

Idiom transaksi

```
db.beginTransaction();
try {
  ...
  db.setTransactionSuccessful();
} finally {
  db.endTransaction();
}
```

Mencadangkan database

Ada baiknya Anda mencadangkan database aplikasi.

Anda bisa melakukannya menggunakan opsi Cloud Backup yang dibahas dalam bab Opsi Storage.

Mengirim database bersama aplikasi Anda

Kadang-kadang Anda mungkin ingin menyertakan database yang sudah terisi bersama aplikasi. Ada sejumlah cara untuk melakukannya, dan konsekuensi dari setiap cara tersebut.

- Sertakan perintah SQL bersama aplikasi dan perintahkan untuk membuat database dan menyisipkan data pada penggunaan pertama. Inilah yang pada dasarnya akan Anda lakukan dalam praktik storage data. Jika jumlah data yang ingin dimasukkan dalam database kecil, hanya satu contoh agar pengguna bisa melihat sesuatu, Anda bisa menggunakan metode ini.
- Kirimkan data bersama APK sebagai sumber daya, dan bangun database saat pengguna membuka aplikasi untuk pertama kali. Metode ini sama dengan metode pertama, namun sebagai ganti mendefinisikan data dalam kode, Anda memasukkannya dalam sumber daya, misalnya, dalam format CSV. Selanjutnya Anda bisa membaca data dengan aliran masukan dan menambahkannya ke database.
- Bangun dan isi dahulu database SQLite lalu sertakan dalam APK. Dengan metode ini, tulis aplikasi yang akan membuat dan mengisi database. Anda bisa melakukan ini pada emulator. Selanjutnya Anda bisa menyalin file tersebut di tempat penyimpanan database sebenarnya (direktori "/data/data/YOUR_PACKAGE/databases/") dan menyertakannya sebagai aset bersama aplikasi. Saat aplikasi dimulai untuk pertama kali, salin kembali file database ke dalam direktori "/data/data/YOUR_PACKAGE/databases/".

Kelas SQLiteAssetHelper, yang bisa Anda unduh dari Github, memperluas SQLiteOpenHelper untuk membantu Anda melakukannya. Dan entri blog Stackoverflow akan membahas topik ini lebih detail.

Perhatikan, untuk database yang lebih besar, mengisi database harus dilakukan di latar belakang, dan aplikasi Anda tidak akan mogok jika belum ada database, atau database masih kosong.

Praktik terkait

Dokumentasi praktik terkait ada di Dasar-Dasar Developer Android: Praktik.

- Storage Data SQLite
- Menelusuri Database SQLite

Ketahui selengkapnya

- Opsi Storage
- Menyimpan Data di Database SQL
- Kelas SQLiteDatabase
- Kelas ContentValues
- Kelas SQLiteOpenHelper
- Kelas Cursor
- Kelas SQLiteAssetHelper dari Github

11.1: Bagikan Data Melalui Penyedia Materi

Materi:

- Apa yang dimaksud dengan Penyedia Materi?
- Apa yang dimaksud dengan Resolver Materi?
- Contoh berbagi data aplikasi menggunakan Penyedia Materi
- Apa kegunaan Penyedia Materi
- Arsitektur Aplikasi dengan Penyedia Materi
- Mengimplementasikan Penyedia Materi
- Data
- Kontrak
- Metode: insert, delete, update, query
- Metode query()
- Menggunakan Resolver Materi
- Izin untuk berbagi
- Praktik terkait
- Ketahui selengkapnya

Apa yang dimaksud dengan Penyedia Materi?

ContentProvider adalah komponen yang berinteraksi dengan repositori. Aplikasi ini tidak perlu tahu di mana atau bagaimana data disimpan, diformat, atau diakses.

Penyedia materi:

- Memisahkan data dari kode antarmuka aplikasi
- Menyediakan cara standar mengakses data
- Memungkinkan aplikasi berbagi data dengan aplikasi lainnya
- Agnostik terhadap repositori, yang mungkin melalui database, sistem file, atau awan.

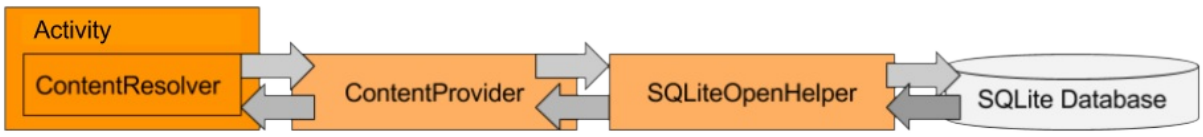
Apa yang dimaksud dengan Resolver Materi?

Untuk mendapatkan data dan berinteraksi dengan penyedia materi, aplikasi menggunakan ContentResolver untuk mengirimkan permintaan ke penyedia materi.

Objek ContentResolver menyediakan metode query(), insert(), update(), dan delete() untuk mengakses data dari penyedia materi.

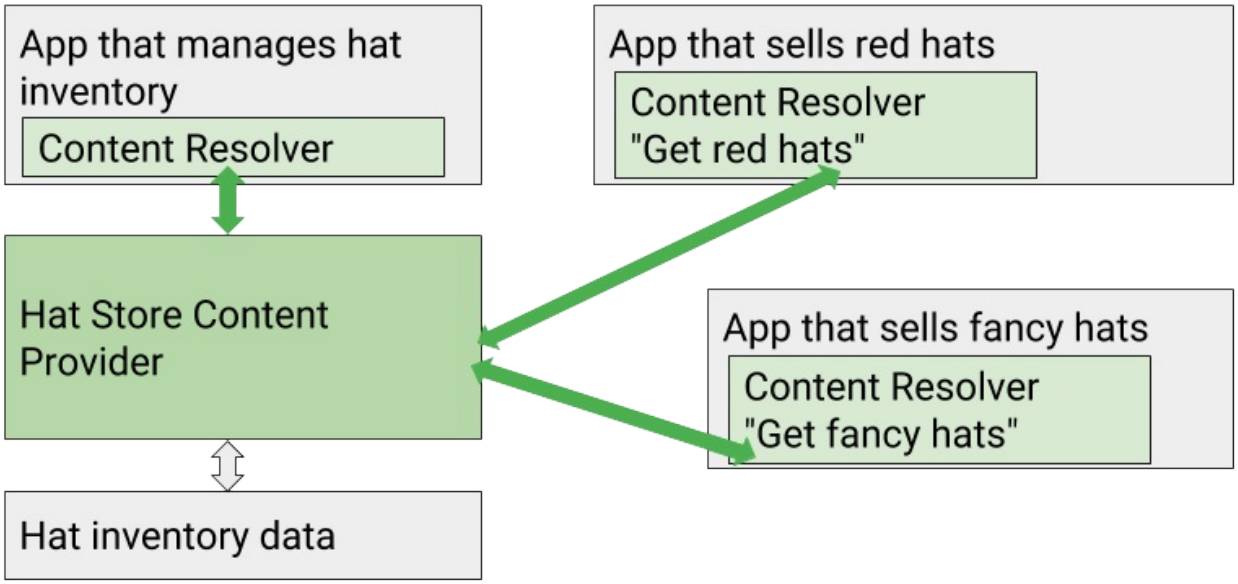
Masing-masing permintaan terdiri dari URI dan kueri mirip-SQL, dan responsnya adalah objek Cursor.

Catatan: Anda telah mempelajari tentang kursor dalam bab Storage Data, dan nanti akan ada rangkuman dalam bab ini. Diagram berikut menampilkan alur kueri dari aktivitas dengan menggunakan resolver materi, ke penyedia materi, ke data dalam database SQL, dan kembali lagi. Perhatikan, penyimpanan data umumnya dalam database SQLite, namun tidak wajib.



Contoh berbagi data aplikasi menggunakan Penyedia Materi

Anggaplah seperti aplikasi yang menyimpan persediaan topi dan menyediakannya untuk aplikasi lain yang ingin menjual topi. Aplikasi yang memiliki data mengelola persediaan, namun tidak memiliki antarmuka yang berinteraksi dengan pelanggan. Dua aplikasi, yang satu menjual topi merah dan yang satu lagi menjual topi hias, mengakses repositori persediaan, dan masing-masing mengambil data yang relevan untuk aplikasi belanja mereka.



Apa kegunaan Penyedia Materi

Penyedia materi berguna untuk aplikasi yang ingin menyediakan data bagi aplikasi lain.

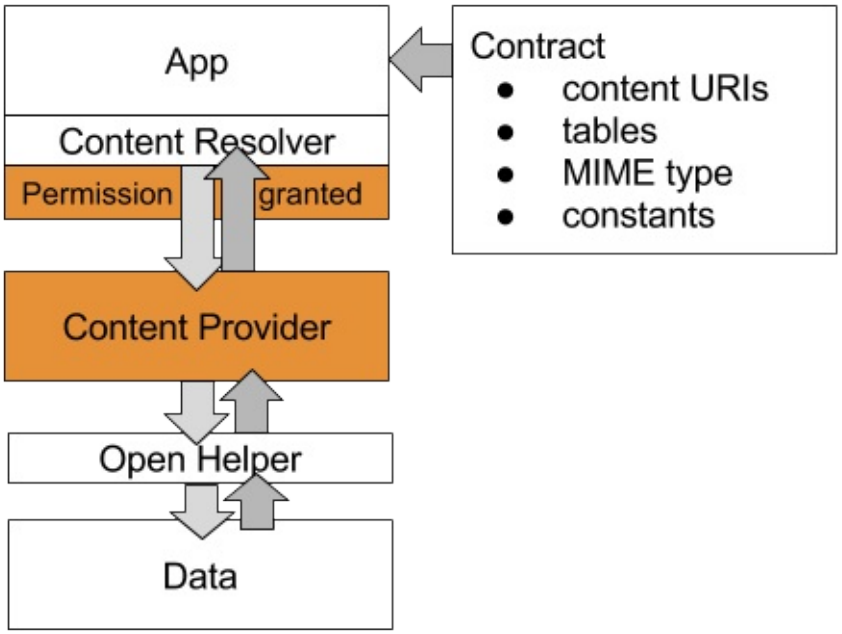
- Dengan penyedia materi, Anda bisa memungkinkan beberapa aplikasi lainnya untuk mengakses, menggunakan, dan memodifikasi sumber data tunggal yang disediakan oleh aplikasi Anda. Contoh: Persediaan di gudang untuk toko retail, skor game, atau kumpulan masalah fisika untuk universitas.
- Untuk kontrol akses, Anda bisa menetapkan tingkatan izin penyedia materi, yang menetapkan cara aplikasi lain bisa mengakses data. Misalnya, toko mungkin tidak boleh mengubah data persediaan di gudang.
- Anda bisa menyimpan data secara independen dari aplikasi, karena penyedia materi berada di antara antarmuka pengguna dan data yang disimpan. Anda bisa mengubah cara data disimpan tanpa perlu mengubah kode yang berinteraksi dengan pengguna. Misalnya, Anda bisa membangun prototipe aplikasi belanja menggunakan data persediaan tiruan, kemudian menggantinya nanti dengan database SQL untuk data sungguhan. Anda bahkan bisa menyimpan sebagian data di sistem awan serta sebagian lagi secara lokal, dan semua itu sama saja bagi pengguna.
- Manfaat lain memisahkan data dari antarmuka pengguna dengan penyedia materi adalah karena tim development bisa bekerja secara independen pada antarmuka pengguna dan repositori data aplikasi Anda. Untuk aplikasi yang lebih besar dan kompleks, umumnya antarmuka pengguna dan backend data dikembangkan oleh tim yang berbeda, bahkan bisa berupa aplikasi terpisah; yakni, aplikasi dengan penyedia materi tidak harus memiliki antarmuka pengguna. Misalnya, aplikasi persediaan bisa terdiri dari data dan penyedia materi saja.
- Ada kelas lain yang diharapkan berinteraksi dengan penyedia materi. Misalnya, Anda harus memiliki penyedia materi untuk menggunakan loader, seperti CursorLoader, untuk memuat data di latar belakang. Anda akan mempelajari tentang loader di bab berikutnya.

Catatan: Jika hanya aplikasi tersebut yang menggunakan data, dan Anda mengembangkannya sendiri, Anda mungkin tidak memerlukan penyedia materi.

Arsitektur Aplikasi dengan Penyedia Materi

Secara arsitektur, penyedia materi adalah layer di antara backend storage data aplikasi penyedia materi dan bagian aplikasi lainnya, yang memisahkan data dan antarmuka.

Untuk memberi Anda gambaran arsitektur penyedia materi secara menyeluruh, bagian ini menampilkan dan merangkum semua bagian arsitektur penyedia materi yang diimplementasikan, seperti yang ditampilkan dalam diagram berikut. Setiap bagian nanti akan dibahas secara detail.



Data dan Open Helper: Repositori data. Data bisa berada dalam database, file, internet, dihasilkan secara dinamis, atau bahkan campuran semua ini. Misalnya, jika Anda memiliki aplikasi kamus, kamus dasar bisa disimpan dalam database SQLite pada perangkat pengguna. Jika tidak ada dalam database, definisi bisa diambil dari internet, dan jika gagal, aplikasi bisa meminta pengguna untuk menyediakan definisi atau memeriksa ejaannya.

Data yang digunakan bersama penyedia materi umumnya disimpan dalam database SQLite, dan API penyedia materi akan mencerminkan asumsi ini.

Kontrak: Kontrak adalah kelas publik yang mengekspos informasi penting tentang penyedia materi ke aplikasi lainnya. Kontrak biasanya menyertakan skema URI, konstanta penting, dan struktur data yang akan dikembalikan. Misalnya, untuk aplikasi persediaan topi, kontrak bisa mengekspos nama kolom yang berisi harga serta nama produk, dan URL untuk mendapatkan item persediaan berdasarkan nomor komponen.

Penyedia Materi: Penyedia materi memperluas kelas ContentProvider dan menyediakan metode query(), insert(), update(), serta delete() untuk mengakses data. Selain itu, penyedia materi menyediakan antarmuka publik yang aman ke data, sehingga aplikasi lainnya bisa mengakses data dengan izin yang sesuai. Misalnya, untuk mendapatkan informasi dari database aplikasi Anda, aplikasi topi retail akan menghubungkan dengan penyedia materi, bukan dengan database secara langsung, karena ini tidak diizinkan.

Aplikasi yang memiliki data akan menetapkan izin apa yang diperlukan aplikasi lainnya untuk bekerja dengan penyedia materi. Misalnya, jika memiliki aplikasi yang menyediakan persediaan untuk toko retail, aplikasi Anda memiliki data dan menentukan izin akses aplikasi lainnya ke data. Izin ditetapkan dalam Manifes Android.

Resolver Materi: Penyedia materi selalu diakses melalui resolver materi. Anggaplah resolver materi sebagai kelas helper yang mengelola semua detail hubungan dengan penyedia materi untuk Anda. Pencerminan API penyedia materi, objek ContentResolver memberi Anda metode query(), insert(), update(), dan delete() untuk mengakses data penyedia materi. Misalnya, untuk mendapatkan semua item persediaan berupa topi merah, aplikasi toko merah akan membangun kueri untuk topi merah, dan menggunakan resolver materi untuk mengirim kueri itu ke penyedia materi.

Mengimplementasikan Penyedia Materi

Dengan merujuk diagram sebelumnya, untuk mengimplementasikan penyedia materi, Anda perlu:

- Data, misalnya, dalam sebuah database.

- Cara untuk mengakses storage data, misalnya, melalui open helper untuk database.
- Deklarasi penyedia materi Anda dalam Manifes Android untuk menyediakannya bagi aplikasi Anda sendiri dan aplikasi lainnya.
- Subkelas ContentProvider yang mengimplementasikan metode query(), insert(), delete(), update(), count(), dan getType().
- Kelas kontrak publik yang mengekspos skema URI, nama tabel, tipe MIME, dan konstanta penting ke kelas dan aplikasi lainnya. Walaupun tidak wajib, tanpa kelas ini, aplikasi lainnya tidak bisa mengetahui cara mengakses penyedia materi Anda.
- Resolver materi untuk mengakses penyedia materi dengan menggunakan metode dan kueri yang sesuai.

Mari kita lihat setiap komponen ini.

Data

Data sering kali disimpan dalam database SQLite, namun ini tidak wajib. Data bisa disimpan dalam file atau sistem file, di internet, atau dibuat secara dinamis. Atau bahkan campuran berbagai opsi ini. Bagi aplikasi, resolver materi, data yang diambil selalu berupa objek Cursor, seolah-olah berasal dari sumber yang sama dan dalam format yang sama.

Penyedia materi bisa mengakses data secara langsung jika berupa file, atau melakukannya melalui kelas helper. Misalnya, umumnya aplikasi menggunakan open helper untuk berinteraksi dengan database SQLite, dan penyedia materi berinteraksi dengan open helper untuk mendapatkan data.

Umumnya, data disajikan kepada penyedia materi oleh penyimpan data berupa tabel, mirip seperti tabel database, dengan masing-masing baris menyatakan satu entri dan setiap kolom menyatakan atribut untuk entri itu. Misalnya, setiap baris berisi satu kontak, dan mungkin memiliki kolom untuk alamat email dan nomor telepon. Struktur tabel akan diekspos dalam kontrak.

Catatan: Jika hanya bekerja dengan file, Anda bisa menggunakan kelas FileProvider yang telah didefinisikan sebelumnya.

Kontrak

Kontrak adalah kelas publik yang mengekspos informasi penting tentang penyedia materi aplikasi sehingga aplikasi lain tahu cara mengakses dan menggunakan penyedia materi.

Menggunakan kontrak akan memisahkan informasi aplikasi yang bersifat publik dari privat, desain dari implementasi, dan memberi aplikasi lain satu tempat untuk mendapatkan semua informasi yang mereka perlukan untuk bekerja dengan penyedia materi. Walaupun aplikasi yang mendasarinya mungkin berubah, kontrak mendefinisikan API yang idealnya tidak berubah setelah aplikasi dipublikasikan.

Kontrak untuk penyedia materi umumnya meliputi:

- **URI materi dan skema URI.** Skema URI menampilkan cara membangun URI untuk mengakses data penyedia materi. Ini adalah API untuk data.
- **Konstanta tabel.** Menyediakan tabel dan nama kolom sebagai konstanta, karena keduanya dibutuhkan untuk mengekstrak data dari objek kursor yang dikembalikan.
- **Tipe MIME**, yang memiliki informasi mengenai format data, sehingga aplikasi bisa memproses data yang dikembalikan dengan semestinya. Misalnya, data bisa dienkode dalam JSON atau HTML, atau menggunakan format khusus.
- Konstanta bersama lainnya yang membuat aplikasi lebih praktis menggunakan penyedia materi.

Catatan: Kontrak tidak terbatas untuk penyedia materi. Anda bisa menggunakan kontrak kapan saja diinginkan untuk berbagi konstanta dengan kelas-kelas aplikasi atau menyediakan informasi tentang aplikasi Anda kepada aplikasi lainnya.

Skema URI & URI Materi

Aplikasi mengirim permintaan ke penyedia materi menggunakan Uniform Resource Identifiers atau URI.

URI materi untuk penyedia materi memiliki bentuk umum ini:

```
scheme://authority/path/ID
```

- **scheme** selalu **content://** untuk URI materi.
- **authority** menyatakan domain, dan untuk penyedia materi biasanya berakhiran `.provider`
- **path** adalah jalur menuju data.
- **ID** secara unik mengidentifikasi rangkaian data yang akan ditelusuri.

Misalnya, URI berikut bisa digunakan untuk meminta semua entri di tabel "words":

```
content://com.android.example.wordcontentprovider.provider/words
```

Skema URI untuk mengakses penyedia materi didefinisikan dalam Contract sehingga hanya tersedia bagi aplikasi yang ingin melakukan kueri ke penyedia materi ini. Biasanya, ini dilakukan dengan mendefinisikan konstanta untuk AUTHORITY, CONTENT_PATH, dan CONTENT_URI.

- AUTHORITY. Menyatakan domain. Untuk penyedia materi, ini menyertakan nama paket yang unik dan berakhiran `.provider`
- CONTENT_PATH. Jalur materi adalah identifier semantik abstrak dari data yang diminati. Jalur data tidak memprediksi atau menganggap dalam bentuk apa data disimpan atau diatur di latar belakang. Dengan demikian, jalur bisa diurai ke dalam nama tabel, nama file, atau nama daftar.
- CONTENT_URI. Ini adalah URI bergaya content:// ke satu rangkaian data. Jika memiliki beberapa "kontainer data" di backend, Anda harus membuat satu URI materi untuk masing-masing kontainer. Misalnya, Anda ingin membuat URI materi ke masing-masing tabel yang bisa dikueri. Gunakan kelas helper URI untuk membangun dan memanipulasi URI.

Dalam kode, ini akan terlihat seperti berikut:

```
public static final String AUTHORITY =
"com.android.example.minimalistcontentprovider.provider";

public static final String CONTENT_PATH =  "words";
public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY +
"/" + CONTENT_PATH);
```

Tabel dalam Kontrak

Cara umum untuk mengatur suatu kelas kontrak adalah menempatkan definisi yang bersifat global terhadap database Anda ke dalam level kelas akar. Biasanya, ini adalah nama database.

Buat kelas dalam abstrak statis untuk setiap tabel dengan nama kolom. Kelas dalam ini biasanya mengimplementasikan antarmuka BaseColumns. Dengan mengimplementasikan antarmuka BaseColumns, kelas Anda bisa mewarisi bidang kunci utama bernama _ID yang diharapkan ada oleh beberapa kelas Android, seperti adapter kursor. Hal ini tidak diharuskan, namun bisa membantu database Anda untuk bekerja harmonis dengan kerangka kerja Android.

Kode Anda dalam kontrak akan tampak seperti ini:

```
public static final String DATABASE_NAME = "wordlist";

public static abstract class WordList implements BaseColumns {
    public static final String WORD_LIST_TABLE = "word_entries";
    // Column names...
    public static final String KEY_ID = "_id";
    public static final String KEY_WORD = "word"
}
```

Tipe MIME

Tipe MIME memberi tahu aplikasi, tentang tipe dan format data yang diterima, sehingga bisa memproses data dengan semestinya. Tipe MIME yang umum antara lain `text/html` untuk laman web, dan `application/json`. Jika penyedia materi mengembalikan data dalam salah satu dari kedua format standar itu, Anda harus menggunakan tipe MIME standar. Daftar lengkap tipe standar ini tersedia di situs web IANA MIME Media Types.

Akan tetapi, penyedia materi mungkin akan mengembalikan data khusus untuk aplikasi Anda. Dalam hal itu, Anda akan perlu menetapkan tipe MIME khusus.

Untuk URI materi yang menunjuk ke sebuah baris atau beberapa baris data tabel, dan yang bersifat unik untuk aplikasi Anda, tipe MIME harus dalam format MIME khusus vendor Android. Format umum adalah:

```
type.subtype/provider-specific-part
```

Dengan bagian-bagiannya:

- Bagian type: vnd
- Bagian subtype:
  - Jika pola URI adalah untuk satu baris tunggal: android.cursor.item/
  - Jika pola URI adalah untuk lebih dari satu baris: android.cursor.dir/
- Bagian provider-specific: vnd..
- Anda menyediakan dan .
  - Nilai secara global harus unik. Pilihan bagus untuk adalah nama perusahaan Anda atau beberapa bagian dari nama paket Android aplikasi.
  - Nilai harus unik untuk pola URI yang bersangkutan. Pilihan bagus untuk adalah string yang mengidentifikasi tabel yang terkait dengan URI.

Misalnya, jika otoritas penyedia adalah com.example.app.provider, dan mengekspos tabel bernama "words", maka tipe MIME untuk beberapa baris dalam "words" adalah:

```
vnd.android.cursor.dir/vnd.com.example.provider.words
```

Untuk baris tunggal "words", tipe MIME adalah:

```
vnd.android.cursor.item/vnd.com.example.provider.words
```

Dan Anda harus menetapkannya dalam kontrak sebagai:

```
static final String SINGLE_RECORD_MIME_TYPE =
        "vnd.android.cursor.item/vnd.com.example.provider.words";
static final String MULTIPLE_RECORDS_MIME_TYPE =
        "vnd.android.cursor.item/vnd.com.example.provider.words";
```

Suatu aplikasi bisa memanggil metode getType() dari penyedia materi untuk mengetahui tipe data yang diharapkan. Metode getType() akan terlihat seperti ini:

```
@Override
public String getType(Uri uri) {
   switch (sUriMatcher.match(uri)) {
       case URI_ALL_ITEMS_CODE:
           return MULTIPLE_RECORDS_MIME_TYPE;
       case URI_ONE_ITEM_CODE:
           return SINGLE_RECORD_MIME_TYPE;
       default:
           return null;
   }
}
```

Baca selengkapnya tentang tipe MIME untuk penyedia materi dalam dokumentasi developer Android.

Catatan: Tipe MIME tidak memberi tahu klien cara memproses data. Dengan demikian, tipe MIME khusus hanya menyediakan petunjuk, dan kontrak harus menyediakan informasi tambahan kepada klien mengenai format data yang diharapkan.

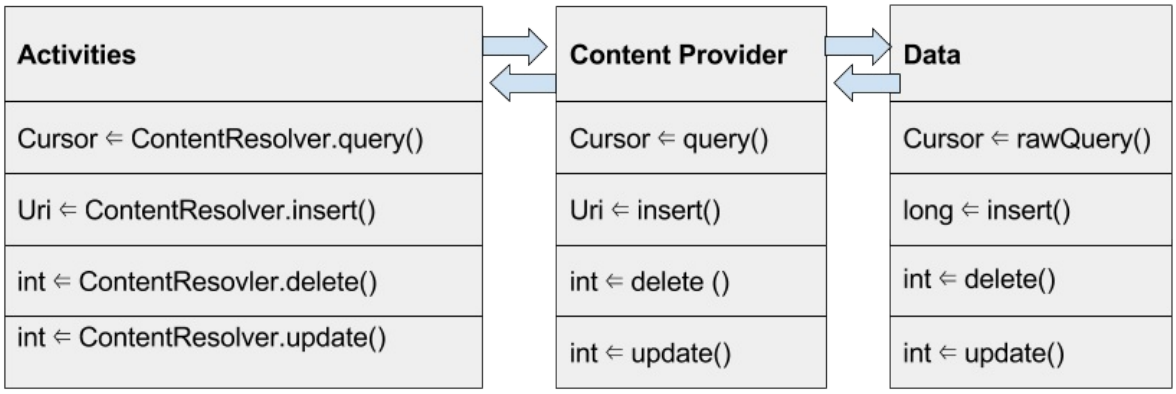
Metode: insert, delete, update, query

Resolver materi menyediakan metode query, insert, delete, serta update, dan penyedia materi mengimplementasikan metode yang sama itu untuk mengakses data.

Dengan demikian, praktik yang baik adalah mempertahankan nama, argumen metode, dan nilai kembalian yang konsisten di antara semua komponen. Hal ini membuat implementasi dan pemeliharaan menjadi jauh lebih mudah.

Diagram berikut menampilkan API di antara blok pembangunan konseptual aplikasi yang menggunakan penyedia materi untuk mengakses data. Terutama perhatikan:

- Metode tersebut diberi nama yang sama dan mengembalikan tipe data yang sama di seluruh tumpukan (kecuali untuk insert()).
- Karena biasanya penyedia materi menghubungkan ke database, metode kueri akan mengembalikan kursor. Jika backend Anda bukan database, penyedia materi harus melakukan pekerjaan mengonversi data yang dikembalikan ke dalam format kursor.
- Diagram ini tidak menampilkan kelas helper tambahan, seperti open helper untuk database, yang mungkin juga menggunakan konvensi API yang sama.



Bila Anda membuat penyedia materi dengan memperluas kelas ContentProvider, Anda perlu mengimplementasikan metode insert, delete, update, dan query. Jika Anda mengikuti prinsip pembuatan tanda tangan metode sama di seluruh komponen, maka penerusan data bolak-balik tidak memerlukan banyak kode.

Inilah metode contoh dalam penyedia materi. Perhatikan, penyedia materi menerima nilai untuk disisipkan dalam tipe yang tepat, seperti ContentValues, memanggil database, dan membangun serta mengembalikan URI yang diperlukan untuk resolver materi.

Metode insert

```
/**
* Inserts one row.
*
* @param uri Uri for insertion.
* @param values Container for Column/Row key/value pairs.
* @return URI for the newly created entry.
*/
@Override
public Uri insert(Uri uri, ContentValues values) {
   long id = mDB.insert(values);
   return Uri.parse(CONTENT_URI + "/" + id);
}
```

Metode delete

```
/**
* Deletes records(s) specified by selectionArgs.
*
* @param uri URI for deletion.
* @param selection Where clause.
* @param selectionArgs Where clause arguments.
* @return Number of records affected.
*/
@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
   return mDB.delete(parseInt(selectionArgs[0]));
}
```

Metode update

```
/**
* Updates records(s) specified by selection/selectionArgs combo.
*
* @param uri URI for update.
* @param values Container for Column/Row key/value pairs.
* @param selection Where clause.
* @param selectionArgs Where clause arguments.
* @return Number of records affected.
*/
@Override
public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
   return mDB.update(parseInt(selectionArgs[0]), values.getAsString("word"));
}
```

Metode query()

Metode query dalam penyedia materi memiliki tanda tangan berikut:

```
public Cursor query(Uri uri, String[] projection, String selection,String[] selectionArgs, String sortOrder){}
```

Argumen menyatakan bagian-bagian kueri SQL dan akan dibahas di bawah ini. Metode kueri penyedia materi harus mem-parse argumen URI dan menentukan aksi yang sesuai.

Pencocokan URI

Praktik yang baik adalah menggunakan instance kelas UriMatcher untuk mencocokkan URI. UriMatcher adalah kelas helper untuk mencocokkan URI bagi penyedia materi.

1. Buat UriMatcher baru dalam penyedia materi Anda.

   ```
   private static UriMatcher sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
   ```

2. Dalam onCreate() tambahkan URI yang akan dicocokkan dengan matcher. Ini adalah URI materi yang didefinisikan dalam Contract. Anda mungkin perlu melakukannya dalam metode tersendiri, initializeUriMatching() yang Anda panggil dalam onCreate(). Kode contoh menyertakan URI yang meminta semua item, satu item berdasarkan ID, dan hitungan item.

```
/**
* Defines the accepted Uri schemes for this content provider.
* Calls addURI() for all of the content URI patterns that the provider should recognize.
*/
private void initializeUriMatching() {

   // Matches a URI that is just the authority + the path,
   // triggering the return of all words.
   sUriMatcher.addURI(AUTHORITY, CONTENT_PATH, URI_ALL_ITEMS_CODE);

   // Matches a URI that references one word in the list by its index.
   sUriMatcher.addURI(AUTHORITY, CONTENT_PATH + "/#", URI_ONE_ITEM_CODE);

   // Matches a URI that returns the number of rows in the table.
   sUriMatcher.addURI(AUTHORITY, CONTENT_PATH + "/" + COUNT, URI_COUNT_CODE);
}
```

Metode query mengaktifkan URI pencocokan untuk melakukan kueri database bagi semua item, satu item, atau satu hitungan item, seperti yang ditampilkan dalam kode contoh ini.

```
@Override
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
                    String sortOrder) {

   Cursor cursor = null;

   // Determine integer code from the URI matcher and switch on it.
   switch (sUriMatcher.match(uri)) {
       case URI_ALL_ITEMS_CODE:
           cursor =  mDB.query(ALL_ITEMS);
           Log.d(TAG, "case all items " + cursor);
           break;
       case URI_ONE_ITEM_CODE:
           cursor =  mDB.query(parseInt(uri.getLastPathSegment()));
           Log.d(TAG, "case one item " + cursor);
           break;
       case URI_COUNT_CODE:
           cursor = mDB.count();
           Log.d(TAG, "case count " + cursor);
           break;
       case UriMatcher.NO_MATCH:
           // You should do some error handling here.
           Log.d(TAG, "NO MATCH FOR THIS URI IN SCHEME: " + uri);
           break;
       default:
           // You should do some error handling here.
           Log.d(TAG, "INVALID URI - URI NOT RECOGNIZED: "  + uri);
   }
   return cursor;
}
```

Menggunakan Resolver Materi

Objek ContentResolver menyediakan metode untuk data query(), insert(), delete(), dan update(). Karena itu, resolver materi mencerminkan API penyedia materi dan mengelola semua interaksi dengan penyedia materi untuk Anda. Dalam kebanyakan situasi, Anda bisa menggunakan penyedia materi default yang disediakan oleh sistem Android.

Kursor

Penyedia materi selalu menyajikan hasil kueri sebagai Cursor dalam format tabel yang menyerupai database SQL. Hal ini tidak dipengaruhi oleh cara penyimpanan data sebenarnya.

Kursor adalah pointer ke dalam baris data terstruktur. Anda bisa menganggapnya sebagai daftar baris yang ditautkan. Kelas Cursor menyediakan metode untuk menggerakkan kursor melalui struktur itu, dan metode untuk mendapatkan data dari kolom setiap baris.

Bila suatu metode mengembalikan Cursor, ulangi hasilnya, ekstrak data, lakukan sesuatu dengan data, dan terakhir tutup kursor untuk membebaskan memori.

Jika Anda menggunakan database SQL, seperti yang ditampilkan di atas, Anda bisa mengimplementasikan open helper untuk mengembalikan kursor, kemudian sekali lagi, penyedia materi mengembalikan kursor melalui resolver materi. Jika storage data mengembalikan data dalam format berbeda, Anda nanti harus mengonversinya menjadi kursor, biasanya MatrixCursor.

Metode query()

Untuk membuat kueri ke penyedia materi:

1. Buat sebuah kueri bergaya SQL.
2. Gunakan resolver materi untuk berinteraksi dengan penyedia materi guna mengeksekusi kueri dan mengembalikan sebuah Cursor.
3. Proses hasilnya di Cursor.

Metode kueri memiliki tanda tangan berikut:

```
public Cursor query(Uri uri, String[] projection, String selection,String[] selectionArgs, String sortOrder){}
```

Argumen untuk metode ini menyatakan bagian-bagian kueri SQL. Sekalipun menggunakan backend jenis lain, Anda tetap harus menerima kueri dalam gaya ini dan menangani argumen dengan semestinya.

uri	URI materi lengkap yang dikueri. Tidak boleh nol. Dapatkan informasi untuk URI yang benar dari kontrak. Misalnya: `String queryUri = Contract.CONTENT_URI.toString();`
projection	Sebuah larik string dengan nama-nama kolom yang akan dikembalikan untuk setiap baris. Menyetelnya ke nol akan mengembalikan semua kolom. Misalnya: `String[] projection = new String[] {Contract.CONTENT_PATH};`
selection	Menunjukkan baris/catatan objek mana yang ingin Anda akses. Ini adalah klausa WHERE yang mengecualikan where yang sebenarnya. Misalnya: `String where = KEY_WORD + " LIKE ?";`
selectionArgs	Nilai-nilai argumen untuk kriteria selection. Jika Anda menyertakan ?s dalam selection, maka itu akan digantikan oleh nilai-nilai dari selectionArgs, sesuai urutan munculnya. PENTING: Praktik keamanan terbaik adalah selalu memisahkan selection dan selectionArgs. Misalnya: `String[]whereArgs = new String[]{searchString};`
sortOrder	Urutan mengurutkan hasil. Diformat berupa klausa SQL ORDER BY (dengan mengecualikan kata kunci ORDER BY). Biasanya ASC atau DESC; null akan meminta susunan urutan default, yang boleh jadi tidak berurutan.

Dan buatlah kueri ke penyedia materi seperti ini:

```
Cursor cursor = getContentResolver().query(Uri.parse(queryUri), projection, selectionClause, selectionArgs, sortOrder
);
```

Catatan: Metode insert, delete, dan update disediakan untuk kepraktisan dan kejelasan. Secara teknis, metode query bisa menangani semua permintaan, termasuk untuk menyisipkan, menghapus, dan memperbarui data.

Izin untuk berbagi

Secara default, aplikasi tidak bisa mengakses data aplikasi lainnya. Kedua aplikasi yang terlibat dalam berbagi data harus memiliki izin untuk melakukannya.

- Penyedia materi harus mengizinkan aplikasi lainnya untuk mengakses datanya.
- Pengguna harus mengizinkan aplikasi klien untuk mengakses penyedia materi.

Izin dari penyedia materi

Agar penyedia materi terlihat dan tersedia untuk aplikasi lain, Anda perlu mendeklarasikan dalam AndroidManifest penyedia.

```
<provider android:name=".WordListContentProvider" android:authorities="com.android.example.wordlistsqlwithcontentprov
ider.provider" android:exported="true"/>
```

Properti `android:exported` membuatnya eksplisit karena aplikasi lain bisa menggunakan penyedia materi ini.

Bila tidak ada izin yang disetel secara eksplisit, semua aplikasi lainnya bisa mengakses penyedia materi untuk membaca dan menulis. Untuk membatasi dan membuat batasan akses yang eksplisit, setel izin di dalam tag provider pada penyedia materi, dengan myapp adalah nama unik aplikasi Anda:

```
android:readPermission="com.android.example.wordlistsqlwithcontentpfonrovider.PERMISSION"
android:writePermission="com.android.example.wordlistsqlwithcontentprovider.PERMISSION"
```

- String izin harus unik untuk penyedia materi Anda, sehingga hanya memberi privilese untuk penyedia materi.
- Walaupun string bisa berupa apa saja, penggunaan nama paket akan menjamin keunikannya.

Izin yang diminta aplikasi klien dari pengguna

Untuk mengakses penyedia materi, aplikasi klien perlu mendeklarasikan izin dalam Manifes Android untuk penyedia materi itu.

```
<uses-permission android:name = "com.android.example.wordlistsqlwithcontentprovider.PERMISSION"/>
```

Izin tidak dibahas secara detail dalam konsep ini.

Anda bisa mengetahui selengkapnya dalam Mendeklarasikan Izin, Izin Sistem, dan Mengimplementasikan Izin Penyedia Materi.

Praktik terkait

Dokumentasi praktik terkait ada di Dasar-Dasar Developer Android: Praktik.

- Penyedia Materi Minimalis
- Menambahkan Penyedia Materi pada WorldListSQL
- Berbagi Materi dengan Aplikasi Lainnya

Ketahui selengkapnya

Dokumentasi Developer:

- Dasar-Dasar Penyedia Materi
- Penyedia Materi

- Uniform Resource Identifier atau URI
- Tipe MIME
- MatrixCursor dan Cursor
- Bekerja dengan Izin Sistem
- Mengimplementasikan Izin Penyedia Materi

Video:

- Arsitektur Aplikasi Android
- Arsitektur Aplikasi Android: Miliaran Pengguna Berikutnya

12.1: Loader

Materi:

- Arsitektur loader
- Mengimplementasikan CursorLoader
- Praktik terkait
- Ketahui selengkapnya

Salah satu alasan utama pengguna meninggalkan aplikasi adalah waktu startup. Penelitian menunjukkan bahwa jika sebuah aplikasi atau laman membutuhkan waktu lebih dari 3 detik untuk memuat, 40% pengguna akan meninggalkannya. Bagaimanapun juga, angka ini bervariasi bergantung pada penelitian, namun fakta yang tersebar luas adalah bahwa retensi pengguna berkaitan erat dengan kecepatan pemuatan aplikasi.

Waktu pemuatan aplikasi berkaitan langsung dengan apa saja yang terjadi pada thread UI. Semakin sedikit pekerjaan yang harus dilakukan thread UI Anda, semakin cepat pengguna melihat laman. Banyak faktor yang memengaruhi waktu startup aplikasi, dan Anda mengetahui selengkapnya tentang kinerja aplikasi dalam bab berikutnya. Salah satu tindakan besar dan nyata yang memengaruhi kinerja adalah berapa lama waktu yang diperlukan aplikasi Anda untuk memuat data.

Jika tahu persis asal data, Anda mungkin bisa mengoptimalkan dengan memuatnya sendiri. Jika data disediakan oleh penyedia materi, Anda mungkin tidak tahu backend-nya, dan mungkin tidak tahu apakah untuk pengguna tertentu akan ada data dalam jumlah kecil atau besar.

Solusinya adalah memuat sebagian besar atau semua data Anda di latar belakang, selagi menampilkan informasi yang relevan, yang disimpan secara lokal, kepada pengguna. Misalnya, Anda bisa menampilkan pada mereka informasi cuaca terbaru dalam cache, hingga mendapatkan informasi baru yang menampilkan cuaca saat ini untuk lokasi saat ini.

Loader adalah kelas keperluan khusus yang mengelola pemuatan dan pemuatan ulang asinkron di latar belakang dengan menggunakan AsyncTask.

Loader yang diperkenalkan di Android 3.0 memiliki karakteristik ini:

- Loader tersedia untuk setiap Aktivitas dan Fragmen.
- Loader menyediakan pemuatan data asinkron di latar belakang.
- Loader memantau sumber data mereka dan secara otomatis mengirimkan hasil baru bila materi berubah. Misalnya, jika Anda menampilkan data dalam RecyclerView, saat data yang mendasarinya berubah, CursorLoader secara otomatis memuat rangkaian data yang telah diperbarui, dan bila pemuatan selesai, bisa memberi tahu RecyclerView.Adapter untuk memperbarui apa yang ditampilkan kepada pengguna.
- Loader secara otomatis menghubungkan kembali ke kursor loader saat dibuat kembali setelah perubahan konfigurasi. Karena itu, loader tidak perlu melakukan kueri ulang datanya untuk ditampilkan kepada Anda.

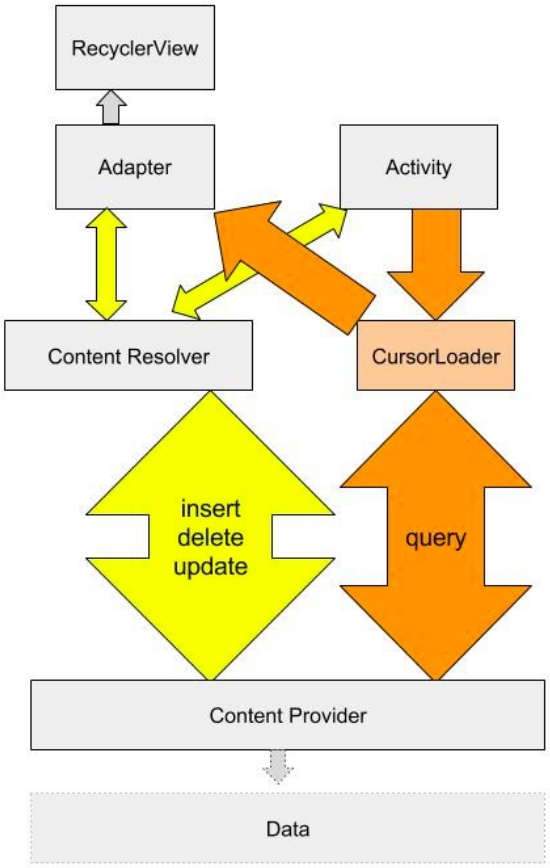
Dalam bab sebelumnya Anda telah mempelajari tentang AsyncTask sebagai kelas serba guna untuk melakukan pekerjaan di latar belakang, dan menggunakan AsyncTaskLoader untuk menjaga data tetap tersedia bagi pengguna melalui perubahan konfigurasi.

Walaupun Anda bisa membuat loader khusus dengan menjadikan kelas Loader sebagai subkelas, kerangka kerja Android menyediakan CursorLoader yang langsung bisa digunakan dan berlaku untuk banyak kasus penggunaan. CursorLoader memperluas AsyncTaskLoader agar secara khusus bekerja dengan penyedia materi, sehingga banyak menghemat pekerjaan Anda.

Perhatikan, loader khusus bisa dibangun. Namun, karena sistem Android menyediakan solusi elegan yang banyak menghemat pekerjaan, pertimbangkan cara Anda menggunakannya seperti yang sudah ditentukan sebelum mengimplementasikan solusi sendiri dari awal. Sebelum menulis loader sendiri, selalu pertimbangkan apakah Anda bisa memperbaiki desain aplikasi untuk bekerja dengan CursorLoader.

Arsitektur loader

Seperti yang ditampilkan dalam diagram di bawah ini, loader akan mengganti panggilan kueri resolver materi ke penyedia materi. Diagram menampilkan versi sederhana dari arsitektur aplikasi bersama sebuah loader. Loader melakukan kueri untuk item dalam latar belakang. Loader mengamati data untuk Anda, dan jika data berubah, loader secara otomatis mendapatkan rangkaian data baru dan menyerahkannya ke adapter.



Mengimplementasikan CursorLoader

Aplikasi yang menggunakan loader biasanya menyertakan yang berikut ini:

- Sebuah Aktivitas atau Fragmen.
- Sebuah instance LoaderManager.
- Sebuah CursorLoader untuk memuat data yang dicadangkan oleh ContentProvider. Atau, Anda bisa mengimplementasikan subkelas Loader atau AsyncTaskLoader sendiri untuk memuat data dari beberapa sumber lainnya.
- Implementasi untuk LoaderManager.LoaderCallbacks. Inilah tempat membuat loader baru dan mengelola referensi Anda ke loader yang sudah ada.
- Cara menampilkan data loader, seperti SimpleCursorAdapter atau RecyclerViewAdapter.
- Sumber data, seperti ContentProvider (dengan CursorLoader).

LoaderManager

LoaderManager adalah kelas praktis yang mengelola semua loader Anda. Anda hanya perlu satu pengelola loader per aktivitas dan secara umum mendapatkannya pada onCreate() aktivitas, tempat mendaftarkan loader yang akan Anda gunakan.

Pengelola loader menangani pendaftaran sebuah observer pada penyedia materi, yang akan menerima callback bila data di penyedia materi berubah.

Satu-satunya panggilan untuk pengelola loader yang perlu Anda buat adalah untuk mendaftarkan loader, dan memulainya lagi saat Anda harus membuang semua data yang dimuat. Parameter pertama adalah ID loader, yang kedua adalah argumen opsional, dan yang ketiga adalah konteks tempat callback didefinisikan.

```
getLoaderManager().initLoader(0, null, this);
getLoaderManager().restartLoader(0, null, this);
```

LoaderManager.LoaderCallbacks

Untuk berinteraksi dengan loader, aktivitas Anda harus mengimplementasikan serangkaian callback yang ditetapkan dalam antarmuka LoaderCallbacks dari LoaderManager. Bila keadaan loader berubah, metode ini akan dipanggil karenanya. Metode tersebut adalah:

- onCreateLoader()—Dipanggil bila loader baru telah dibuat. Mengaitkan data dengan sumber data yang harus dimuat dan diamatinya. (Anda tidak harus melakukan tambahan apa pun bagi loader untuk mengamati sumber data.)
- onLoadFinished()—Dipanggil setiap kali loader selesai memuat. Memicu pembaruan data yang terlihat pengguna dalam metode ini.
- onLoaderReset()—Bila loader disetel ulang, Anda biasanya ingin melakukan invalidasi data yang saat ini ditahan hingga data baru dimuat.

Untuk mengimplementasikan callback ini, Anda perlu mengimplementasikan callback LoaderManager untuk tipe loader yang dimiliki. Untuk loader kursor, ubah tanda tangan aktivitas Anda seperti berikut, kemudian implementasikan callback.

```
public class MainActivity extends AppCompatActivity implements LoaderManager.LoaderCallbacks<Cursor>
```

onCreateLoader()

Callback ini membuat instance dan mengembalikan instance loader baru dengan tipe yang diinginkan. Karena pengelola loader bisa mengelola beberapa loader sekaligus, argumen ID akan mengidentifikasi loader yang akan dibuat instance-nya. Setelah dibuat, loader akan mulai memuat data, dan akan mengamati tanggal perubahan Anda, serta memuat ulang jika diperlukan.

Untuk membuat CursorLoader, Anda memerlukan:

- uri—URI untuk materi yang diambil dari penyedia materi. URI mengidentifikasi penyedia materi dan data yang akan diamati loader.
- projection — Daftar kolom yang akan dikembalikan. Meneruskan nol akan mengembalikan semua kolom, jadi tidak efisien.
- selection — Filter yang mendeklarasikan baris yang akan dikembalikan, diformat sebagai klausa WHERE dari SQL (tidak termasuk WHERE itu sendiri). Meneruskan nol akan mengembalikan semua baris untuk URI yang diberikan.
- selectionArgs — Anda dapat menyertakan ?s dalam selection, yang akan digantikan dengan nilai dari selectionArgs, agar muncul dalam selection. Nilai-nilai akan diikat sebagai String.
- sortOrder — Cara menyusun baris, diformat sebagai klausa ORDER BY dari SQL (tidak termasuk ORDER BY itu sendiri). Meneruskan nol akan menggunakan urutan sortir default, yang mungkin tidak berurutan.

```
@Override
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
    String queryUri = CONTENT_URI.toString();
    String[] projection = new String[] {CONTENT_PATH};
    return new CursorLoader(this, Uri.parse(queryUri),
            projection, null, null, null);
}
```

Perhatikan betapa miripnya dengan inisiasi resolver materi:

```
Cursor cursor = mContext.getContentResolver().query(Uri.parse(uri),
projection, selectionClause, selectionArgs, sortOrder);
```

onLoadFinished()

Menetapkan apa yang terjadi dengan data setelah loader mendapatkannya. Dalam fungsi ini Anda harus:

- Merilis data lama.
- Menyimpan data baru dan, misalnya, menyediakannya untuk adapter Anda.

Loader kursor memantau data untuk Anda, jadi Anda tidak perlu, dan tidak seharusnya dalam keadaan apa pun, melakukannya sendiri.

Loader juga membersihkannya sendiri setelah dirinya sendiri, jadi Anda tidak perlu menutup kursor.

Jika Anda menggunakan RecyclerView untuk menampilkan data, yang perlu Anda lakukan hanyalah meneruskan data ke adapter bila pemuatan atau pemuatan ulang selesai.

```
@Override
public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {
    mAdapter.setData(cursor);
}
```

onLoaderReset()

Dipanggil bila loader yang dibuat sebelumnya akan disetel ulang, sehingga datanya tidak tersedia. Anda harus membersihkan semua referensi ke data di saat ini. Lagi, jika Anda meneruskan data ke adapter untuk ditampilkan dalam RecyclerView, adapter akan melakukan pekerjaan sebenarnya, Anda hanya perlu menginstruksikannya.

```
@Override
public void onLoaderReset(Loader<Cursor> loader) {
    mAdapter.setData(null);
}
```

Menggunakan data yang dikembalikan oleh loader

Dalam praktiknya, Anda menggunakan RecyclerView yang dikendalikan oleh adapter untuk menampilkan data yang diambil oleh loader. Setelah menerima data, loader menyerahkannya ke adapter melalui, misalnya, panggilan setData(). Metode setData() memperbarui variabel instance dalam adapter yang berisi set data terbaru, dan memberi tahu adapter bahwa ada data baru.

```
public void setData(Cursor cursor) {
    mCursor = cursor;
    notifyDataSetChanged();
}
```

Manfaat kursor

Anda mungkin sudah mengetahui bahwa database menggunakan kursor, penyedia materi menggunakan kursor, loader juga menggunakan kursor. Dengan menggunakan tipe data yang sama di seluruh backend, dan hanya mengekstraknya dalam adapter, tempat isi kursor dipersiapkan untuk ditampilkan, akan membuat backend seragam dengan antarmuka yang bersih. Hal ini akan memudahkan penulisan kode, memudahkan pengujian, dan memudahkan debug. Hal ini juga membuat kode lebih sederhana dan lebih pendek.

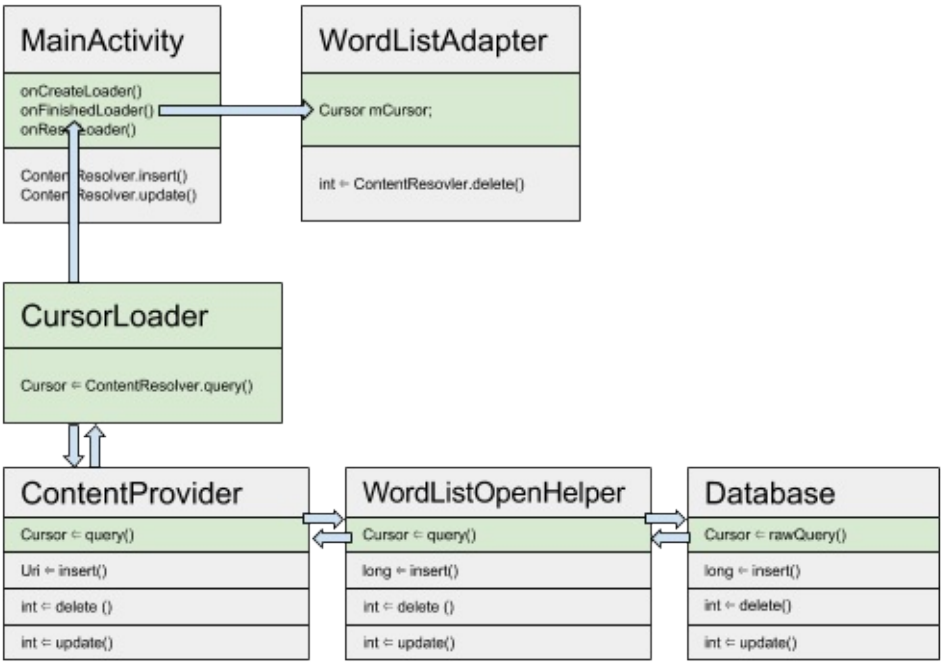
Lengkapi aplikasi dengan metode

Diagram berikut menampilkan metode dan tipe data yang menghubungkan aneka bagian aplikasi yang menggunakan:

- Database SQLite untuk menyimpan data, dan subkelas SQLiteOpenHelper untuk mengelola database.
- Penyedia materi untuk menyediakan data ke aplikasi ini (dan aplikasi lainnya).
- Loader untuk memuat data yang ditampilkan kepada pengguna.

- RecyclerView.Adapter yang menampilkan dan memperbarui data yang ditampilkan kepada pengguna di RecyclerView.

Kotak warna hijau menampilkan tumpukan panggilan dan perjalanan kursor melalui beberapa layer aplikasi untuk query(). Perhatikan bagaimana penyisipan, penghapusan, dan pembaruan tetap ditangani oleh resolver materi. Akan tetapi, loader akan memberitahukan setiap perubahan yang buat oleh operasi penyisipan, penghapusan, atau pembaruan, dan akan memuat ulang data bila diperlukan.



Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Memuat dan Menampilkan Data yang Diambil dari Penyedia Materi

Ketahui selengkapnya

Dokumentasi Developer:

- Loader
- Menjalankan kueri bersama CursorLoader
- Kelas CursorLoader

13.1: Izin, Kinerja, dan Keamanan

Materi:

- Izin
- Kinerja
- Praktik terbaik keamanan

Sekarang Anda telah mempelajari keterampilan inti mendasar yang diperlukan untuk membangun aplikasi Android. Pelajaran ini membahas praktik terbaik yang berkaitan dengan izin, kinerja dan keamanan. Pelajaran ini tidak berisi praktik yang bersangkutan.

Izin

Saat melakukan praktik, ada saatnya aplikasi Anda perlu mendapatkan izin untuk melakukan sesuatu, termasuk saat memerlukannya untuk:

- menghubungkan ke Internet.
- menggunakan penyedia materi di aplikasi lain.

Bagian ini memberikan ringkasan tentang izin sehingga Anda memahami bagaimana dan kapan aplikasi perlu meminta izin agar bisa berfungsi dan bertindak.

Mintalah izin jika tidak memilikinya

Aplikasi bebas menggunakan sumber daya atau data yang dibuatnya, namun harus mendapatkan izin untuk menggunakan sesuatu—data, sumber daya, perangkat keras, perangkat lunak—yang bukan miliknya. Misalnya, aplikasi Anda harus mendapatkan izin untuk membaca data Kontak milik pengguna, atau menggunakan kamera perangkat. Wajar jika aplikasi memerlukan izin untuk membaca Kontak pengguna, namun Anda mungkin bertanya-tanya mengpa perlu izin untuk menggunakan kamera. Hal ini karena perangkat keras kamera bukan milik aplikasi, dan aplikasi Anda harus selalu mendapatkan izin untuk menggunakan apa pun yang bukan bagian dari aplikasi itu sendiri.

Meminta izin

Untuk meminta izin, tambahkan atribut `<uses-permission>` ke file manifes Android, bersama nama izin yang diminta. Misalnya, untuk mendapatkan izin menggunakan kamera:

```
<uses-permission android:name="android.permission.CAMERA"/>
```

Contoh izin

Kerangka kerja Android menyediakan lebih dari 100 izin yang telah didefinisikan sebelumnya. Ini termasuk beberapa hal yang sudah jelas, antara lain izin untuk mengakses atau menulis data pribadi pengguna seperti:

- membaca dan menulis daftar Kontak, kalender, atau pesan suara milik pengguna
- mengakses lokasi perangkat
- mengakses data dari sensor tubuh

Sebagian izin lain yang telah didefinisikan sebelumnya kurang jelas, seperti izin untuk mengumpulkan statistik baterai, izin untuk menghubungkan ke internet, dan izin untuk menggunakan perangkat keras seperti perangkat keras sidik jari atau kamera.

Android menyertakan beberapa izin yang telah didefinisikan sebelumnya untuk inisialisasi panggilan telepon tanpa mengharuskan pengguna mengonfirmasinya, membaca log panggilan, mengambil keluaran video, mem-boot ulang perangkat, mengubah tanggal dan zona waktu, dan banyak lagi yang lainnya.

Anda bisa melihat semua izin yang didefinisikan sistem di https://developer.android.com/reference/android/Manifest.permission.html.

Izin normal dan berbahaya

Android mengklasifikasikan izin sebagai normal atau berbahaya.

Izin normal adalah untuk tindakan yang tidak memengaruhi privasi pengguna atau data pengguna, seperti menghubungkan ke Internet.

Izin berbahaya adalah untuk aksi yang memengaruhi privasi pengguna atau data pengguna, seperti izin untuk menulis ke pesan suara pengguna.

Android secara otomatis memberikan izin normal namun meminta pengguna untuk memberikan izin berbahaya secara eksplisit.

Catatan: Aplikasi harus mencantumkan semua izin yang digunakan dalam manifes Android, bahkan izin normal.

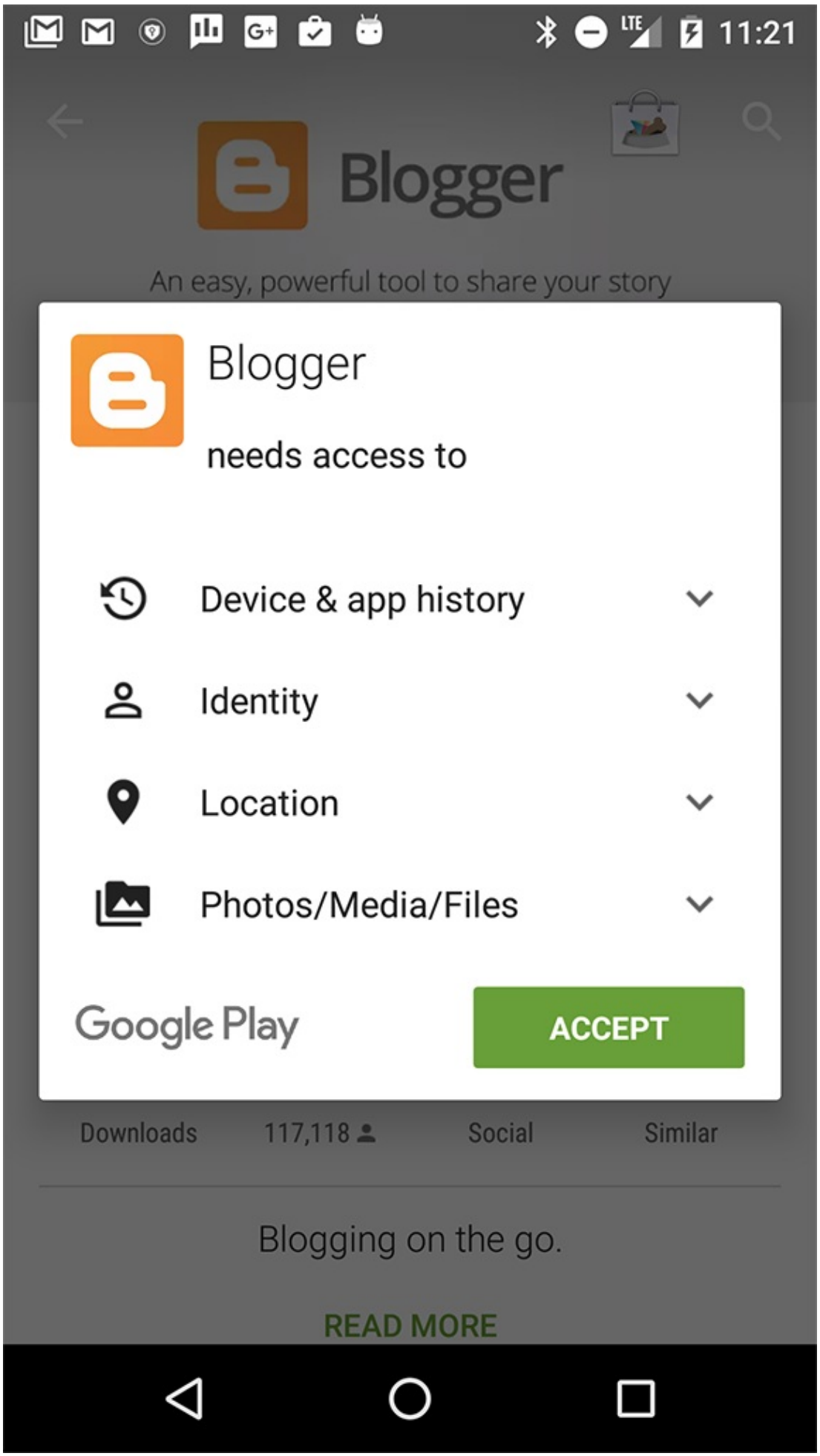
Cara pengguna memberikan dan mencabut izin

Cara pengguna memberikan dan mencabut izin bergantung pada:

- versi Android yang dijalankan perangkat.
- versi Android untuk aplikasi yang dibuat.

Sebelum Marshmallow (Android 6.0)

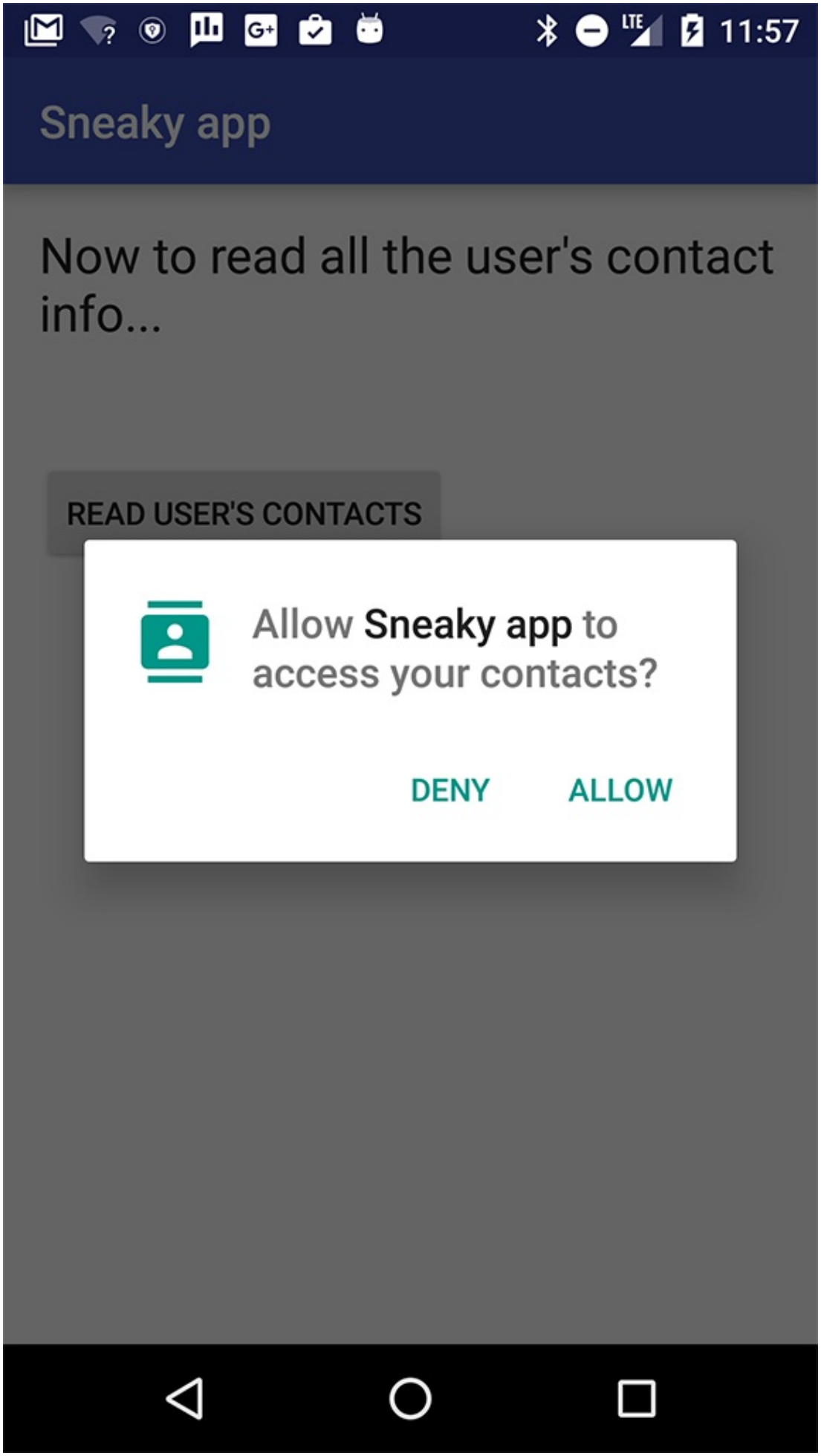
Jika aplikasi *dibuat* untuk versi Android sebelum 6.0 (Marshmallow) ***atau*** *berjalan* pada perangkat yang menggunakan versi Android sebelum Marshmallow, Google Play akan meminta pengguna untuk memberikan izin berbahaya yang diperlukan ***sebelum memasang aplikasi***.


11:21
Blogger
An easy, powerful tool to share your story
Blogger
needs access to
Device & app history
Identity
Location
Photos/Media/Files
Google Play
ACCEPT
Downloads
117,118
Social
Similar
Blogging on the go.
READ MORE

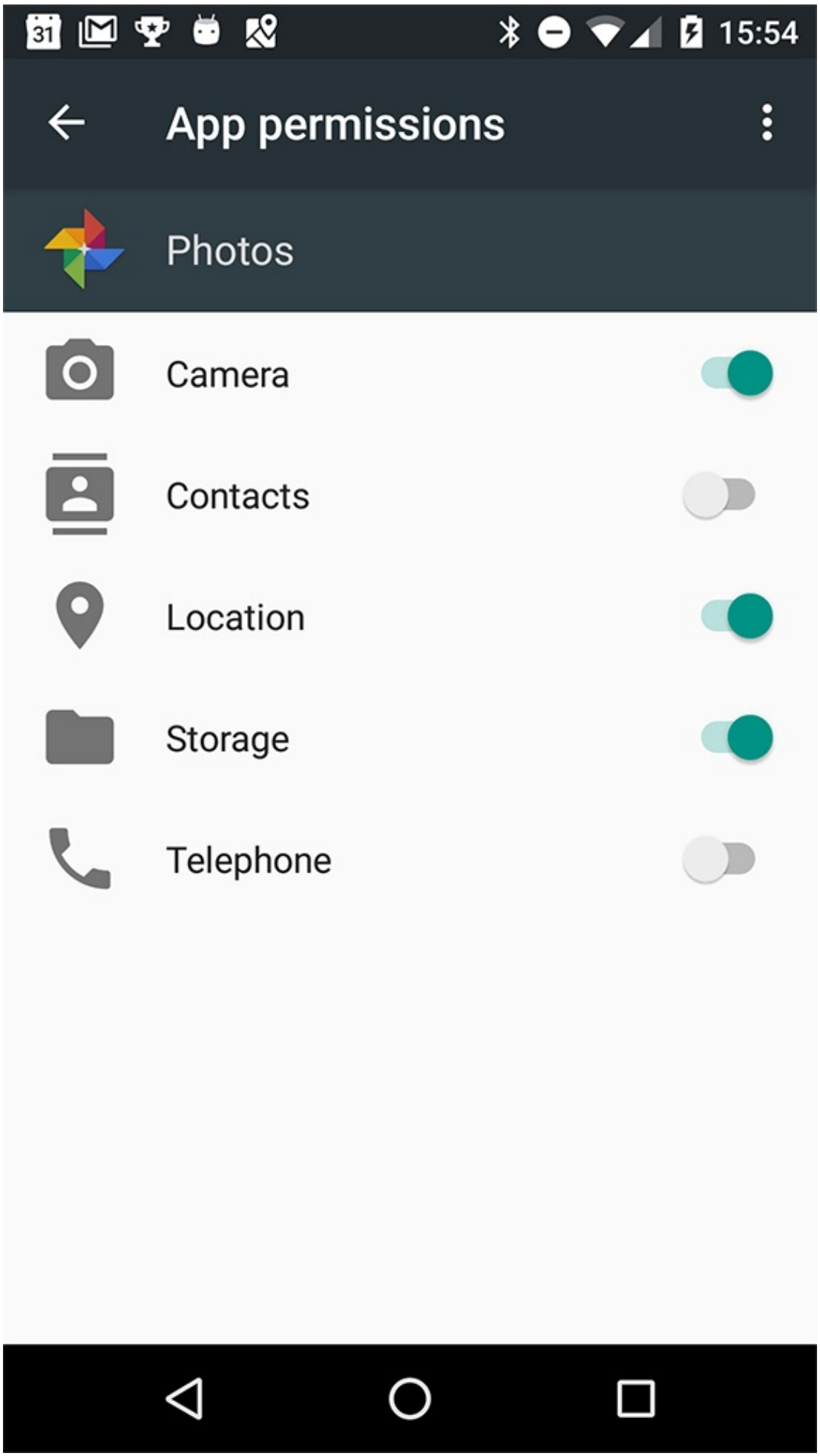
Jika pengguna berubah pikiran dan ingin menolak izin ke aplikasi setelah dipasang, satu-satunya hal yang bisa dilakukan adalah dengan mencopot pemasangan aplikasi.

Marshmallow dan selanjutnya

Jika aplikasi dibuat untuk versi Android dari Android 6.0 (Marshmallow) dan selanjutnya ***dan*** sedang berjalan pada perangkat yang menggunakan versi Android dari Marshmallow dan selanjutnya, maka Google Play tidak akan meminta pengguna untuk memberikan izin berbahaya pada aplikasi sebelum memasangnya. Sebagai gantinya, bila pengguna mulai melakukan sesuatu di aplikasi yang memerlukan level izin tersebut, Android akan menampilkan kotak dialog yang meminta pengguna untuk memberikan izin.


11:57
Sneaky app
Now to read all the user's contact info...
READ USER'S CONTACTS
Allow Sneaky app to access your contacts?
DENY
ALLOW

Pengguna bisa memberikan atau mencabut izin individual kapan saja. Mereka melakukannya dengan masuk ke Settings App, memilih Apps, dan memilih aplikasi yang relevan. Di bagian Permissions, mereka bisa mengaktifkan atau menonaktifkan izin yang digunakan aplikasi.


15:54
App permissions
Photos
Camera
Contacts
Location
Storage
Telephone

Perbedaan dalam model izin memengaruhi developer

Dalam model izin "lama", Google Play dan Kerangka Kerja Android bekerja sama untuk mendapatkan izin dari pengguna. Developer tinggal memastikan aplikasi mencantumkan izin yang diperlukan dalam file manifes Android. Developer bisa beranggapan bahwa jika aplikasi dijalankan, berarti pengguna telah memberikan izin. Developer tidak perlu menulis kode untuk memeriksa apakah izin telah diberikan atau tidak.

Dalam model izin "baru", Anda tidak bisa lagi beranggapan bahwa jika aplikasi dijalankan berarti pengguna telah memberikan izin yang diperlukan. Pengguna bisa memberikan izin saat pertama menjalankan aplikasi, kemudian, kapan saja, mereka berubah pikiran dan mencabut salah satu atau semua izin yang diperlukan aplikasi.

Jadi, aplikasi harus memeriksa apakah masih memiliki izin setiap kali melakukan sesuatu yang memerlukan izin. Android SDK menyertakan API untuk memeriksa jika izin telah diberikan. Inilah cuplikan kode untuk memeriksa apakah aplikasi memiliki izin untuk menulis ke kalender pengguna:

```
// Assume thisActivity is the current activity
int permissionCheck = ContextCompat.checkSelfPermission(thisActivity,
   Manifest.permission.WRITE_CALENDAR);
```

Kerangka kerja Android untuk Android 6.0 (API level 23) menyertakan metode untuk memeriksa dan meminta izin. Pustaka Dukungan juga menyertakan metode untuk memeriksa dan meminta izin.

Kami menyarankan Anda menggunakan metode pustaka dukungan untuk menangani izin, karena metode izin di pustaka dukungan menangani pemeriksaan versi Android yang dijalankan pada aplikasi Anda, dan mengambil aksi yang tepat. Misalnya, jika perangkat pengguna menjalankan versi lama, maka metode `checkSelfPermission()` di pustaka dukungan akan memeriksa apakah pengguna sudah memberikan izin pada waktu proses, namun jika perangkat menjalankan Marshmallow atau yang lebih baru, maka metode tersebut akan memeriksa apakah izin masih diberikan, dan jika tidak, akan menampilkan dialog kepada pengguna untuk meminta izin.

Pelajaran ini tidak membahas secara detail tentang cara menggunakan API untuk menangani izin. Lihat Meminta Izin pada Waktu Proses untuk detailnya.

Praktik terbaik untuk izin

Bila aplikasi meminta izin terlalu banyak, pengguna akan curiga. Pastikan aplikasi Anda hanya meminta izin untuk fitur dan tugas yang benar-benar diperlukannya, dan pastikan pengguna memahami alasan diperlukannya izin tersebut.

Bila memungkinkan, gunakan Maksud sebagai ganti meminta izin untuk melakukannya sendiri. Misalnya, jika aplikasi Anda perlu menggunakan kamera, kirim Maksud ke aplikasi kamera, dan dengan cara itu aplikasi kamera akan melakukan semua pekerjaan untuk Anda dan aplikasi tidak perlu mendapatkan izin untuk menggunakan kamera (dan itu akan lebih memudahkan Anda menulis kode daripada jika mengakses API kamera secara langsung).

Ketahui selengkapnya tentang Izin di Android

- Izin yang telah didefinisikan sebelumnya
- Praktik terbaik untuk izin
- Entri blog tentang izin waktu proses

Kinerja

Anda telah berusaha membuat aplikasi yang paling bermanfaat, menarik, dan indah. Akan tetapi, untuk membuatnya tampil beda, Anda juga harus menjadikannya sekecil, secepat, dan seefisien mungkin. Pertimbangkan kemungkinan dampak aplikasi Anda pada baterai, memori, dan ruang disk perangkat. Dan yang terpenting, pertimbangkan paket data pengguna. Saran berikut ini hanya bagian kecil dari masalah kinerja, namun ini akan memberi Anda ide untuk memulai.

Penting: Memaksimalkan kinerja adalah masalah keseimbangan, menemukan dan membuat konsekuensi terbaik antara

kompleksitas aplikasi, fungsi, dan visual untuk memberikan pengalaman terbaik kepada pengguna aplikasi Anda.

Pindahkan tugas yang berjalan lama dari thread utama

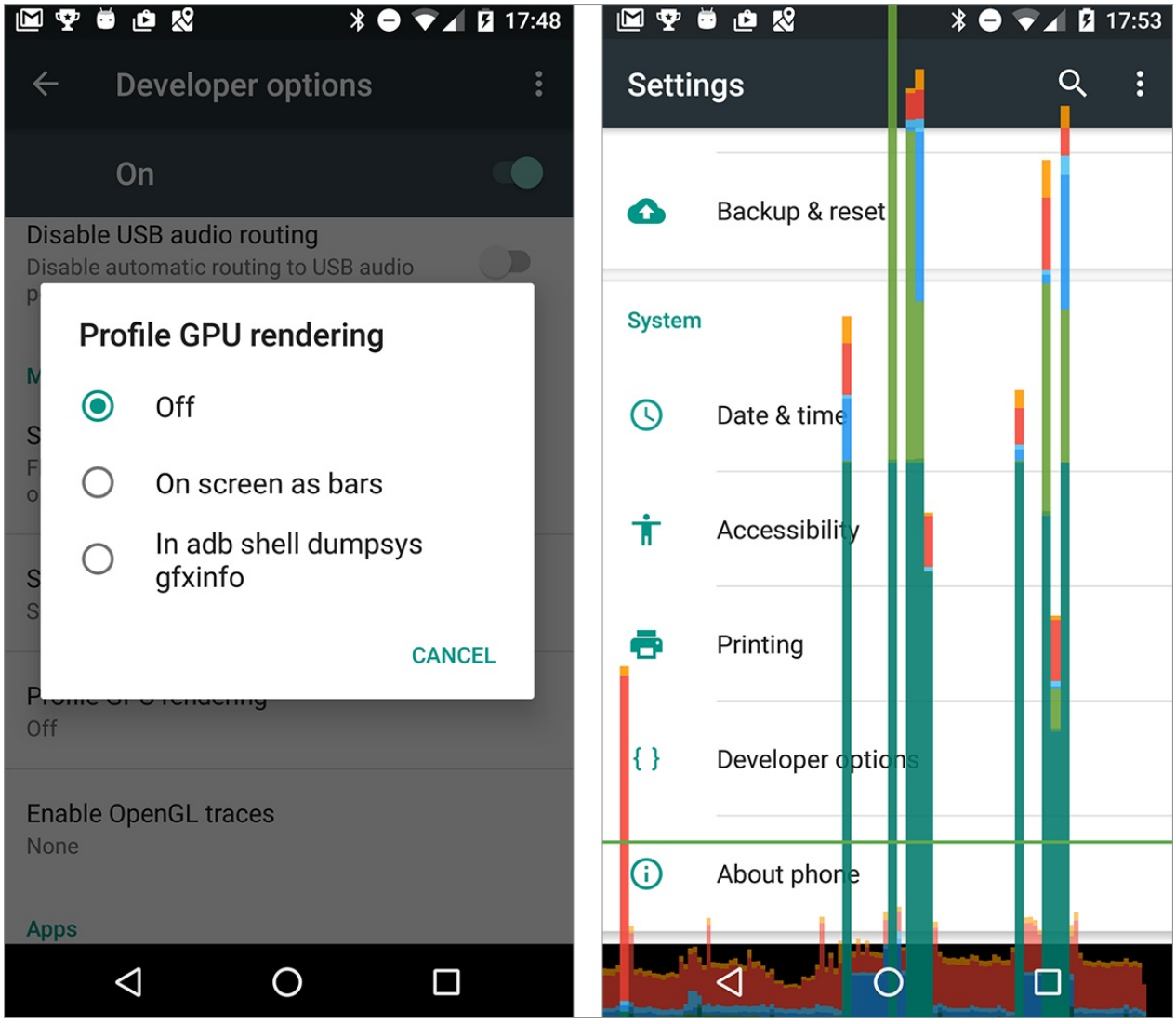
Kursus ini sudah membahas tentang memindahkan pekerjaan dari thread utama ke latar belakang untuk membantu menjaga UI tetap lancar dan responsif bagi pengguna. Perangkat keras yang merender tampilan ke layar biasanya memperbarui layar setiap 16 milidetik, jadi jika thread utama melakukan pekerjaan yang memerlukan waktu lebih lama dari 16 milidetik, aplikasi mungkin akan melewatkan bingkai, tersendat, atau mogok, semua itu mungkin akan mengganggu pengguna Anda.

Anda bisa memeriksa seberapa baik aplikasi Anda di layar rendering dalam batas 16 milidetik dengan menggunakan alat (bantu) **Profile GPU Rendering** di perangkat Android Anda.

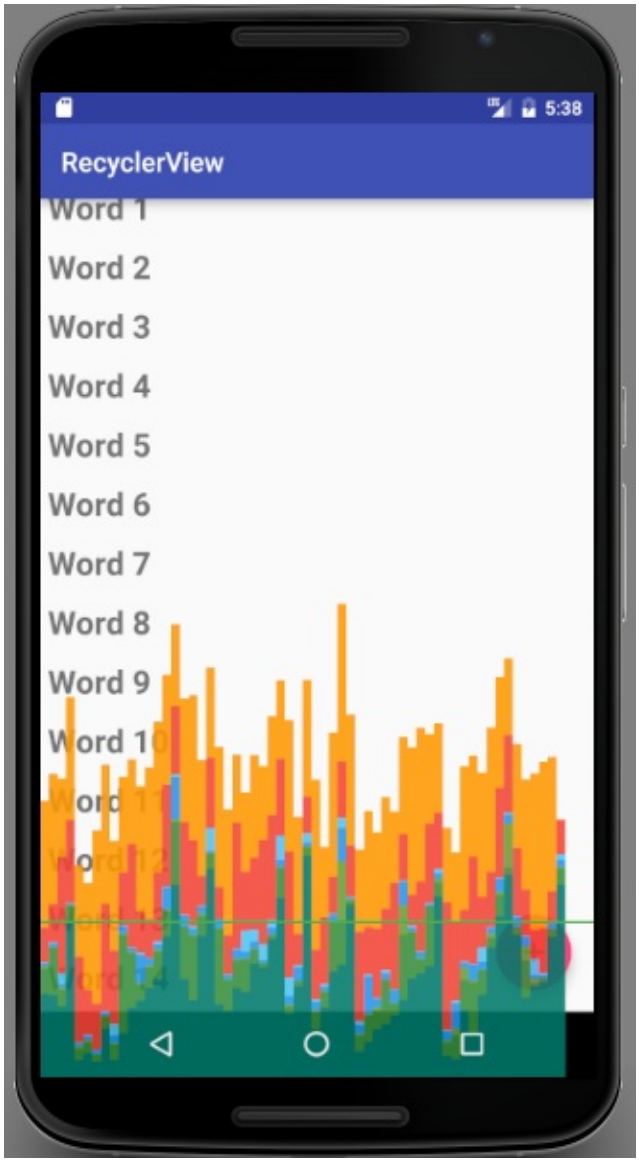
1. Masuk ke **Settings** > **Developer options**.
2. Gulir ke bawah ke bagian **Monitoring**.
3. Pilih **Profile GPU rendering**.
4. Pilih **On screen as bars** di kotak dialog.

Anda akan segera mulai melihat bilah berwarna pada layar.

Catatan: Anda bisa menjalankan alat (bantu) pada emulator, hanya untuk mencobanya, namun data tersebut tidak menunjukkan cara kerja aplikasi pada perangkat sesungguhnya.



Buka aplikasi Anda, dan amati bilah berwarna.



Satu bilah menyatakan satu layar yang dirender. Jika bilah berada di atas garis hijau, berarti butuh waktu lebih dari 16 md untuk merender. Warna bilah menyatakan beragam tahapan merender layar.

Baca tentang cara menafsirkan hasil dan maksud beragam tahapan dalam Menganalisis dengan Profile GPU Rendering. Jika Anda menghabiskan waktu menggunakan alat (bantu) Profile GPU rendering di aplikasi, hal itu akan membantu mengidentifikasi bagian interaksi UI mana yang lebih lambat dari perkiraan, kemudian Anda bisa mengambil aksi untuk meningkatkan kecepatan UI aplikasi.

Misalnya, jika bagian bilah Input hijau besar, berarti aplikasi menghabiskan banyak waktu untuk menangani kejadian masukan, yaitu mengeksekusi kode yang dipanggil sebagai hasil callback kejadian masukan. Untuk memperbaikinya, pertimbangkan waktu dan cara meminta masukan pengguna, dan apakah Anda bisa menanganinya dengan lebih efisien.

Sederhanakan UI Anda

Kursus ini telah membahas tentang cara membuat aplikasi menjadi menarik dan memikat secara visual dengan menggunakan pedoman desain material dan mengajari Anda cara menggunakan Layout Editor untuk membuat layout. Anda telah mengetahui bahwa Anda bisa membuat hierarki layout tersarang. Anda telah mengetahui cara menggunakan sumber daya dapat digambar sebagai elemen latar belakang untuk tampilan. Elemen ini memungkinkan Anda membuat layout yang disarangkan secara kompleks dengan berbagai latar belakang dan tampilan yang tumpang tindih satu sama lain di seluruh bagian aplikasi.

Akan tetapi, layout Anda akan digambar lebih cepat dan menggunakan daya baterai lebih sedikit daya jika Anda menghabiskan waktu untuk mendesainnya dengan cara yang paling efisien.

Cobalah hindari:

- Layout yang disarangkan terlalu dalam—Jika layout Anda sempit dan dalam, sistem Android harus melakukan banyak penerusan untuk menata semua tampilan daripada jika hierarki tampilan Anda lebar dan dangkal. Pertimbangkan cara menggabung, meratakan, atau bahkan menghilangkan tampilan.
- Tampilan tumpang tindih—hal ini mengakibatkan "overdraw", yakni aplikasi memboroskan waktu untuk menggambar piksel yang sama beberapa kali, dan hanya tampilan terakhir yang terlihat oleh pengguna. Pertimbangkan cara mengukur dan mengatur tampilan sehingga setiap piksel hanya digambar sekali atau dua kali.

Sederhanakan layout

Pastikan layout Anda hanya menyertakan tampilan dan fungsi yang diperlukan aplikasi. Bagi pengguna, layout sederhana umumnya lebih menarik secara visual, dan digambar lebih cepat, sehingga memberi Anda keuntungan ganda.

Ratakan layout sebisa mungkin, sehingga mengurangi jumlah level yang disarangkan di hierarki tampilan aplikasi Anda. Misalnya, jika layout berisi LinearLayout di dalam LinearLayout di dalam LinearLayout, Anda mungkin bisa menyusun semua tampilan di dalam satu ConstraintLayout.

Lihat panduan Mengoptimalkan UI Anda untuk informasi selengkapnya tentang meningkatkan kinerja UI aplikasi Anda.

Minimalkan tampilan tumpang tindih

Bayangkan mengecat pintu rumah Anda dengan warna merah. Kemudian Anda cat lagi dengan warna hijau. Lalu dicat lagi dengan warna biru. Akhirnya, satu-satunya warna yang terlihat adalah biru, namun Anda memboroskan banyak energi untuk mengecat pintu beberapa kali.

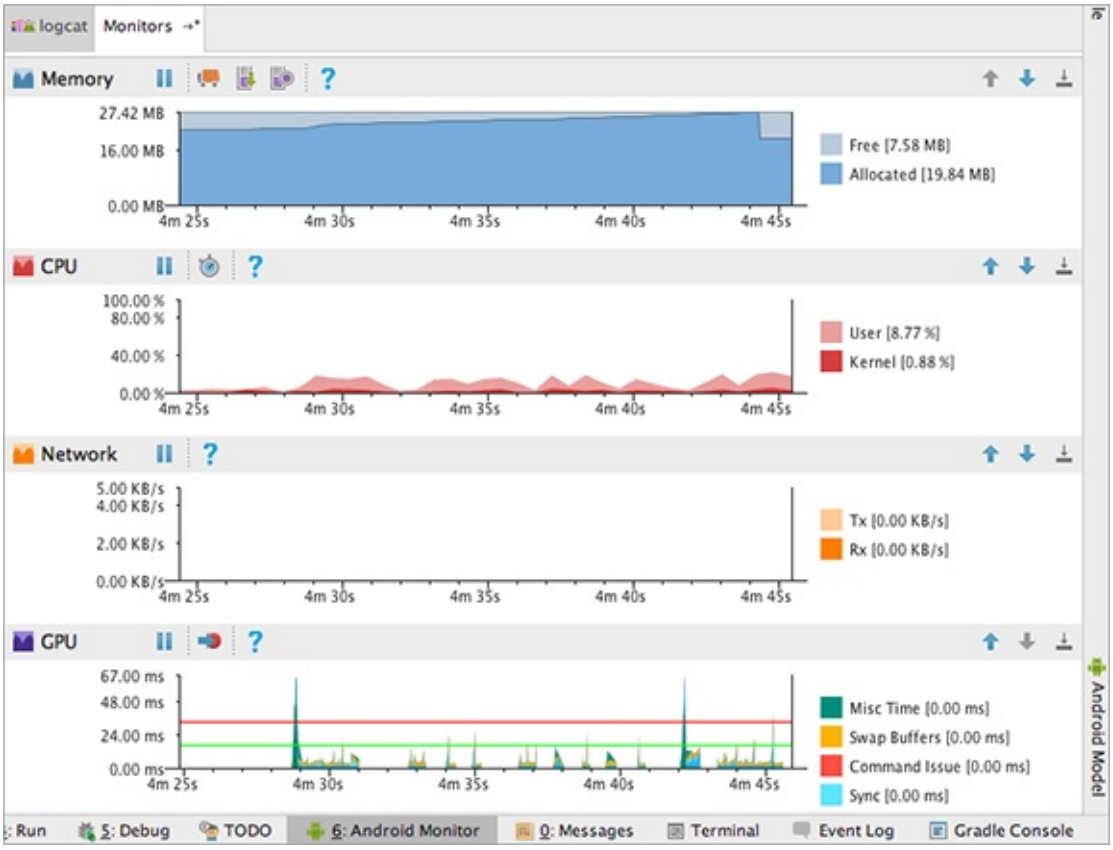
Setiap layout di aplikasi Anda seperti pintu tersebut. Aplikasi memerlukan waktu setiap kali "mengecat" (menggambar) piksel. Jika layout memiliki tampilan tumpang tindih, maka aplikasi Anda menggunakan waktu dan sumber daya untuk terus-menerus menggambar piksel. Coba kurangi jumlah waktu yang digunakan aplikasi Anda untuk menggambar piksel secara berlebihan, dengan mengurangi tampilan tumpang tindih. Berhati-hatilah dalam menggunakan latar belakang yang dapat digambar pada tampilan tumpang tindih dan hanya gunakan bila terlihat.

Lihat panduan Mengurangi Overdraw untuk informasi selengkapnya.

Pantau kinerja aplikasi Anda yang sedang berjalan

Android Studio memiliki alat (bantu) untuk mengukur penggunaan memori, GPU, CPU, dan kinerja jaringan oleh aplikasi. Aplikasi yang mogok sering kali berkaitan dengan kebocoran memori, yang terjadi bila aplikasi mengalokasikan memori dan tidak membebaskannya. Jika aplikasi Anda mengalami kebocoran memori, atau menggunakan memori lebih banyak daripada yang disediakan perangkat, pada akhirnya aplikasi akan menggunakan semua memori yang tersedia pada perangkat. Gunakan alat (bantu) Memory Monitor yang disertakan bersama Android Studio untuk mengamati cara aplikasi menggunakan memori.

1. Di Android Studio, di bagian bawah jendela, klik tab **Android Monitor**. Secara default ini akan membuka logcat.
2. Klik tab **Monitors** di sebelah logcat. Gulir atau perbesar jendela untuk melihat keempat monitor: Memori, CPU, Jaringan, dan GPU.
3. Jalankan aplikasi Anda dan berinteraksilah dengannya. Monitor akan diperbarui untuk mencerminkan penggunaan sumber daya oleh aplikasi. Perhatikan, untuk mendapatkan data yang akurat, Anda harus melakukannya pada perangkat fisik, bukan virtual.



Monitor tersebut adalah:

- **Monitor memori**—Melaporkan cara aplikasi mengalokasikan memori dan membantu Anda memvisualisasikan memori yang digunakan aplikasi.
- **Monitor CPU**—Memungkinkan Anda memantau penggunaan unit pemroses pusat (CPU) oleh aplikasi. Monitor menampilkan penggunaan CPU secara realtime.
- **Monitor GPU**—Memberikan representasi visual tentang seberapa lama waktu yang diperlukan unit pemroses grafik (GPU) untuk merender bingkai ke layar.
- **Monitor Jaringan**—Menampilkan waktu kapan aplikasi membuat permintaan jaringan. Ini memungkinkan Anda melihat cara dan waktu aplikasi mentransfer data, serta mengoptimalkan kode yang mendasarinya dengan semestinya.

Baca laman Android Monitor untuk mengetahui selengkapnya tentang penggunaan monitor.

Ketahui selengkapnya tentang meningkatkan kinerja aplikasi Anda

- Anda, Aplikasi Anda, dan Kinerja
- Melampaui batas kecepatan Android

Overdraw:

- Gambarlah apa yang Anda lihat saja
- Mengurangi Overdraw

Alat:

- Alat profil kinerja
- Mengoptimalkan UI Anda
- Menganalisis dengan Profile GPU Rendering
- Android Monitor

Praktik terbaik keamanan

Kebanyakan beban dalam membangun aplikasi yang aman telah ditangani oleh Kerangka Kerja Android untuk Anda. Misalnya, aplikasi diisolasi satu sama lain agar tidak bisa mengakses satu sama lain atau menggunakan data masing-masing tanpa izin.

Akan tetapi, sebagai developer aplikasi, Anda bertanggung jawab memastikan aplikasi memperlakukan data pengguna secara aman dan berintegritas. Aplikasi Anda juga bertanggung jawab menjaga keamanan data miliknya.

Menangani data pengguna

Pelajaran ini sudah membahas cara Android menggunakan izin untuk memastikan aplikasi tidak bisa mengakses data pribadi pengguna tanpa seizin mereka. Namun sekalipun pengguna mengizinkan aplikasi Anda mengakses data pribadi mereka, jangan melakukannya kecuali jika benar-benar diperlukan. Dan jika Anda melakukannya, perlakukan data dengan integritas dan rasa hormat. Misalnya, hanya karena pengguna mengizinkan aplikasi untuk memperbarui kalender, bukan berarti Anda diizinkan menghapus semua entri kalender mereka.

Aplikasi Android beroperasi atas dasar kepercayaan tersirat. Pengguna percaya bahwa aplikasi akan menggunakan data mereka dengan cara yang wajar dalam konteks aplikasi.

Jika aplikasi Anda berupa aplikasi perpesanan, kemungkinan pengguna akan memberikan izin untuk membaca kontak mereka. Itu tidak berarti aplikasi Anda diizinkan membaca semua kontak pengguna dan mengirim pesan spam ke semua orang.

Aplikasi Anda hanya boleh membaca dan menulis data pengguna bila benar-benar diperlukan, dan hanya dengan cara yang diperkirakan oleh pengguna. Setelah aplikasi membaca data privat, Anda harus menjaga data tersebut tetap aman dan jangan sampai bocor. Jangan berbagi data privat dengan aplikasi lainnya.

Bergantung pada cara aplikasi menggunakan data pengguna, Anda juga mungkin perlu menyediakan pernyataan tertulis mengenai praktik privasi bila mempublikasikan aplikasi di Google Play store.

Ketahuilah bahwa data yang diperoleh, diunduh, atau dibeli pengguna di aplikasi Anda adalah milik mereka, dan aplikasi harus menyimpannya dengan cara yang tetap memungkinkan akses pengguna, bahkan jika pengguna mencopot pemasangannya.

Penting: Log adalah sumber daya bersama di semua aplikasi. Aplikasi yang memiliki izin READ_LOGS bisa membaca semua log. **Jangan** menulis data privat pengguna ke log.

Wi-Fi Umum

Banyak orang menggunakan aplikasi seluler melalui Wi-Fi umum. Kapan terakhir kali Anda mengakses Internet dari ponsel melalui Wi-Fi umum di kedai kopi, bandara, atau stasiun kereta api?

Desainlah aplikasi Anda untuk melindungi data pengguna bila mereka terhubung ke Wi-Fi umum. Gunakan `https` daripada `http` bila memungkinkan untuk menghubungkan ke situs web. Enkripsilah data pengguna yang akan ditransmisikan, bahkan data yang mungkin tampak polos seperti nama mereka.

Untuk mentransmisikan data sensitif, implementasikan komunikasi yang diautentikasi dan dienkripsi tingkat-soket dengan menggunakan kelas SSLSocket. Kelas ini menambahkan layer perlindungan keamanan pada protokol transportasi jaringan yang mendasarinya. Perlindungan tersebut menyertakan perlindungan terhadap modifikasi pesan oleh wiretapper, autentikasi yang disempurnakan dengan server, dan perlindungan privasi yang ditingkatkan.

Memvalidasi masukan pengguna

Jika aplikasi menerima masukan (dan hampir setiap aplikasi melakukannya!), Anda perlu memastikan bahwa masukan tersebut tidak membawa sesuatu yang berbahaya.

Jika aplikasi menggunakan kode asli, membaca data dari file, menerima data melalui jaringan, atau menerima data dari sumber eksternal, maka aplikasi Anda berpotensi menimbulkan masalah keamanan. Masalah paling umum adalah buffer meluap, pointer bergetar, dan off-by-one error atau OBOE. Android menyediakan sejumlah teknologi yang mengurangi tingkat eksploitasi kesalahan ini, namun tidak memecahkan masalah pokoknya. Anda bisa menghindari celah keamanan ini dengan menangani pointer dan mengelola buffer secara hati-hati.

Jika aplikasi mengizinkan pengguna untuk memasukkan kueri yang dikirimkan ke database SQL atau penyedia materi, Anda harus mewaspadai injeksi SQL. Inilah teknik yang memungkinkan pengguna jahat bisa menyuntikkan perintah SQL ke dalam pernyataan SQL dengan memasukkan data dalam bidang. Perintah SQL yang disuntikkan bisa mengubah pernyataan SQL dan mengganggu keamanan aplikasi serta database.

Bacalah tentang penggunaan kueri berparameter untuk melindungi terhadap injeksi SQL di bagian penyedia materi pada panduan Tips Keamanan.

Hal lain yang bisa dilakukan untuk membatasi risiko injeksi SQL adalah menggunakan atau memberikan izin READ_ONLY atau WRITE_ONLY untuk penyedia materi.

WebViews

Salah satu kelas View di Android adalah WebView yang menampilkan laman web.

Kursus ini tidak membahas WebView, namun Anda mungkin telah menemukan dan mencobanya sendiri. Kami menyebutkannya di sini karena meskipun sangat keren untuk menampilkan laman web dengan cepat di aplikasi, WebView membawa masalah keamanan.

Karena WebView mengonsumsi materi web yang bisa berisi HTML dan JavaScript, maka bisa mengakibatkan masalah keamanan web umum seperti penulisan skrip lintas situs (injeksi JavaScript).

Secara default, WebView tidak menyediakan widget seperti browser, tidak mengaktifkan JavaScript, dan mengabaikan kesalahan laman web. Jika tujuan Anda hanya untuk menampilkan beberapa HTML sebagai bagian dari UI, maka hal itu bisa diterima asalkan pengguna tidak perlu berinteraksi dengan laman web sesudah membacanya, dan laman web tidak perlu berinteraksi dengan pengguna. Jika Anda ingin web browser berfungsi penuh, panggil aplikasi Browser dengan Maksud URL, bukan menampilkan laman dengan WebView. Ini juga merupakan opsi yang lebih aman daripada memperluas kelas WebView dan mengaktifkan fitur seperti JavaScript.

Keamanan, seperti hal kinerja, adalah topik besar yang tidak bisa dibahas dalam beberapa paragraf. Anda bertanggung jawab memperlakukan data pengguna dengan hati-hati dan menjaga keamanannya setiap saat. Gunakan sumber daya di bawah ini untuk Anda pelajari sebanyak mungkin tentang memperlakukan pengguna dan data mereka dengan perhatian tertinggi.

Ketahui selengkapnya tentang praktik terbaik keamanan

- Tips keamanan
- Validasi masukan
- Tips keamanan untuk penyedia materi

14.1: Firebase dan AdMob

Materi:

Pelajaran ini membahas tiga topik:

- Firebase, serangkaian alat (bantu) untuk developer aplikasi web dan seluler
- Menghasilkan uang dari aplikasi Android Anda
- Menjalankan iklan menggunakan AdMob

Firebase

Firebase adalah serangkaian alat (bantu) untuk developer aplikasi, namun bukan hanya untuk developer aplikasi Android. Ini untuk developer aplikasi iOS dan juga developer aplikasi web. Akan tetapi, karena kursus ini adalah tentang development Android, pelajaran ini hanya membahas tentang cara menggunakan Firebase dengan aplikasi Android.

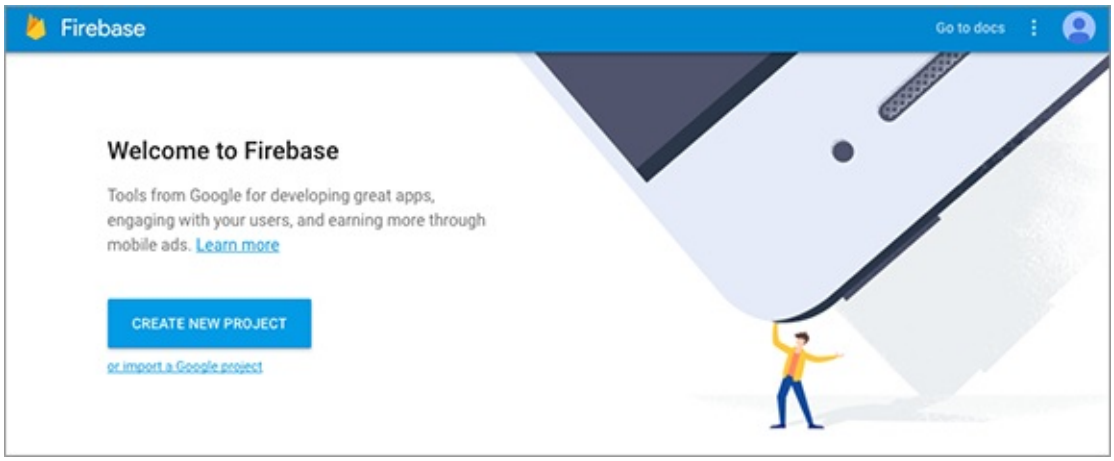
Sebagai developer Android, Anda menggunakan Android Studio untuk membangun aplikasi, namun Anda bisa menggunakan Firebase untuk menambahkan fitur ke aplikasi, mendapatkan pengguna aplikasi yang lebih luas, menguji aplikasi, menghasilkan pendapatan dari aplikasi, dan mendapatkan analisis tentang penggunaan aplikasi tersebut.

Bab ini tidak membahas segala sesuatu tentang Firebase, melainkan memperkenalkan Firebase, membantu Anda memulai penggunaannya, dan menyoroti fitur penting yang mungkin ingin digunakan.

Mulai dengan Firebase

Untuk menggunakan Firebase, buka Konsol Firebase di https://console.firebase.google.com/. Anda perlu memiliki akun Google.

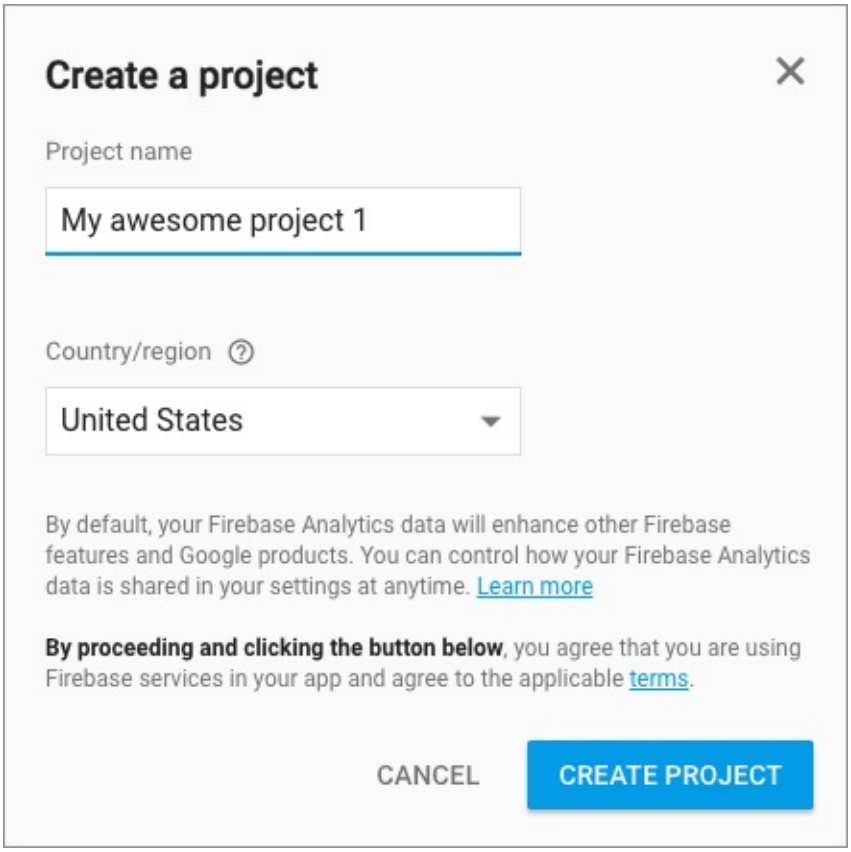
Pertama kali membuka Firebase, Anda akan melihat layar sambutan yang menyertakan tombol untuk membuat proyek baru.



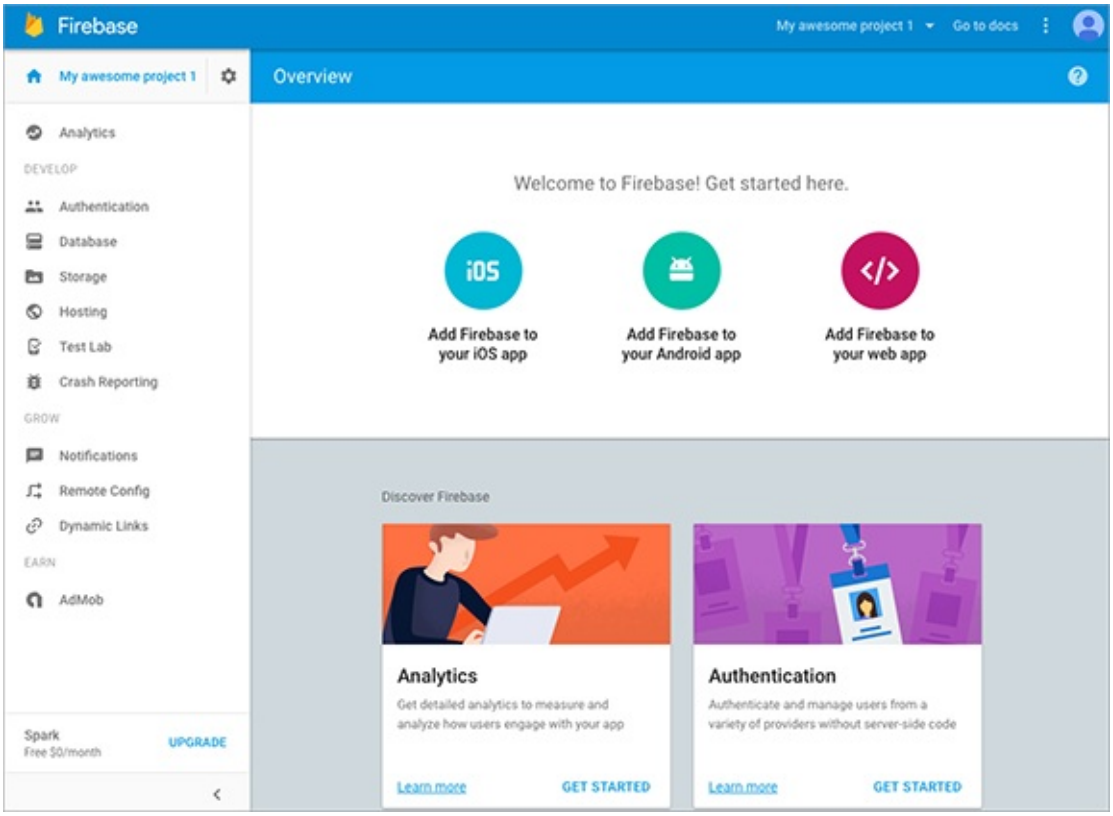
Untuk menggunakan fitur Firebase bersama aplikasi Android, pertama buat proyek Firebase, kemudian tambahkan aplikasi Android ke proyek Firebase.

Proyek adalah kontainer untuk aplikasi Anda di semua platform: Android, iOS, dan web. Anda bisa memberi nama proyek Anda apa pun yang diinginkan; tidak perlu mencocokkan nama aplikasi.

1. Di Konsol Firebase, klik tombol **CREATE NEW PROJECT**.
2. Masukkan nama proyek Firebase di kotak dialog yang muncul kemudian klik **Create Project.**



3. Konsol Firebase akan terbuka. Lihat di bilah menu untuk nama proyek Anda.



Tambahkan aplikasi Android ke proyek Firebase Anda

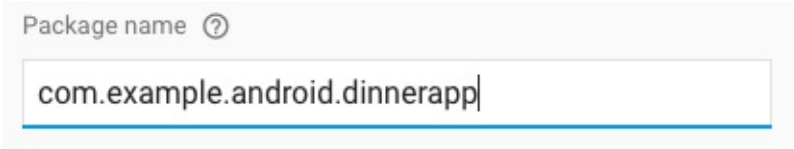
Langkah berikutnya adalah mengaitkan aplikasi Android dengan proyek Firebase Anda. Pertama, dapatkan informasi yang Anda perlukan. Untuk menghubungkan Firebase ke aplikasi Android, Anda perlu mengetahui nama paket yang digunakan di aplikasi. Sebaiknya buka dahulu aplikasi Anda di Android Studio sebelum memulai proses.

Untuk saling menghubungkan proyek Firebase dan aplikasi Android Anda:

1. Di Android Studio, buka aplikasi Android.
2. Pastikan Anda telah memasang Google Play Services terbaru.

   Tools > Android SDK Manager > SDK Tools tab > Google Play services.

3. Perhatikan paket kode sumber di aplikasi Android Anda.
4. Di konsol Firebase, klik **Add Firebase to your Android app**.
5. Masukkan nama paket aplikasi Anda di kotak dialog yang muncul.



6. Klik **Add App**. Firebase akan mengunduh file bernama `google_services.json` ke komputer Anda. Simpan di lokasi yang mudah dicari, misalnya di Desktop. (Jika Anda mengalami masalah, klik Project Settings (cogwheel di sebelah Overview) dan unduh file secara manual.)
7. Cari file yang telah diunduh di komputer Anda.
8. Salin atau pindahkan file itu ke dalam folder **app** proyek Anda di Android Studio. Wizard di Firebase akan menampilkan lokasi untuk memasukkan file di Android Studio.
9. Di konsol Firebase, klik **Continue**. Layar sekarang akan menampilkan petunjuk untuk memperbarui file build.gradle.
10. Di Android Studio, perbarui file build.gradle tingkat-proyek (Project: app) dengan versi terbaru paket google-services (x.x.x adalah nomor versi terbaru. Lihat Panduan persiapan Android untuk versi terbaru.)

```
buildscript {
  dependencies {
    // Add this line
    classpath 'com.google.gms:google-services:x.x.x'
  }
}
```

11. Di Android Studio, di file build.gradle tingkat-aplikasi (Modul: app), di bagian bawah, terapkan plugin google-services.

    `// Add to the bottom of the file apply plugin: 'com.google.gms.google-services'`

12. Di Android Studio, sinkronkan file Gradle.

13. Di konsol Firebase, klik **Finish**.

Aplikasi Android dan proyek Firebase Anda sekarang saling terhubung.

Firebase Analytics

Anda bisa mengaktifkan Firebase Analytics di aplikasi untuk melihat data tentang cara dan lokasi penggunaa aplikasi. Anda bisa melihat data seperti jumlah orang yang menggunakan aplikasi Anda dari waktu ke waktu dan lokasi mereka menggunakannya di seluruh dunia.

Semua data yang dikirim aplikasi ke Firebase dianonimkan, jadi Anda tidak pernah melihat identitas sebenarnya dari *orang* yang menggunakan aplikasi itu.

Untuk mendapatkan data penggunaan tentang aplikasi, tambahkan pustaka `firebase-core` ke aplikasi seperti berikut:

1. Pastikan Anda telah menambahkan aplikasi ke proyek Firebase.
2. Di Android Studio, pastikan Anda telah memasang Google Play services terbaru.

   Tools > Android SDK Manager > tab SDK Tools > Google Play services

3. Tambahkan dependensi untuk firebase-core ke file build.gradle tingkat-aplikasi (Modul: app):

   `compile 'com.google.firebase:firebase-core:x.x.x'`

Setelah menambahkan pustaka `firebase-core` ke aplikasi, maka data penggunaan secara otomatis dikirim bila orang menggunakan aplikasi Anda. Tanpa harus menambahkan kode lainnya, Anda akan mendapatkan serangkaian data default tentang cara, waktu, dan lokasi orang menggunakan aplikasi tersebut. Selain tidak perlu menambahkan kode untuk menghasilkan data penggunaan default, Anda bahkan tidak perlu mempublikasikan aplikasi atau melakukan hal lain selain menggunakannya.

Bila seseorang melakukan sesuatu di aplikasi Anda, seperti mengeklik tombol atau membuka aktivitas lain, aplikasi akan *membuat log kejadian Analytics*. Ini berarti aplikasi akan memaketkan data tentang kejadian yang berlangsung, dan memasukkannya dalam antrean untuk dikirim ke Firebase.

Periksa untuk mengetahui adanya kejadian Analytics

Android mengirim kejadian Analytics dalam beberapa batch untuk meminimalkan penggunaan jaringan dan baterai, seperti yang dijelaskan dalam entri blog ini. Secara umum, kejadian Analytics dikirim kurang-lebih setiap jam atau lebih ke server, namun juga memerlukan waktu tambahan bagi server Analytics untuk memproses data dan membuatnya tersedia untuk dilaporkan di konsol Firebase.

Sewaktu mengembangkan dan menguji aplikasi, Anda tidak harus menunggu data muncul di dasbor Analytics untuk memeriksa apakah aplikasi membuat log kejadian Analytics. Saat Anda menggunakan aplikasi, pastikan logcat menampilkan pesan "Debug". Di log tersebut, carilah pernyataan seperti ini:

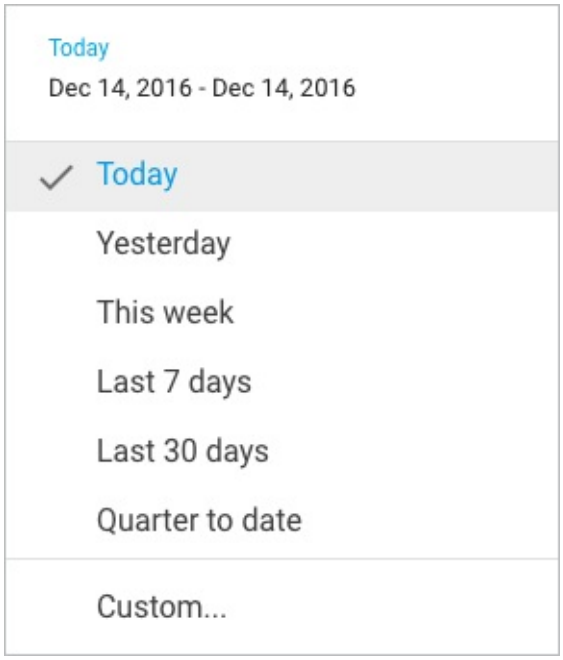
```
Logging event (FE): _e, Bundle[{_o=auto, _et=5388, _sc=SecondActivity, ...}]
```

Jenis pesan log ini menunjukkan bahwa kejadian Analytics telah dihasilkan. Dalam contoh ini, kejadian Analytics dihasilkan saat `SecondActivity` dimulai.

Tampilkan laporan di dasbor Firebase Analytics

Untuk melihat laporan Analytics, buka Konsol Firebase dan pilih **Analytics**. Dasbor akan dibuka dengan menampilkan laporan untuk aplikasi Anda selama 30 hari terakhir.

Catatan: Tanggal akhir selalu **Yesterday** secara default. Untuk melihat statistik penggunaan termasuk **Today**, ubah rentang tanggal default ke **Today**. Cari ikon kalender di bagian kanan atas layar dan ubah rentang tanggal.



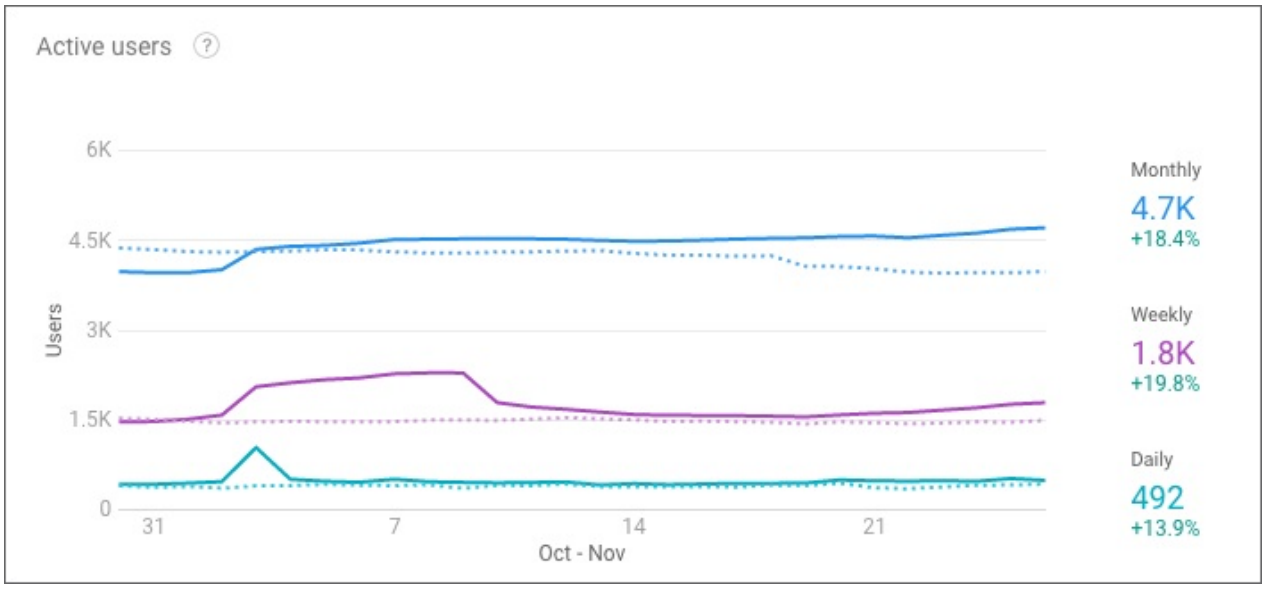
Analytics Default

Informasi analitik yang Anda dapatkan secara default menyertakan:

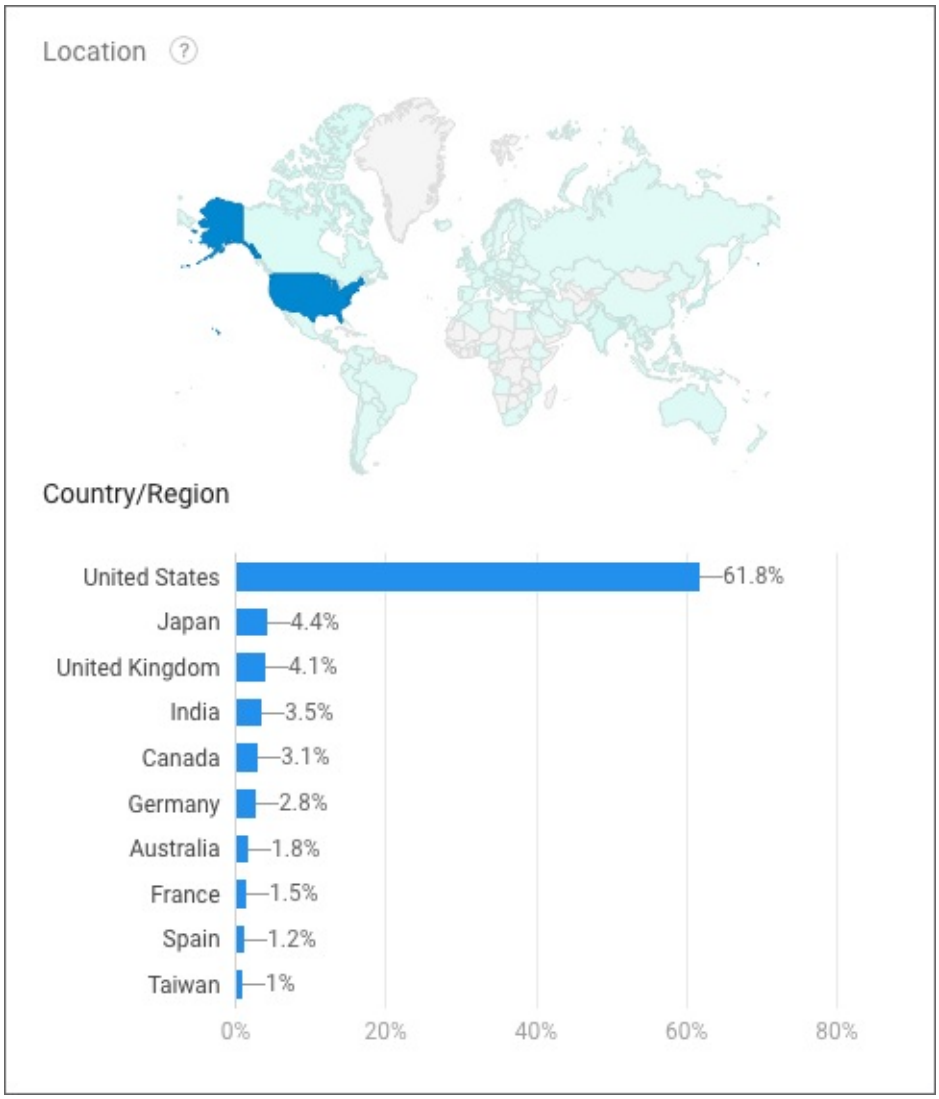
- Jumlah pengguna
- Perangkat yang digunakan pengguna
- Lokasi pengguna
- Demografi seperti jenis kelamin
- Versi aplikasi
- Interaksi pengguna
- Dan lainnya

Untuk daftar lengkap laporan yang tersedia, lihat bantuan Firebase untuk Dasbor.

Inilah contoh laporan yang menampilkan jumlah orang yang menggunakan aplikasi dalam 30 hari terakhir:



Inilah contoh laporan yang menampilkan lokasi pengguna:



Jenis data Analytics lainnya

Untuk melihat data penggunaan aplikasi di luar laporan default, Anda perlu menambahkan kode ke aplikasi untuk mengirim "kejadian Analytics" pada titik yang sesuai di aplikasi.

- Dalam aktivitas utama aplikasi, definisikan variabel instance `FirebaseAnalytics` untuk aplikasi.

  ```
  FirebaseAnalytics mFirebaseAnalytics;
  ```

- Di `onCreate()` aktivitas utama, dapatkan instance `FirebaseAnalytics`.

  ```
  mFirebaseAnalytics = FirebaseAnalytics.getInstance(this);
  ```

- Untuk mengirim data penggunaan di aplikasi, panggil `logEvent()` di instance `FireBaseAnalytics`.

Anda bisa membuat log kejadian Analytics untuk kejadian yang telah didefinisikan sebelumnya atau untuk kejadian khusus. Kejadian yang telah didefinisikan sebelumnya antara lain:

- ADD_PAYMENT_INFO
- ADD_TO_CART
- LEVEL_UP
- LOGIN
- SIGN_UP

dan banyak lagi.

Misalnya, untuk mengirim kejadian Analytics bila pengguna membuka level berikutnya di aplikasi, Anda bisa menggunakan kode seperti:

```
mFirebaseAnalytics.logEvent(LEVEL_UP, null);
```

Argumen kedua adalah `Bundle` berisi informasi yang menjelaskan kejadian.

Anda bisa mendalami kejadian yang telah didefinisikan sebelumnya dan parameternya dalam dokumentasi referensi `FirebaseAnalytics.Event` dan `FirebaseAnalytics.Param`.

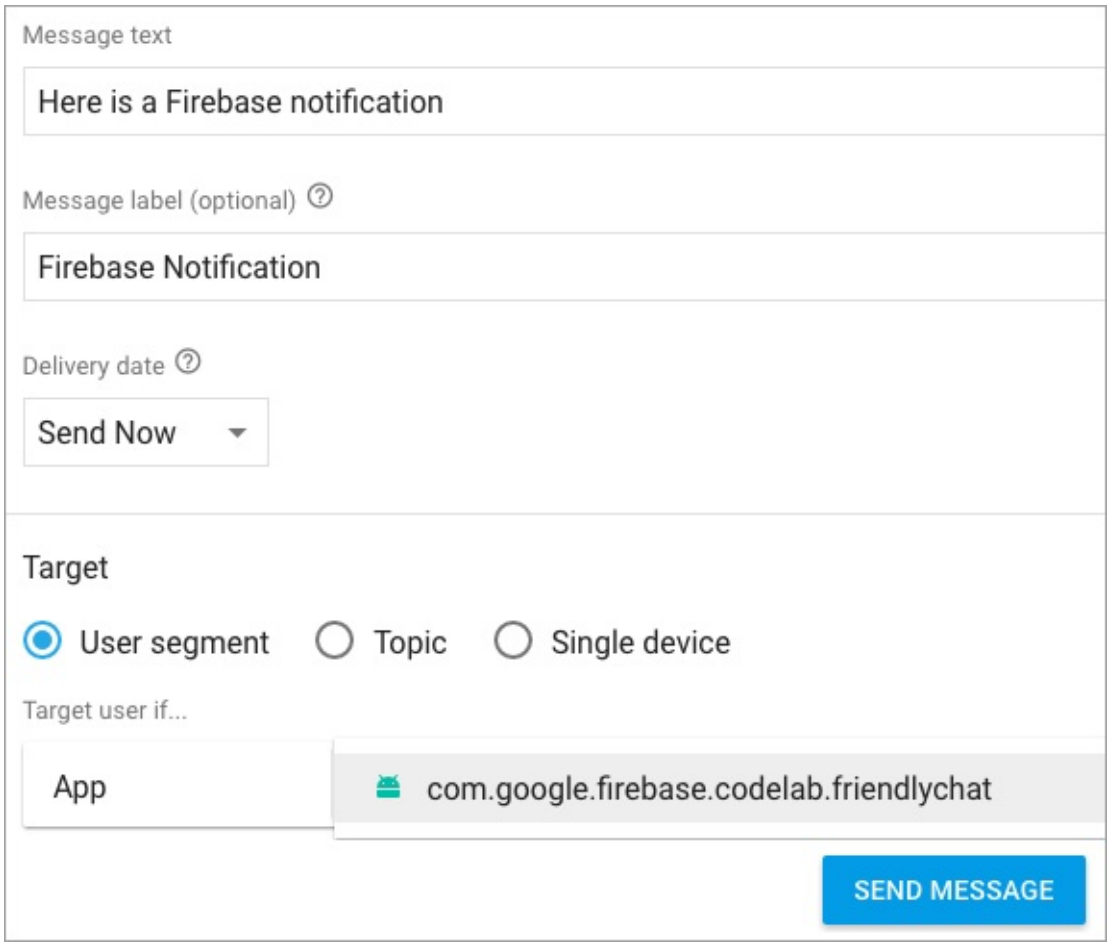
Baca selengkapnya tentang membuat log kejadian Analytics di panduan Pembuatan Log Kejadian Analytics.

Firebase Notifications

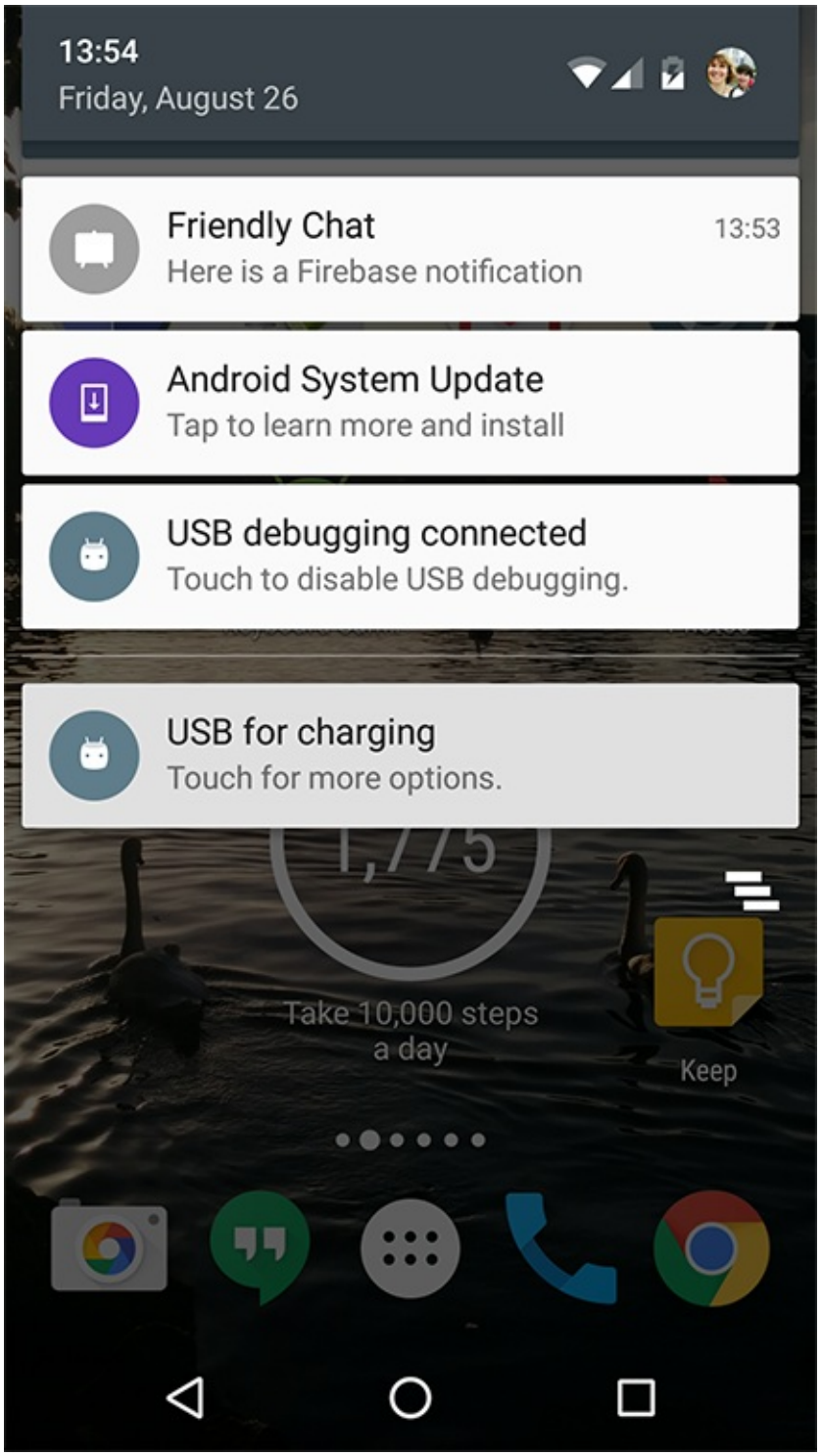
Anda telah mengetahui di pelajaran sebelumnya cara mengaktifkan *aplikasi* untuk mengirim notifikasi ke pengguna. Dengan menggunakan Konsol Firebase, *Anda* bisa mengirim notifikasi ke semua pengguna, atau ke subset pengguna.

Ketik pesan di bagian Notifications pada konsol, pilih segmen pengguna yang akan dikirimi pesan, kemudian kirimkan pesan.

Untuk mengirim pesan ke semua pengguna aplikasi, setel **User Segment** ke **App**, dan pilih paket untuk aplikasi Anda.



Di belakang layar, Firebase menggunakan Firebase Cloud Messaging untuk mengirim notifikasi ke perangkat yang ditargetkan, tempat aplikasi yang dipilih dipasang.



Praktik terbaik untuk mengirim notifikasi

Jenis praktik terbaik serupa berlaku untuk mengirimkan notifikasi dari Konsol Firebase saat digunakan untuk mengirimkan notifikasi dari aplikasi. Pertimbangkan pengguna. Kirim notifikasi yang penting saja. Jangan mengganggu pengguna dengan mengirimkan terlalu banyak notifikasi atau notifikasi dengan isi tidak membantu atau mengganggu.

Firebase Realtime Database

Dalam kursus ini, Anda telah mengetahui beragam cara aplikasi menyimpan data. Anda menggunakan `SharedPreferences` untuk menyimpan data sebagai pasangan nilai-kunci, dan menyimpan data di database SQLite bawaan Android. Anda membuat penyedia materi untuk menyediakan antarmuka agar bisa mengakses data yang disimpan, terlepas dari cara data disimpan, dan juga untuk berbagi data dengan aplikasi lain.

Namun bagaimana Anda berbagi data dengan seluruh klien seperti perangkat dan aplikasi yang berbeda, termasuk Android, iOS, dan aplikasi web, serta memungkinkan mereka memperbarui data dan semuanya tetap sinkron dengan data secara realtime? Karena inilah Anda memerlukan repositori data berbasis awan yang terpusat.

Firebase menawarkan database yang menyediakan storage data berbasis awan, yang memungkinkan klien untuk tetap sinkron saat data berubah. Jika aplikasi offline, data tetap tersedia. Bila aplikasi terhubung kembali ke Internet, data akan disinkronkan dengan keadaan database terbaru.

Untuk informasi selengkapnya, lihat panduan database Firebase di:

- firebase.google.com/docs/database/

Cara penyimpanan data

Firebase Realtime Database adalah database NoSQL. Data disimpan sebagai objek JSON. (Anda telah menggunakan JSON dalam pelajaran tentang menghubungkan ke Internet.)

Anda bisa menganggap database sebagai pohon JSON yang di-host di awan. Tidak seperti database SQL, tidak ada tabel atau catatan. Bila Anda menambahkan data ke pohon JSON, data akan menjadi simpul di struktur JSON yang ada bersama kunci terkait.

Inilah contoh pohon JSON data yang menyimpan data tentang buku dan film:

```
{
  "books": {
    "book one": {
      "title": "How to develop Android apps",
      "author": "Jane Developer"
    },
    "book two": {
      "title": "How to use Firebase",
      "author": "Adam Writer"
      },
    "book three": {
      "title": "How to search Google",
      "author": "Ava Searcher"
    }
  },
  "movies": {
    "movie one": {
      "title": "Saving the world",
      "main role": "Super man"
    },
    "movie two": {
      "title": "Saving the moon",
      "main role": "Bat girl"
      },
    "movie three": {
      "title": "Saving Mars",
      "main role": "Martian dog"
    }
   }
}
```

Inilah contoh bagaimana data ini muncul di konsol Firebase Database:

Bila menulis kode untuk mengakses data di database, Anda mengambil item data berdasarkan lokasinya. Lokasi data dibuat dengan melintasi pohon.

Misalnya, lokasi ke:

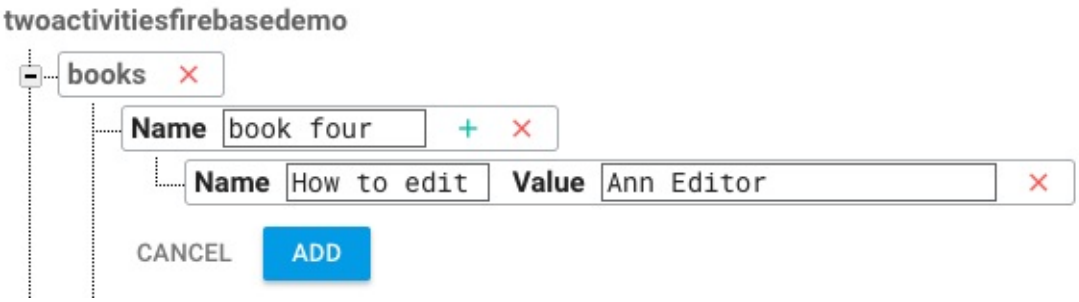
- simpul **books** adalah twoactivitiesfirebasedemo/database/data/books
- **book one** adalah twoactivitiesfirebasedemo/database/data/books/book%20one
- **author untuk book one** adalah twoactivitiesfirebasedemo/database/data/books/book%20one/author

Mendapatkan dan menyetel data

Anda bisa menggunakan Konsol Firebase untuk menampilkan dan memperbarui data di database. *Aplikasi Anda* bisa menggunakan panggilan API untuk mendapatkan dan menyetel data di database.

Menampilkan dan menulis data di Konsol Firebase

Di Konsol Firebase, Anda bisa menambahkan, mengedit, dan menghapus data.



Anda juga bisa mengimpor dan mengekspor data di konsol. Untuk mengimpor data, Anda harus memformat data sebagai JSON dan menyimpannya di file dengan ekstensi `.json`.

Hati-hati jika Anda mengimpor file JSON, karena akan **menggantikan** simpul data apa pun yang didefinisikan di file yang sudah ada di database.

Membaca and menulis data di aplikasi Anda

Untuk menggunakan database Firebase di aplikasi, tambahkan pustaka `firebase-database` ke aplikasi Anda, kemudian gunakan metode di `DatabaseReference`:

1. Pastikan Anda sudah menambahkan aplikasi ke proyek Firebase.
2. Tambahkan dependensi untuk database firebase ke file build.gradle tingkat-aplikasi (Modul:app) (dengan x adalah versi terbaru, lihat panduan developer Database Firebase Android untuk versi terbaru): `compile 'com.google.firebase:firebase-database:x.x.x'`
3. Untuk mengakses data dari aplikasi Android, gunakan metode di kelas `DatabaseReference`.

Cara termudah untuk membuat dan menangani data adalah dengan membuat objek Java yang menyatakan data itu. Misalnya, Anda bisa membuat kelas `Book`:

```
public class Book {

    public String title;
    public String author;

    public Book() {
        // Default constructor is required
    }

    public Book(String title, String author) {
        this.title = title;
        this.author = author;
    }

}
```

Untuk mengakses data dari aplikasi Android, gunakan metode di kelas `DatabaseReference` untuk mengakses, membuat, dan memperbarui data. Kelas `DatabaseReference` menyatakan referensi ke item data.

Misalnya:

- `child()` mengakses simpul anak.
- `setValue()` menyetel data pada simpul, dan membuat simpul jika belum ada.

Untuk membuat simpul data baru:

1. Dapatkan referensi ke induk simpul data baru
2. Panggil `child()` untuk membuat referensi ke simpul anak baru.
3. Gunakan `setValue()` untuk menyetel nilai simpul baru.

Misalnya, anggaplah Anda ingin menambahkan buku baru ke database bersama detail berikut:

- bookId = "book 4"
- title = "How to edit data"
- author = "Ann Editor"

Untuk menambahkan buku baru ini ke serangkaian data contoh yang ditampilkan sebelumnya, buat instance `DatabaseReference`, dan lakukan inisialisasi di metode `onCreate()` aktivitas utama:

```
private DatabaseReference mDbBooksRef;
@Override
protected void onCreate(Bundle savedInstanceState) {
  // Create a new reference to the "books" node
  mDbBooksRef = FirebaseDatabase.getInstance().getReference("books");
  // rest of onCreate() …
}
```

Definisikan metode untuk membuat objek `Book` baru dan tambahkan ke database pada simpul "books":

```
private void addNewBook(String bookId, String title, String author) {
    // Create a new Book object with the relevant details
    Book book = new Book(title, author);
    // mDbBooksRef is a reference to the "books" node
    // Create a child node whose key is bookId
    // and set its value to the new book
    mDbBooksRef.child(bookId).setValue(book);
}
```

Panggil metode `addBook()` seperti berikut untuk membuat item data baru di database:

```
addNewBook("book four", "How to edit data", "Ann Editor");
```

Untuk mengetahui selengkapnya tentang membaca dan menulis data Firebase di aplikasi, lihat:

- Persiapkan Firebase Realtime Database untuk Android
- Membaca dan Menulis Data di Android.

Kontrol akses

Firebase menyediakan serangkaian aturan untuk menentukan siapa yang diizinkan untuk menampilkan dan memperbarui data di database Firebase. Pelajari cara membuat aturan di Mulai dengan aturan Database.

Firebase Test Lab

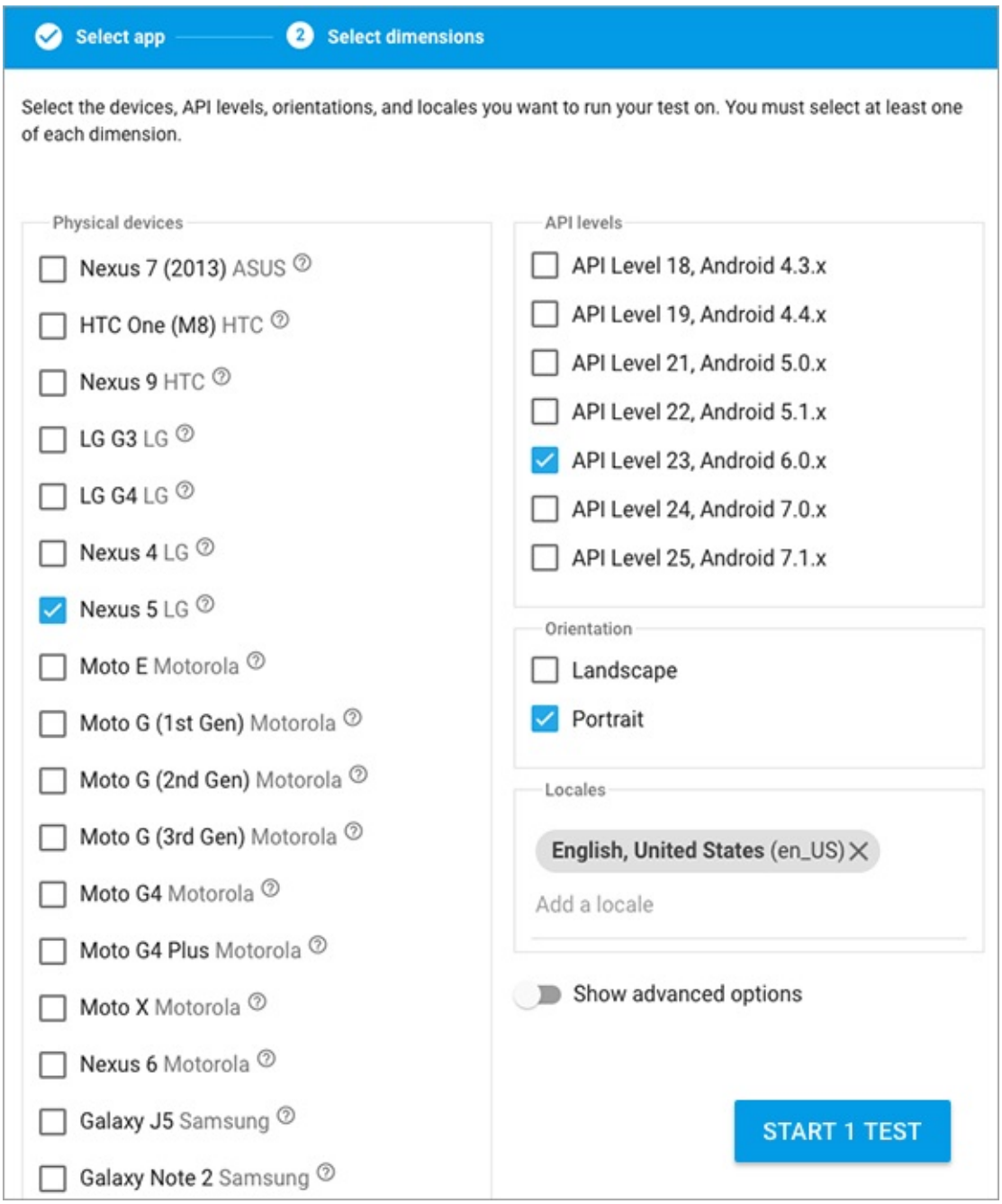
Firebase Test Lab memungkinkan Anda menguji aplikasi pada berbagai macam perangkat yang di-host di pusat data Google dan mendapatkan laporan hasilnya.

Test Lab membuat pengujian yang disebut **pengujian robo** untuk Anda, dan Anda juga bisa membuat dan menjalankan pengujian sendiri. Untuk menjalankan aplikasi Anda di Test Lab:

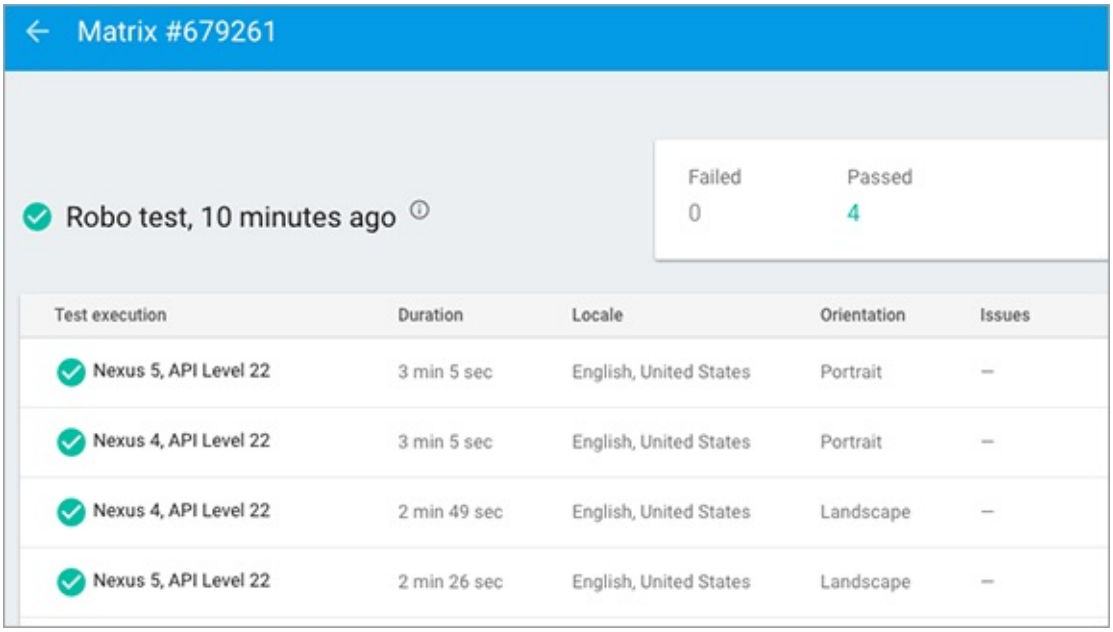
1. Di Android Studio: **Build** > **Build APK.**
2. Cari APK di sistem file komputer Anda. APK mungkin berada di `app/build/apk` atau `app/build/outputs/apk` di direktori tempat menyimpan proyek Android Studio.

   Di Mac, Anda bisa mengeklik **Reveal in Finder** untuk melihat lokasi APK.

3. Di Konsol Firebase, pilih **Test Lab** > **Run a Test**.
4. Unggah APK Anda.
5. Dari matriks perangkat dan API level yang muncul, pilih kombinasi perangkat yang akan menjalankan aplikasi.



6. Pilih **Start N Tests** di bagian kanan bawah matriks perangkat, dengan N adalah kombinasi valid untuk perangkat dan API level yang dipilih.
7. Tampilkan laporan bila pengujian selesai.



Seperti halnya menjalankan pengujian "robo" yang dibuat secara otomatis, Anda juga bisa menjalankan pengujian instrumentasi sendiri, yakni pengujian yang Anda tulis secara khusus untuk menguji aplikasi Anda. Misalnya, Anda bisa menjalankan pengujian Espresso di Test Lab.

Bila Anda menulis pengujian instrumentasi, buat APK kedua untuk mengunggah ke Test Lab bersama APK aplikasi.

Ketahui selengkapnya di Ringkasan Firebase Test Lab for Android.

Demo Firebase

Ada proyek demo Firebase umum yang bisa Anda gunakan untuk menjelajahi Konsol Firebase. Proyek demo Firebase adalah proyek Firebase standar dengan analitik yang berfungsi penuh, pelaporan kerusakan, lab pengujian, dan banyak lagi. Siapa saja yang memiliki akun Google bisa mengaksesnya. Inilah cara bagus untuk melihat data aplikasi sebenarnya dan menyusuri rangkaian fitur Firebase.

Anda tidak bisa mengirim notifikasi dari proyek demo karena tidak memiliki akses pemilik. Anda juga tidak bisa melihat dan memodifikasi data di database.

Data di proyek demo adalah data sungguhan dari aplikasi yang disebut Flood-It. Anda bisa mengunduh aplikasi ini dan menggunakannya untuk berkontribusi ke data sendiri. Flood-It adalah permainan sederhana, di sini akan terlihat seberapa cepat Anda bisa menutupi papan dengan satu warna. Flood-It dibuat oleh Lab Pixies, yang saat ini merupakan perusahaan milik Google.

Lanjutkan dan unduh serta mainkan Flood-It jika suka, namun jangan lupa kembali dan terus belajar! Dapatkan aplikasi Flood-It dari Google Play di sini.

Ketahui cara mengakses demo Firebase di pusat bantuan Firebase.

Fitur Firebase selengkapnya

Firebase memiliki lebih banyak alat untuk membantu Anda mengembangkan aplikasi dan menjangkau pengguna. Akan tetapi, tujuan kami membahas Firebase di kursus ini adalah memperkenalkan Firebase dan membuat Anda mengetahui fiturnya, serta memberi Anda inspirasi untuk mengetahui selengkapnya tentang Firebase dengan cara sendiri.

Untuk mulai mempelajari Firebase sendiri:

- Ikuti kursus online "Firebase in a Weekend" www.udacity.com/course/ud0352
- Bekerja melalui Firebase codelab codelabs.developers.google.com/codelabs/firebase-android

Ketahui selengkapnya tentang Firebase

Mulai dengan Firebase

- Konsol Firebase
- Tambahkan Firebase ke proyek Android Anda
- Demo umum Firebase
- Kursus online Firebase in a Weekend
- Firebase codelab

Firebase Analytics

- Tambahkan Firebase Analytics ke aplikasi Android Anda
- Metrik kunci yang tersedia di dasbor Firebase Analytics
- Kejadian yang telah didefinisikan sebelumnya: FirebaseAnalytics.Event
- Parameter yang telah didefinisikan sebelumnya: FirebaseAnalytics.Param

Firebase Notifications

- Firebase Notifications

Database Firebase

- Database Firebase
- Persiapkan Firebase Realtime Database untuk Android
- Membaca dan Menulis Data di Android

Firebase Test Lab

- Firebase Test Lab for Android
- Gunakan Firebase Test Lab for Android dari Konsol Firebase

Hasilkan uang dari aplikasi Anda

Membuat aplikasi dan melihatnya berjalan adalah hal menarik. Namun bagaimana cara menghasilkan uang dari aplikasi Anda?

Pertama, Anda perlu membuat aplikasi yang bekerja dengan baik, cukup cepat, tidak mogok, dan berguna atau menghibur. Aplikasi harus menarik sehingga pengguna tidak hanya ingin memasang dan menggunakannya, melainkan ingin terus menggunakannya.

Dengan anggapan aplikasi Anda sudah sempurna dan menyediakan fitur berguna, menghibur, atau menarik, ada beragam cara menghasilkan uang dari aplikasi tersebut.

Cara menghasilkan uang

Model monetisasi:

- Model Premium—pengguna membayar untuk mengunduh aplikasi.
- Model Freemium:
  - gratis mengunduh aplikasi.
  - pengguna membayar untuk peningkatan versi atau pembelian dalam aplikasi.
- Berlangganan—pengguna membayar biaya berkala untuk aplikasi.
- Periklanan—aplikasi gratis namun menampilkan iklan.

Aplikasi premium

Pengguna membayar di depan untuk mengunduh aplikasi premium. Untuk aplikasi yang menyediakan fungsiolitas yang diinginkan kepada sedikit pengguna yang sangat ditargetkan, menyertakan harga ke aplikasi bisa memberikan sumber pendapatan. Ketahuilah bahwa sebagian pengguna akan menolak mengunduh aplikasi jika harus membayar, atau jika tidak bisa mencobanya terlebih dahulu secara gratis. Jika pengguna bisa menemukan aplikasi lainnya yang serupa dan tersedia gratis atau lebih rendah, mereka mungkin lebih memilih untuk mengunduh dan mencoba tersebut.

Aplikasi freemium

Aplikasi "freemium" adalah kompromi antara aplikasi yang sepenuhnya gratis dengan aplikasi yang mengenakan biaya pemasangan. Aplikasi tersebut tersedia untuk pemasangan gratis, baik dengan fungsionalitas terbatas maupun selama durasi terbatas. Sasaran Anda untuk aplikasi freemium seharusnya untuk meyakinkan pengguna terhadap nilai aplikasi tersebut, sehingga setelah menggunakannya untuk sementara waktu, mereka akan bersedia membayar agar tetap bisa menggunakannya atau untuk meningkatkan versi demi mendapatkan fitur lainnya. Pernahkah Anda mengunduh aplikasi gratis, kemudian membayar untuk meningkatkan versi demi fungsionalitas? Apa yang Anda sukai pada aplikasi tersebut sehingga bersedia membayarnya?

Cara lain untuk menghasilkan uang dari aplikasi freemium adalah menyediakan pembelian dalam aplikasi. Untuk game, aplikasi Anda mungkin menawarkan level materi baru dalam game, atau item baru untuk membuatnya jadi lebih menyenangkan. Pikirkan tentang game seluler yang pernah Anda mainkan. Manakah yang menawarkan pembelian dalam aplikasi dan untuk apa? Pernahkah Anda membuat pembelian dalam aplikasi di aplikasi seluler?

Berlangganan

Pada model berlangganan, pengguna membayar biaya berkala untuk menggunakan aplikasi. Ini sangat mirip dengan model premium, hanya saja pengguna membayar pada siklus penagihan teratur, bukan sekali saja pada waktu pemasangan. Anda bisa mempersiapkan langganan agar pengguna membayar setiap bulan atau setiap tahun. Jika Anda menyediakan aplikasi secara berlangganan, pertimbangkan penawaran pembaruan materi teratur atau beberapa layanan lainnya yang menjamin pembayaran berulang.

Jika Anda memutuskan untuk menawarkan aplikasi dengan model berlangganan, sebaiknya biarkan pengguna mencobanya secara gratis. Banyak pengguna akan menolak memasang aplikasi yang membuat mereka harus membayar sebelum mencoba untuk melihat apakah aplikasi itu sesuai dengan kebutuhan.

Periklanan

Salah satu strategi monetisasi umum adalah menyediakan aplikasi gratis, namun menjalankan iklan di dalamnya. Pengguna bisa menggunakan aplikasi Anda sebanyak yang mereka suka, namun sesekali aplikasi akan menampilkan iklan.

Jika aplikasi Anda menampilkan iklan, selalu pertimbangkan pengguna. Jika aplikasi Anda menampilkan begitu banyak iklan yang mengganggu pengguna, mereka mungkin berhenti menggunakannya atau mencopot pemasangannya.

Mudah sekali menambahkan iklan ke aplikasi. Cara terbaik untuk memasukkan iklan ke dalam aplikasi Anda adalah menggunakan AdMob.

AdMob

Google menyediakan alat bagi pengiklan untuk membuat iklan dan mendefinisikan kriteria target iklan mereka. Untuk contoh kriteria target, iklan dapat ditargetkan untuk ditampilkan kepada orang-orang di lokasi tertentu. Google memiliki banyak persediaan iklan untuk ditampilkan di situs web dan aplikasi seluler. Anda bisa menampilkan iklan dari persediaan ini di aplikasi dengan menggunakan AdMob (singkatan dari Ads on Mobile).

Untuk menampilkan iklan di aplikasi, tambahkan `AdView` di layout aktivitas dan tulis sedikit kode boilerplate untuk memuat iklan. Bila pengguna menjalankan aplikasi Anda dan membuka aktivitas, iklan itu akan muncul di `AdView`. Anda tidak perlu mempersoalkan pencarian iklan yang akan ditampilkan karena Google yang akan menanganinya.

Cara iklan membantu Anda menghasilkan uang

Google membayar Anda bila pengguna mengeklik iklan di aplikasi Anda. Jumlah persis yang dibayarkan kepada Anda bergantung pada iklan yang ditampilkan, dan seberapa banyak pengiklan bersedia membayar untuk iklan mereka. Jumlah total yang dibayarkan oleh pengiklan dibagi antara Google dan penerbit situs web atau aplikasi tempat iklan itu muncul.

Jangan mengeklik iklan di aplikasi Anda sendiri

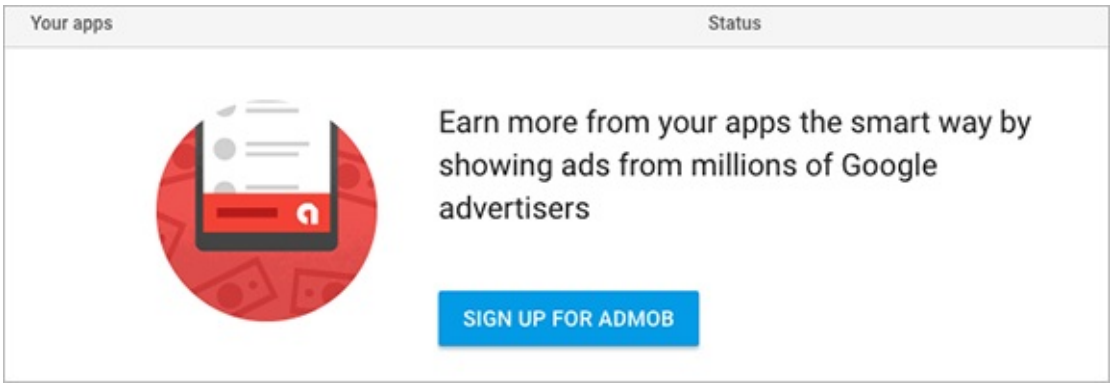
Google memiliki kebijakan yang mencegah penerbit situs web dan penerbit aplikasi mengeklik iklan di situs web dan aplikasi mereka sendiri. Pengiklan membayar bila orang mengklik iklan mereka, sehingga tidak adil bila Anda menampilkan iklan di aplikasi sendiri, kemudian mengekliknya, dan menyebabkan pengiklan membayar Anda karena mengklik iklan di aplikasi sendiri.

Baca selengkapnya tentang kebijakan AdMob di pusat bantuan AdMob.

Buat akun AdMob

Sebelum bisa bereksperimen dengan menjalankan iklan di aplikasi, Anda perlu mengaktifkan AdMob untuk aplikasi itu. Untuk mengaktifkan AdMob, gunakan langkah-langkah ini:

1. Di Konsol Firebase, pilih **AdMob** di navigasi kiri, kemudian pilih **Sign Up For AdMob**.



   Anda akan dialihkan ke konsol AdMob.
2. Ikuti wizard pendaftaran untuk membuat akun AdMob dan menambahkan aplikasi ke AdMob.

Untuk menampilkan iklan di aplikasi, Anda memerlukan ID aplikasi AdMob dan ID unit iklan. Anda bisa mendapatkan keduanya di konsol AdMob.

Implemantasikan AdMob di aplikasi Anda

Untuk menampilkan iklan di aplikasi:

1. Pastikan Anda telah menambahkan aplikasi ke proyek Firebase.
2. Tambahkan dependensi untuk iklan firebase ke file build.gradle level aplikasi (Modul: app): (dengan x adalah versi terbaru) `compile 'com.google.firebase:firebase-ads:x.x.x'`
3. Tambahkan `AdView` ke layout untuk aktivitas yang akan menampilkan iklan.
4. Lakukan inisialisasi iklan AdMob di peluncur aplikasi, dengan memanggil `MobileAds.initialize()` di metode `onCreate()` aktivitas utama Anda.
5. Perbarui `onCreate()` aktivitas itu untuk memuat iklan ke dalam `AdView` .

Bersiaplah menjalankan pengujian iklan

Selagi mengembangkan dan menguji aplikasi, Anda bisa menampilkan dan menguji iklan untuk memastikan aplikasi telah dipersiapkan dengan benar untuk menampilkan iklan. Saat menguji iklan Anda perlu:

- ID (EMEI) perangkat Anda untuk menjalankan pengujian iklan. Untuk mendapatkan ID perangkat:
  - Buka **Settings** > **About phone** > **status> IMEI.**
  - Panggil nomor ***#06#**.
  - ID aplikasi AdMob Anda. Dapatkan di konsol AdMob.
  - ID unit iklan. Dapatkan di konsol AdMob.

Tambahkan AdView untuk menampilkan iklan

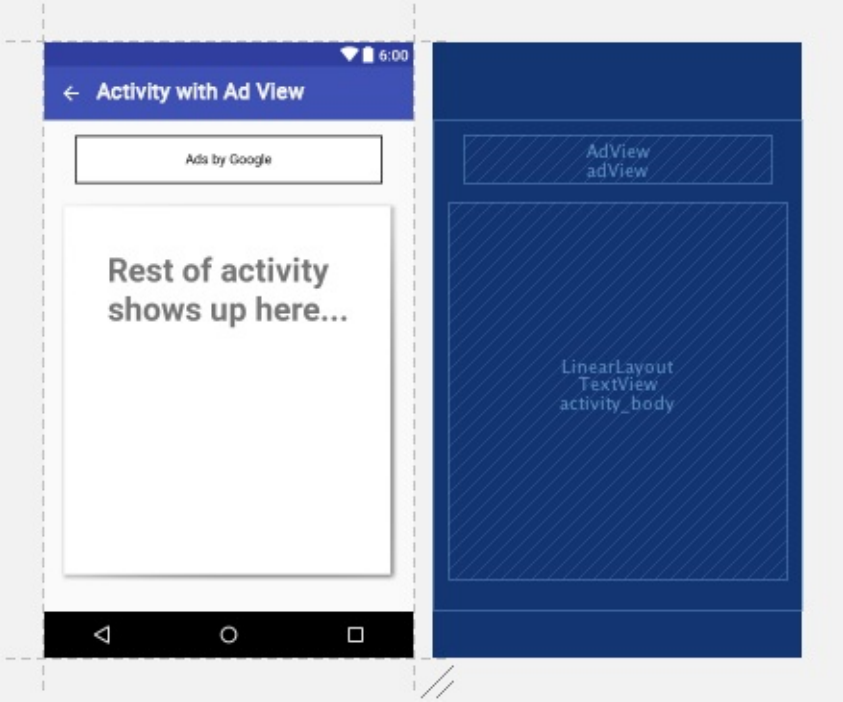
Di file layout aktivitas yang akan memunculkan iklan, tambahkan `AdView` :

```
<com.google.android.gms.ads.AdView
    android:id="@+id/adView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_centerHorizontal="true"
    ads:adSize="BANNER"
    ads:adUnitId="@string/banner_ad_unit_id">
</com.google.android.gms.ads.AdView>
```

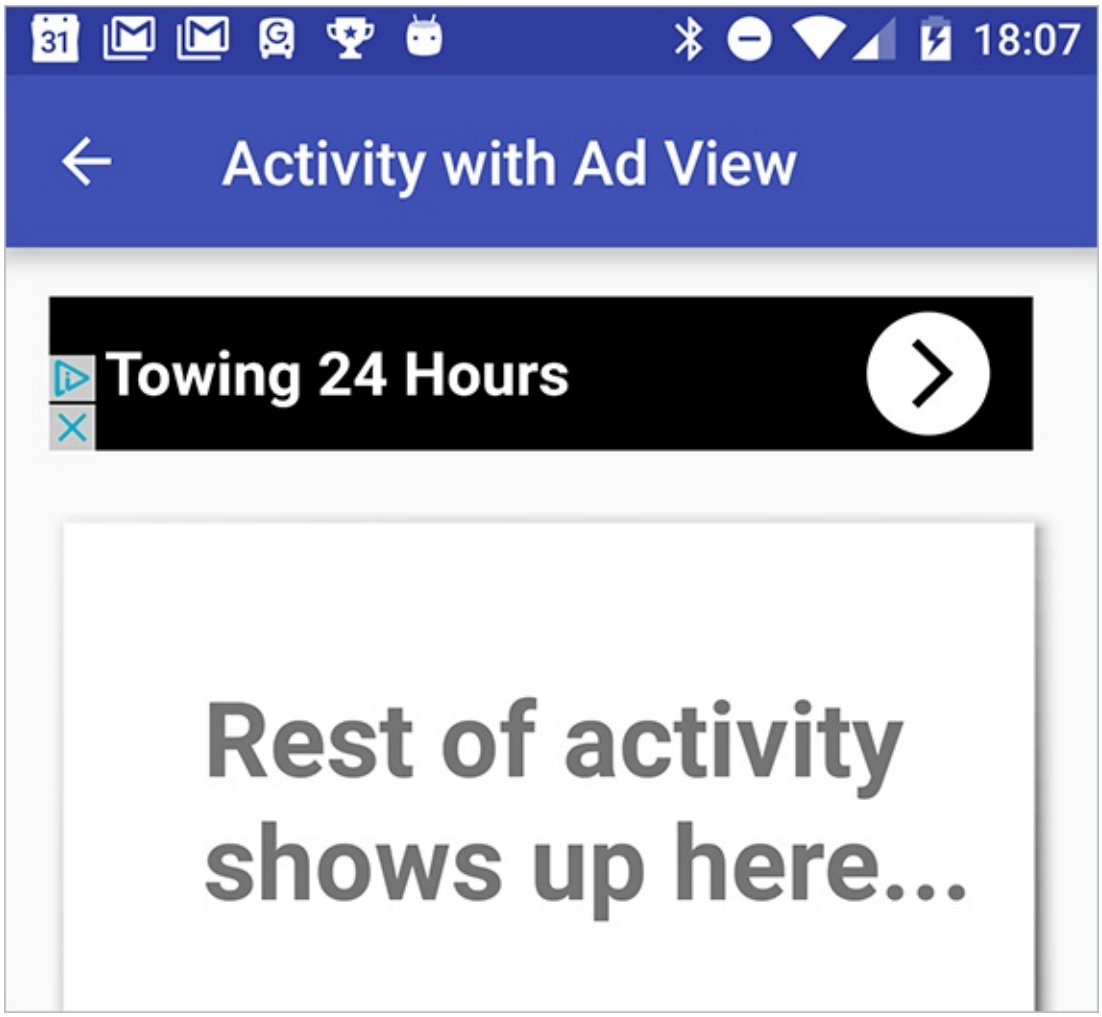
Anda juga perlu menambahkan namespace `ads` ke tampilan akar layout:

```
<RootLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:ads="http://schemas.android.com/apk/res-auto"
    …>
```

Inilah contoh layout dengan `AdView` di Layout Editor:



Bila aplikasi dijalankan, AdView akan diganti dengan iklan.



Lakukan Inisialisasi MobileAds

Kelas `MobileAds` menyediakan metode untuk melakukan inisialisasi iklan AdMob di aplikasi Anda. Panggil `MobileAds.initialize()` di metode `onCreate()` aktivitas utama, dengan meneruskan konteks dan ID aplikasi (yang diperoleh dari konsol AdMob).

```
// Initialize AdMob
MobileAds.initialize(this, "ca-app-pub-1234");
```

Tulis kode untuk memuat iklan

Di metode `onCreate()` aktivitas yang menampilkan iklan, tulis kode untuk memuat iklan.

Selagi mengembangkan dan menguji aplikasi, Anda bisa menampilkan iklan dalam mode pengujian dengan menetapkan perangkat pengujian khusus. Untuk menampilkan iklan di perangkat sendiri, Anda memerlukan ID (IMEI) perangkat, yang bisa diperoleh dari **Settings > About phone > Status> IMEI** atau dengan memanggil ***#06#**.

Untuk memuat iklan:

1. Dapatkan `AdView` tempat iklan akan muncul.
2. Buat `AdRequest` untuk meminta iklan.
3. Panggil `loadAd()` di `AdView` untuk memuat iklan ke dalam `AdView`.

Inilah kode yang harus dituliskan dalam `onCreate()` di aktivitas:

```
// Get the AdView
AdView mAdView = (AdView) findViewById(R.id.adView);
// Create an AdRequest
AdRequest adRequest = new AdRequest.Builder()
  // allow emulators to show ads
  .addTestDevice(AdRequest.DEVICE_ID_EMULATOR)
  // allow your device to show ads
  .addTestDevice("1234") // your device id
  .build();
// Load the ad into the AdView
mAdView.loadAd(adRequest);
```

Bila sudah siap menjalankan iklan sebenarnya, Anda harus membuat beberapa perubahan pada cara Anda membuat objek AdRequest . Lihat Mulai dengan Android AdMob untuk informasi selengkapnya mengenai tindakan yang perlu dilakukan.

Ketahui selengkapnya tentang AdMob

- Hasilkan uang dari aplikasi Android Anda
  - Raih pendapatan dari iklan AdMob
- Mulai dengan AdMob di Android Studio
- Persiapkan Android untuk AdMob
- Mendaftar AdMob
- Akses konsol AdMob setelah mendaftar
- Anda juga bisa mengakses konsol AdMob dari tautan AdMob di Konsol Firebase)
- Pusat bantuan AdMob
- Kebijakan AdMob
  - Kebijakan Anti-spam (jangan mengeklik iklan di aplikasi Anda sendiri)
- Daftar Istilah AdMob

15.1: Publikasikan!

Materi:

Dalam praktik sebelumnya, Anda telah mempelajari tentang cara membangun dan menguji aplikasi, dan sekarang saatnya mempelajari cara mempublikasikannya. Jadi, apa yang harus Anda lakukan untuk mempublikasikan aplikasi? Bab ini membahas langkah-langkah tingkat tinggi untuk mempublikasikan aplikasi Android ke Google Play Store, serta memperkenalkan Google Play Developer Console, tempat Anda bisa mengunggah aplikasi ke Google Play. Bab ini ***tidak*** mengajari Anda segala sesuatu yang perlu diketahui tentang Google Play Developer Console. Namun kami berharap, setelah membaca bab ini, Anda akan tertarik untuk mengunggah aplikasi dan mendalami semua fitur konsol yang berbeda.

Bab ini memiliki dua bagian utama:

- **Siapkan aplikasi Anda untuk dirilis** mendiskusikan tugas yang perlu dilakukan untuk memastikan bahwa aplikasi Anda benar-benar siap dipublikasikan.
- **Publikasikan!** mendiskusikan pengujian alfa dan beta, serta cara menggunakan Google Play Developer Console untuk mempublikasikan aplikasi Anda.

Siapkan aplikasi Anda untuk dirilis

Sasaran utama saat mempersiapkan rilis aplikasi Anda adalah untuk memastikan bahwa aplikasi ***benar-benar*** siap. Pengguna Android mengharapkan aplikasi berkualitas tinggi yang terlihat menarik dan bekerja dengan baik. Akan semakin banyak fitur yang ingin Anda tambahkan ke aplikasi, atau semakin banyak fitur yang diminta pengguna. Namun ada perbedaan besar antara aplikasi hebat yang bisa dibuat lebih baik lagi dengan aplikasi yang sering rusak, menyediakan pengalaman yang tidak lengkap, memiliki jalur aplikasi yang tidak jelas, atau tidak memiliki cara untuk menuju ke bagian penting di aplikasi.

Pikirkan tentang aplikasi favorit Anda. Apa yang Anda sukai pada aplikasi favorit tersebut?

Pikirkan tentang aplikasi yang telah Anda copot pemasangannya atau yang hampir tidak pernah digunakan. Apa yang tidak Anda sukai pada aplikasi tersebut? Pikirkan tentang aplikasi yang tertarik ingin sekali Anda dapatkan, namun kemudian kecewa ketika mulai menggunakannya. Apa penyebab kekecewaan Anda?

Tugas tingkat tinggi untuk mempublikasikan aplikasi Anda ke Google Play Store adalah:

1. Siapkan aplikasi untuk dirilis.
2. Buat APK bertandatangan.
3. Unggah APK ke Google Play Developer Console.
4. Jalankan pengujian alfa dan beta.
5. Publikasikan ke seluruh dunia!

Bab ini membahas tampilan tingkat tinggi di setiap tugas ini.

Apa yang dimaksud dengan APK?

APK adalah file zip yang berisi segala sesuatu yang diperlukan aplikasi Anda untuk berjalan di perangkat pengguna. APK selalu memiliki ekstensi **.apk**. Anda memerlukan APK untuk mempublikasikan aplikasi di Google Play Store.

Anda bisa menggunakan Android Studio untuk membuat APK aplikasi. Sebelum membuat APK untuk aplikasi, Anda perlu melakukan segala sesuatu yang Anda bisa untuk membuat aplikasi berhasil, termasuk:

- Uji aplikasi Anda secara menyeluruh.
- Pastikan aplikasi Anda memiliki filter yang tepat.
- Tambahkan ikon.
- Pilih ID Aplikasi.
- Tetapkan API level target.
- Bersihkan aplikasi Anda.

Bila aplikasi benar-benar siap, Anda bisa mengunggah APK bertandatangan ke Google Play Store.

Lihat daftar periksa peluncuran dalam dokumentasi developer Android.

Uji aplikasi Anda secara menyeluruh

Saat Anda mengembangkannya, ujilah aplikasi di perangkat Android milik sendiri dan di emulator Android Studio. Pastikan Anda menguji aplikasi untuk orientasi dan ukuran layar yang berbeda. Selain itu periksa apakah aplikasi bekerja dengan benar di perangkat lama.

Bagikan aplikasi Anda pada sesama teman developer. Buat file zip dan kirim ke developer lainnya. Kemudian mereka bisa memuat aplikasi Anda ke Android Studio dan menjalankannya.

Setelah mengetahui cara kerja aplikasi seharusnya; Anda mendesain dan membangunnya. Anda mengetahui "cara yang tepat" untuk menggunakan aplikasi tersebut. Akan tetapi, Anda tidak akan percaya bahwa pengguna memiliki cara yang benar-benar kreatif untuk mencoba menggunakan aplikasi, jadi pertimbangkanlah. Mereka mungkin mencoba menggunakan aplikasi atau fiturnya dengan cara yang tidak terpikirkan oleh Anda, atau mungkin mengujinya dengan cara yang belum pernah Anda gunakan. Dorong penguji untuk menguji aplikasi dengan cara berbeda, mencoba mencapai sasaran berbeda, dan menggunakan jalur berbeda saat menyusuri seluruh aktivitas. Hal ini akan membantu menangkap kesalahan, inkonsistensi, atau kegagalan fungsionalitas yang tidak bisa Anda temukan karena sudah sangat familier dengan cara kerja aplikasi yang demikian.

Pastikan Anda menjalankan pengujian formal pada aplikasi, termasuk pengujian unit dan Espresso. Pengujian ini harus mencakup fitur inti aplikasi, dan poin integrasi utama tempat aplikasi Anda memanggil API lain atau mendapatkan kembali data dari web. Inilah poin penting untuk aplikasi Anda, dan mungkin poin itulah yang akan dirusak oleh kode.

Gunakan Firebase Test Lab untuk menjalankan aplikasi pada beragam macam perangkat sungguhan di pusat data Google. Dengan cara ini, Anda bisa memverifikasi fungsionalitas dan kompatibilitas aplikasi di berbagai jenis dan versi perangkat sebelum merilis aplikasi ke pengguna yang lebih luas.

Bacalah tentang Firebase Test Lab for Android di firebase.google.com/docs/test-lab/.

Pastikan aplikasi Anda memiliki filter yang tepat

Bila pengguna menelusuri atau menjelajahi aplikasi di Google Play Store, hasilnya hanya menyertakan aplikasi yang kompatibel dengan perangkat pengguna. Misalnya, jika seseorang menggunakan ponsel yang memiliki layar kecil, hasil penelusuran Google Play tidak akan menyertakan aplikasi yang memerlukan layar berukuran TV besar.

Pastikan aplikasi Anda menetapkan persyaratan yang sesuai untuk memastikan bahwa aplikasi menjangkau pengguna yang tepat. Misalnya, jika aplikasi Anda memerlukan perangkat keras biometrik untuk membaca sidik jari, maka tambahkan persyaratan tersebut di manifes Android.

```
<uses-feature android:name="android.hardware.fingerprint"/>
```

Akan tetapi, menetapkan bahwa aplikasi Anda memerlukan pembaca sidik jari akan membatasi pengguna aplikasi pada orang-orang yang memiliki perangkat dengan pembaca sidik jari. Anda harus berpikir dengan hati-hati sebelum menambahkan pembatasan ke manifes karena dapat membatasi orang yang bisa melihat dan mengunduh aplikasi tersebut.

Jika aplikasi Anda benar-benar mengharuskan adanya atribut tertentu di perangkat pengguna, maka pastikan menyertakan pembatasan tersebut ke manifes, untuk memastikan bahwa siapa saja yang bisa menemukan dan mengunduh aplikasi Anda akan benar-benar bisa menjalankannya. Orang sangat mungkin memberikan ulasan buruk pada aplikasi Anda jika setelah memasangnya dan ternyata aplikasi tersebut tidak bisa dijalankan di perangkat mereka.

Filter perangkat keras

Anda bisa menetapkan apakah aplikasi menggunakan fitur perangkat keras, seperti:

- sensor cahaya
  ```
  <uses-feature android:name="android.hardware.sensor.light" />
  ```
- gamepad
  ```
  <uses-feature android:name="android.hardware.gamepad" />
  ```
- penghitung langkah
  ```
  <uses-feature android:name="android.hardware.sensor.stepcounter" />
  ```
- dan banyak lagi

Lihat daftar lengkap di developer.android.com/guide/topics/manifest/uses-feature-element.html.

Filter perangkat lunak

Anda bisa menetapkan apakah aplikasi mengharuskan perangkat memiliki fitur perangkat lunak seperti:

- telah memasang pustaka bersama tertentu. Misalnya:
  ```
  <uses-library android:name="com.google.android.maps"/>
  ```
- menggunakan level Android API minimum. Misalnya:
  ```
  <uses-sdk android:minSdkVersion="19">
  ```

Negara

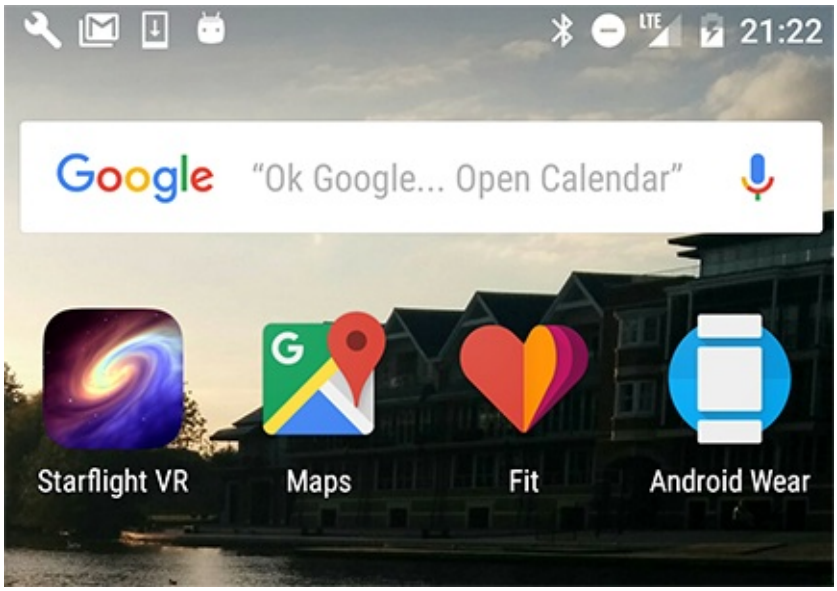
Dalam proses mengunggah aplikasi ke Google Play, Anda bisa memilih di negara mana saja aplikasi itu akan tersedia. Jika Anda menetapkannya, maka hanya pengguna di negara itu yang bisa menemukan dan mengunduh aplikasi.

Tambahkan ikon peluncur ke aplikasi Anda

Ikon peluncur adalah grafik yang menyatakan aplikasi Anda. Ikon peluncur untuk aplikasi muncul di listing Google Play Store. Bila pengguna menelusuri Google Play Store, ikon aplikasi Anda akan muncul di hasil penelusuran.

Bila pengguna telah memasang aplikasi, ikon peluncur akan muncul pada perangkat di beberapa tempat, termasuk:

- Di layar utama
- Di **Manage Applications**
- Di **My Downloads**



Baca panduan desain Ikon Peluncur untuk saran dalam mendesain aplikasi peluncur yang akan mendorong pengguna menggunakan aplikasi Anda.

Tambahkan ID Aplikasi

ID Aplikasi secara unik mengidentifikasi aplikasi. Pastikan aplikasi Anda memiliki ID Aplikasi yang akan selalu berbeda dari semua aplikasi lain yang mungkin telah dipasang pengguna di perangkat mereka.

Bila Anda membuat proyek untuk aplikasi Android, Android Studio secara otomatis memberikan ID Aplikasi ke proyek Anda. Nilai ini pada awalnya sama dengan paket untuk aplikasi. ID Aplikasi didefinisikan dalam file build.gradle. Misalnya:

```
defaultConfig {
   applicationId "com.example.android.materialme"
   minSdkVersion 15
   targetSdkVersion 24
   versionCode 1
   versionName "1.0"
}
```

Anda bisa mengubah ID Aplikasi untuk aplikasi. ID Aplikasi tidak harus sama dengan nama paket aplikasi Anda. Pada saat mengerjakan praktik di kursus ini, Anda membuat salinan proyek Android Studio. Setelah menyalin proyek, Anda mengubah ID Aplikasi untuk memastikannya bersifat unik bila Anda memasang aplikasi itu di perangkat.

Bila Anda siap mempublikasikan aplikasi, tinjaulah ID Aplikasi. ID Aplikasi mendefinisikan identitas aplikasi Anda. Jika Anda mengubahnya, aplikasi akan menjadi aplikasi yang berbeda dan pengguna aplikasi sebelumnya tidak bisa memperbarui ke aplikasi yang baru.

Menetapkan target API Level dan nomor versi

Bila Anda membuat proyek untuk aplikasi Android di Android Studio, pilih API level minimum dan target untuk aplikasi.

- **minSdkVersion** — versi minimum platform Android yang akan digunakan untuk menjalankan aplikasi.

- **targetSdkVersion** — API level yang digunakan untuk menjalankan aplikasi yang didesain.

Anda bisa menyetel nilai-nilai ini dalam file manifes Android, juga dalam file build.gradle tingkat aplikasi.

Catatan: Nilai di build.gradle akan **menggantikan** nilai di file manifes. Agar tidak membingungkan, kami menyarankan agar Anda memasukkan nilai-nilai di build.gradle, dan buang nilai tersebut dari file manifes. Menyetel atribut ini di build.gradle juga memungkinkan Anda menetapkan nilai yang berbeda untuk versi aplikasi yang berbeda.
Bila aplikasi Anda siap dirilis, tinjaulah nilai API level target dan nomor versi serta pastikan sudah benar. Orang tidak akan bisa menemukan aplikasi Anda di Google Play Store jika mereka menggunakan perangkat yang SdkVersion-nya di bawah nilai yang ditetapkan di aplikasi.

Inilah contoh penyetelan nilai atribut itu di build.gradle:

```
android {
  ...
  defaultConfig {
    ...
    minSdkVersion 14
    targetSdkVersion 24
}
```

Catatan: Nilai minSdkVersion dan targetSdkVersion merupakan API level, bukan nomor versi OS Android.

Nama kode		Versi	Tanggal Rilis	API Level
Honeycomb		3.0 - 3.2.6	Feb 2011	11 - 13
Ice Cream Sandwich		4.0 - 4.0.4	Okt 2011	14 - 15

Jelly Bean		4.1 - 4.3.1	Juli 2012	16 - 18
KitKat		4.4 - 4.4.4	Okt 2013	19 - 20
Lollipop		5.0 - 5.1.1	Nov 2014	21 - 22
Marshmallow		6.0 - 6.0.1	Okt 2015	23

Nougat		7.0	Sept 2016	24

Nomor versi

Anda perlu menetapkan nomor versi aplikasi. Saat meningkatkan aplikasi untuk menambahkan fitur baru, Anda perlu memperbarui nomor versi setiap kali merilis versi baru ke Google Play Store. Baca selengkapnya di panduan Versi Android.

Ragam Produk

Anda bisa menghasilkan "ragam produk" yang berbeda untuk aplikasi. Ragam produk adalah versi pembangunan aplikasi yang disesuaikan. Misalnya, Anda bisa memiliki versi demo dan versi produksi. Inilah contoh cara mendefinisikan ragam produk di build.gradle:

```
android {
  ...
  productFlavors {
    demo {
      applicationId "com.example.myapp.demo"
      versionName "1.0-demo"
    }
    full {
      applicationId "com.example.myapp.full"
      versionName "1.0-full"
    }
    }
}
```

Baca selengkapnya tentang ragam produk di panduan developer Konfigurasi Pembangunan.

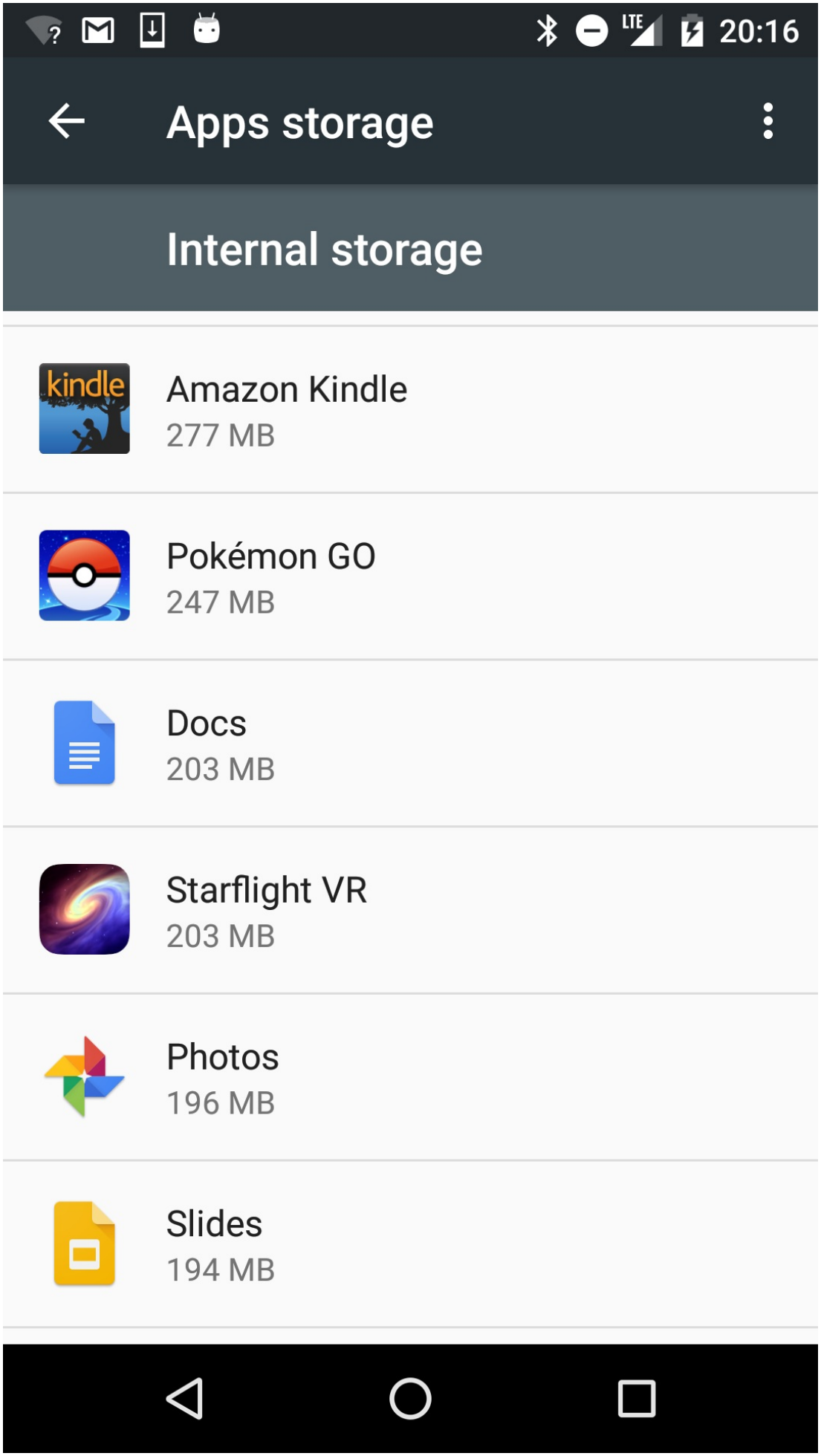
Kurangi ukuran aplikasi Anda

Ukuran APK Anda akan memengaruhi:

- seberapa cepat aplikasi Anda dimuat
- berapa banyak memori yang digunakannya
- berapa banyak daya yang dipakainya

Semakin besar ukuran APK aplikasi, semakin besar kemungkinan sebagian pengguna tidak akan mengunduhnya karena pembatasan ukuran pada perangkat mereka atau pembatasan konektivitas. Pengguna yang memiliki paket pembayaran berdasarkan kuota akan sangat mengkhawatirkan durasi pengunduhan aplikasi. Jika aplikasi Anda menggunakan ruang yang terlalu banyak, pengguna mungkin akan mencopot pemasangannya bila memerlukan ruang untuk aplikasi atau file lainnya.

Lihat aplikasi di ponsel Android Anda sekarang. Aplikasi manakah yang paling banyak memakan tempat? Jika kehabisan tempat di ponsel Android, unduhan aplikasi manakah yang akan Anda copot pemasangannya?


20:16
Apps storage
Internal storage
Amazon Kindle
277 MB
Pokémon GO
247 MB
Docs
203 MB
Starflight VR
203 MB
Photos
196 MB
Slides
194 MB

Tidak ada sulap untuk meminimalkan ukuran APK aplikasi, umumnya yang perlu Anda lakukan adalah berpikir logis. Misalnya:

- Bersihkan proyek untuk membuang sumber daya yang tidak digunakan
- Gunakan kembali sumber daya
- Minimalkan penggunaan sumber daya dari pustaka
- Kurangi kode asli dan Java
- Kurangi tempat yang dibutuhkan untuk gambar

Bersihkan folder proyek

Langkah pertama dalam membuat aplikasi Anda sekecil mungkin adalah membersihkan folder proyek sehingga bila membuat APK, Android Studio hanya menyertakan file yang diperlukan aplikasi.

File APK terdiri dari arsip ZIP yang berisi semua file yang dicakup oleh aplikasi Anda. File ini meliputi file kelas Java, file sumber daya, dan file berisi sumber daya yang dikompilasi.

Folder proyek tingkat tinggi di proyek Anda adalah:

- **src**: Folder ini berisi file sumber untuk aplikasi Anda. Buang file Java apa pun yang tidak digunakan. Pastikan folder tidak berisi file .jar apa pun.
- **lib**: Folder ini berisi file pustaka privat atau pihak ketiga, termasuk pustaka bersama yang telah dibuat sebelumnya dan pustaka statis (seperti file .so). Pastikan Anda membuang file pustaka yang tidak digunakan.
- **jni**: Folder ini berisi file sumber asli yang terkait dengan Android Native Developer Kit, seperti file .c, .cpp, .h, dan .mk.
- **res**: Folder sub res berisi sumber daya seperti layout, warna, string, dan gaya yang digunakan aplikasi Anda.

Catatan: Sewaktu mengembangkan aplikasi, boleh jadi mudah membuat sumber daya tambahan yang pada akhirnya tidak digunakan aplikasi, jadi pastikan untuk memeriksa dan membuang sumber daya yang tidak digunakan.

Nonaktifkan pembuatan log dan debug serta periksa URL produksi

Sewaktu mengembangkan aplikasi, Anda tidak perlu ragu untuk mengujinya dengan hati-hati, mungkin dengan menambahkan pengujian unit, menampilkan toast, dan menulis pernyataan log. Bila menyiapkan rilis aplikasi, Anda perlu membuang semua kode ekstra.

Untuk menonaktifkan pembuatan log dan debug:

- Buang panggilan dan pernyataan log untuk menampilkan toast
- Buang semua panggilan pelacakan Debug dari file kode sumber Anda seperti startMethodTracing() dan stopMethodTracing().
- Menonaktifkan debug di manifes Android dengan:
- Membuang atribut android:debuggable dari tag
- Atau menyetel atribut android:debuggable ke false
- Buang pengujian yang tidak digunakan

Selain itu, pastikan bahwa jika aplikasi mengakses server atau layanan jarak jauh, aplikasi akan menggunakan URL atau jalur produksi untuk server atau layanan dan bukan URL atau jalur pengujian. Demikian juga, banyak perusahaan dengan API publik yang memiliki izin pengujian dan izin tingkat produksi. Pastikan juga keamanan atau sandi untuk mengakses server berada pada tingkat produksi.

Kurangi ukuran gambar aplikasi Anda

Mengurangi ukuran gambar di aplikasi bisa sama hasilnya dengan mengurangi ukuran file APK.

- Pertama, pastikan aplikasi Anda tidak berisi sumber daya gambar yang tidak digunakan.
- Untuk setiap gambar statis yang digunakan aplikasi, Anda perlu membuat versi terpisah dari gambar itu untuk semua ukuran layar tempat aplikasi mungkin dijalankan. Akan tetapi, dalam beberapa kasus, Anda bisa menggunakan objek Drawable, VectorDrawables, dan file 9-patch.
- Jika aplikasi Anda menggunakan gambar statis, pastikan untuk mengecilkan file PNG, dan memadatkan file PNG dan JPEG untuk meminimalkan ukurannya. Anda bisa menggunakan Google untuk menelusuri alat (bantu) pengecilan, pemecahan, dan pemadatan gambar.
- Pertimbangkan penggunaan format WebP untuk gambar. Android mendukung WebP dari Android 4.0+.

Baca selengkapnya tentang cara mengurangi ukuran aplikasi di panduan Kurangi ukuran APK.

Catatan: Objek Drawable didefinisikan di XML. Android akan menggambarnya bila aplikasi perlu menampilkannya, yang berarti aplikasi Anda tidak perlu menyimpan gambar untuk objek tersebut. Akan tetapi, karena Android menghasilkannya bila diperlukan, maka perlu waktu lama untuk memunculkan gambar di layar, jadi sebaiknya gunakan objek Drawable untuk gambar yang lebih kecil seperti ikon dan logo. Objek Drawable tidak mendukung kompleksitas dan detail yang sama dengan yang bisa Anda dapatkan pada bitmap.
Di Android 5.0 (API level 21) dan di atasnya, Anda bisa mendefinisikan *sumber daya dapat digambar untuk vektor*, yakni gambar yang didefinisikan oleh suatu jalur. Sumber daya dapat digambar untuk vektor menskalakan tanpa kehilangan definisi. Sebagian besar sumber daya dapat digambar untuk vektor menggunakan file SVG, yaitu file teks biasa atau file biner kompresi yang menyertakan koordinat dua dimensi sebagai cara menggambar pada layar. Karena file SVG berupa teks, file tersebut menggunakan ruang lebih sedikit dibandingkan file gambar lainnya. Selain itu, Anda hanya memerlukan satu file untuk gambar vektor sebagai ganti satu file untuk setiap kepadatan layar.

Pertimbangkan kasus penggunaan gambar Anda, dan gunakan objek Drawable serta file 9-patch bila memungkinkan. Lihat pelajaran *Sumber Daya Dapat Digambar, Gaya, dan Tema* di kursus ini untuk informasi selengkapnya.

Buat APK bertandatangan untuk rilis

Bila aplikasi siap diunggah ke Google Play, Anda harus membuat dan menandatangani APK aplikasi. Android Studio memiliki alat untuk menghasilkan APK dan menandatanganinya dengan sertifikat digital.

Bila menandatangani aplikasi, Android Studio akan membuat sertifikat publik dan kunci privat. Android Studio melampirkan sertifikat publik ke APK. Anda harus menyimpan kunci privat dalam keystore secara aman

Sertifikat kunci publik berfungsi sebagai "sidik jari" yang secara unik mengaitkan APK dengan Anda dan kunci privat yang bersangkutan. Hal ini membantu Android memastikan bahwa pembaruan mendatang untuk APK adalah otentik dan berasal dari Anda, penulis aslinya.

Untuk informasi tentang sertifikat digital, menyimpan kunci privat, dan menghasilkan APK bertandatangan digital, lihat panduan Tanda Tangani Aplikasi Anda.

Publikasikan aplikasi Anda!

Bila telah menguji aplikasi, membersihkannya, mengurangi ukurannya, dan menghasilkan APK, Anda siap mempublikasikannya ke Google Play.

Setelah mengunggah aplikasi ke Google Play, Anda bisa menjalankan pengujian alfa dan beta sebelum merilisnya ke publik. Menjalankan pengujian alfa dan beta memungkinkan Anda berbagi aplikasi dengan pengguna sungguhan, dan mendapatkan masukan dari mereka. Masukan ini tidak muncul sebagai ulasan di Google Play.

Jalankan pengujian alfa sewaktu Anda mengembangkan aplikasi. Gunakan pengujian alfa untuk versi eksperimen awal dari aplikasi yang mungkin berisi fungsionalitas tidak lengkap atau tidak stabil. Menjalankan pengujian alfa juga merupakan cara yang baik untuk berbagi aplikasi Anda dengan teman dan keluarga.

Jalankan pengujian beta dengan pengguna sungguhan dalam jumlah terbatas, untuk melakukan pengujian akhir sebelum aplikasi dipublikasikan.

Setelah aplikasi dipublikasikan, pengguna bisa memberikan ulasan. Jadi, pastikan Anda menguji aplikasi secara menyeluruh sebelum dimasukkan ke Google Play untuk diunduh siapa saja.

Untuk informasi selengkapnya mengenai pengujian alfa dan beta, lihat:

- Panduan developer: developer.android.com/distribute/engage/beta.html
- Pusat bantuan: support.google.com/googleplay/android-developer/answer/3131213

Buat akun di Google Play Developer Console

Bila ingin menjalankan pengujian alfa dan beta, atau mempublikasikan aplikasi ke publik di Google Play, Anda perlu mengunggah APK di Google Play Developer Console.

Buka konsol di **play.google.com/apps/publish/.**

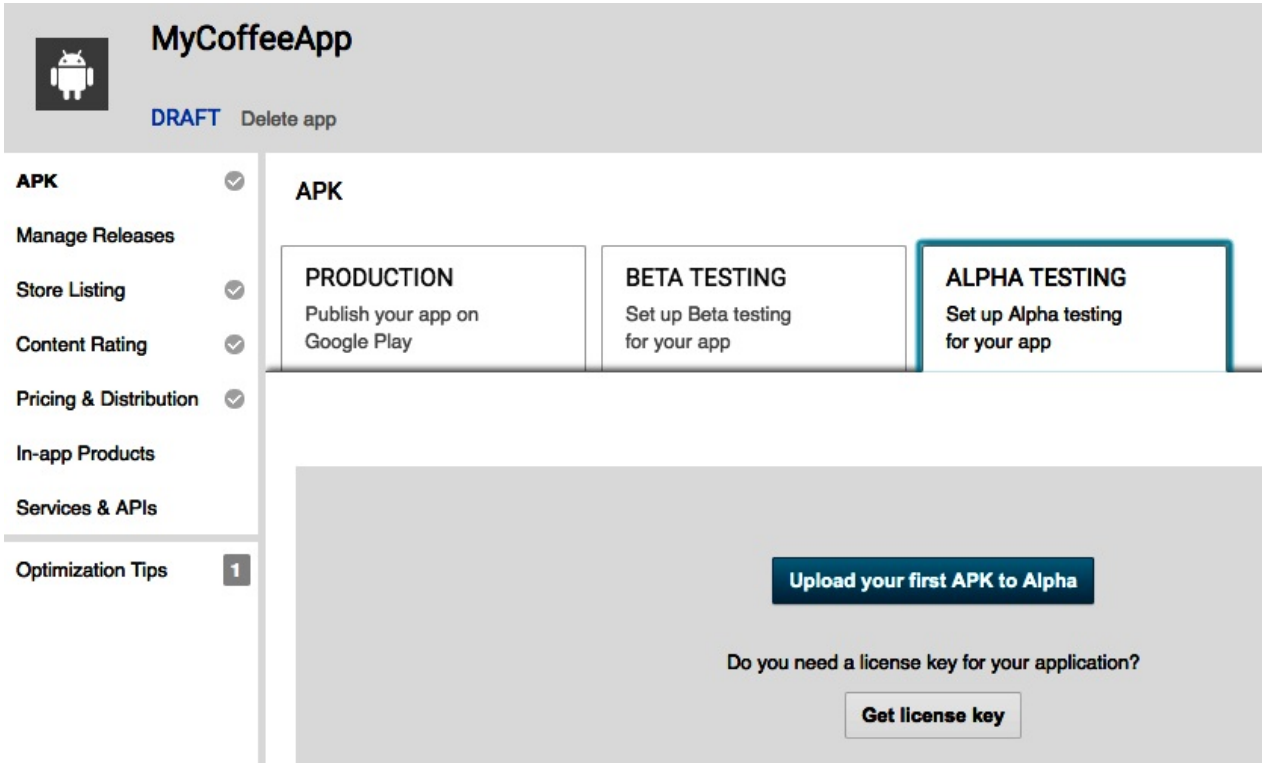
Untuk memulai, dapatkan akun developer Google Play. Anda perlu membayar untuk akun tersebut. Langkah-langkah tingkat tinggi adalah:

1. Buka play.google.com/apps/publish/
2. Terima perjanjian.
3. Bayar biaya pendaftaran.
4. Masukkan detail, seperti nama, alamat, situs web, telepon, dan preferensi email Anda.

Bila telah mempersiapkan akun, Anda bisa mengunggah APK. Dalam antarmuka Google Play Developer Console, pilih:

- Production
- Beta Testing
- Alpha Testing

Kemudian Anda bisa menjelajah ke APK yang akan diunggah, atau menyeret dan melepasnya ke konsol.



Anda perlu memenuhi persyaratan berikut sebelum bisa mempublikasikan aplikasi ke publik:

- menambahkan ikon beresolusi tinggi
- menambahkan grafik fitur (jika aplikasi dipilih sebagai Featured App di Google Play)
- menambahkan 2 tangkapan layar bukan Android TV

- memilih kategori
- memilih rating materi
- menargetkan setidaknya satu negara
- memasukkan URL kebijakan privasi
- menggratiskan aplikasi Anda atau menyetel harganya
- mendeklarasikan apakah aplikasi Anda berisi iklan
- menambahkan rating materi yang diperlukan

Daftar ini mungkin terlihat panjang, namun Google Play Developer Console membantu Anda mengetahui apakah aplikasi siap diluncurkan. Klik tautan "Why can't I publish?" untuk mengetahui apa lagi yang perlu dilakukan agar bisa mempublikasikan aplikasi Anda.

Jalankan laporan pra-peluncuran

Setelah APK diunggah, Anda bisa menjalankan laporan pra-peluncuran untuk mengidentifikasi masalah mogok, masalah tampilan, dan masalah keamanan. Selama pemeriksaan pra-peluncuran, perangkat pengujian secara otomatis diluncurkan dan merayapi aplikasi selama beberapa menit.

Perayapan akan melakukan tindakan dasar setiap beberapa detik pada aplikasi Anda, seperti mengetik, mengetuk, dan menggesek. Pengujian pra-peluncuran menggunakan Firebase Cloud Test Lab.

Untuk informasi selengkapnya tentang dukungan pra-peluncuran, lihat artikel Pusat Bantuan Google Play Gunakan laporan pra-peluncuran untuk mengidentifikasi masalah.

Tinjau kriteria untuk dipublikasikan

Aplikasi Anda harus memenuhi kebijakan Google Play, yang memastikan bahwa semua aplikasi pada Google Play menyediakan pengalaman yang aman untuk siapa saja.

Ada beberapa kebijakan yang mengatur

- Materi terlarang
- Hak kekayaan intelektual, penipuan, dan spam
- Privasi dan keamanan
- Monetisasi dan periklanan
- Listing toko dan promosi
- Keluarga

Ketahui selengkapnya di play.google.com/about/developer-content-policy/

Materi terlarang

Google Play tidak mengizinkan aplikasi yang memuat materi seksual eksplisit, kebencian, rasis, mengatai atau kekerasan, atau memfasilitasi perjudian.

https://play.google.com/about/restricted-content/

Hak kekayaan intelektual, penipuan, dan spam

Google Play tidak mengizinkan aplikasi yang tidak jujur. Dengan kata lain, aplikasi tersebut berpura-pura menjadi aplikasi lain atau berpura-pura berasal dari perusahaan lain atau meniru merek lain. Google Play tidak mengizinkan aplikasi yang berupaya menipu pengguna. Google Play tidak mengizinkan aplikasi yang mengirimkan spam kepada pengguna seperti aplikasi yang mengirimkan pesan yang tidak diminta kepada pengguna. Google Play tidak mengizinkan aplikasi yang merupakan duplikasi atau berkualitas rendah.

https://play.google.com/about/ip-deception-spam/

Privasi dan keamanan

Google Play mewajibkan aplikasi Anda memperlakukan data pengguna dengan aman dan menjaga kerahasiaan informasi pengguna. Jika aplikasi Anda mengakses atau mengirimkan data privat, aplikasi harus mempublikasikan pernyataan tentang cara aplikasi menggunakan data pengguna tersebut.

Google Play tidak mengizinkan aplikasi yang merusak atau secara subversif mengakses perangkat pengguna, aplikasi lain, server, jaringan, atau apa saja yang dilarang. Pada dasarnya, aplikasi Anda tidak boleh mengganggu apa pun, atau mengakibatkan kerusakan terhadap apa pun, atau mencoba mengakses apa pun yang tidak boleh diakses.

Google Play tidak mengizinkan aplikasi yang mencuri data, secara rahasia memantau atau membahayakan pengguna, atau merusak.

https://play.google.com/about/privacy-security/

Monetisasi dan Periklanan

Google Play memiliki peraturan yang menyangkut penerimaan pembayaran untuk pembelian dalam toko dan dalam aplikasi.

Google Play tidak mengizinkan aplikasi yang berisi iklan penipuan atau yang mengganggu.

https://play.google.com/about/monetization-ads/

Listing Toko dan Promosi

Penerbit tidak boleh berupaya mempromosikan aplikasinya secara tidak jujur. Misalnya, Anda tidak boleh meminta 100.000 teman dekat memberikan peringkat bintang 5 untuk aplikasi Anda, sehingga aplikasi muncul dengan ulasan yang sangat baik. Ikon aplikasi, judul, keterangan, dan tangkapan layar semuanya harus menyatakan aplikasi Anda secara jujur, dan tidak membuat klaim yang berlebihan atau menyesatkan.

Dengan kata lain, jangan curang dalam mendapatkan rating atau penempatan Google Play yang lebih baik.

https://play.google.com/about/storelisting-promotional/

Kirim aplikasi Anda untuk dipublikasikan

Bila mengunggah aplikasi untuk produksi, Google akan memeriksa aplikasi Anda. Google menjalankan pemeriksaan otomatis dan manual pada aplikasi Anda.

Jika aplikasi ditolak, perbaiki masalah dan coba lagi!

Rangkuman akhir

Anda sudah mencapai bagian akhir kursus ini. Kami berharap Anda menikmati perjalanan dan siap untuk membangun aplikasi Android sendiri. Kami menantikan aplikasi Anda di Google Play Store!

Ketahui selengkapnya

Mempersiapkan aplikasi Anda

- Mempersiapkan untuk rilis developer.android.com/studio/publish/preparing.html
- Daftar periksa peluncuran developer.android.com/distribute/tools/launch-checklist.html
- Daftar periksa kualitas aplikasi inti developer.android.com/distribute/essentials/quality/core.html
- Menangani data pengguna play.google.com/about/privacy-security/user-data/

- Filter aplikasi developer.android.com/google/play/filters.html
- API level min, maks, dan target developer.android.com/guide/topics/manifest/uses-sdk-element.html
- Ragam produk developer.android.com/studio/build/index.html
- Tetapkan versi aplikasi Anda developer.android.com/studio/publish/versioning.html
- Filter Google Play developer.android.com/google/play/filters.html
- Kurangi ukuran aplikasi developer.android.com/topic/performance/reduce-apk-size.htm
- Tandatangani aplikasi Anda developer.android.com/studio/publish/app-signing.html

Google Play Developer Console

- Masuklah ke konsol: play.google.com/apps/publish/
- Panduan developer: developer.android.com/distribute/googleplay/developer-console.html
- Pusat bantuan: support.google.com/googleplay/android-developer/#topic=3450769
- Mulai mempublikasikan developer.android.com/distribute/googleplay/start.html
- Firebase Test Lab: firebase.google.com/docs/test-lab/

Pengujian Alfa dan Beta

- Panduan developer: developer.android.com/distribute/engage/beta.html
- Pusat bantuan: support.google.com/googleplay/android-developer/answer/3131213