r

Nama : Daniel Maruli Sitohang
Nim : H1101171018
Program Studi : Sistem Informasi
Mata Kuliah : Teknologi Bergerak

Macam-Macam Komponen

1. Komponen aplikasi

Komponen aplikasi adalah blok pembangun penting dari aplikasi Android. Setiap komponen adalah titik masuk tempat sistem atau pengguna dapat memasuki aplikasi Anda. Beberapa komponen bergantung pada komponen lainnya.

Ada empat macam tipe komponen aplikasi:

- Aktivitas
- Layanan
- Penerima siaran
- Penyedia materi

Setiap tipe memiliki kegunaan tersendiri dan daur hidupnya sendiri yang mendefinisikan cara komponen dibuat dan dimusnahkan. Bagian berikut menguraikan empat tipe komponen aplikasi.

Aktivitas

Aktivitas adalah titik masuk untuk berinteraksi dengan pengguna. Ini mewakili satu layar dengan antarmuka pengguna. Misalnya, aplikasi email mungkin memiliki satu aktivitas yang menampilkan daftar email baru, aktivitas lain untuk menulis email, dan aktivitas satunya lagi untuk membaca email. Walaupun semua aktivitas bekerja sama untuk membentuk pengalaman pengguna yang kohesif dalam aplikasi email, masing-masing tidak saling bergantung. Karenanya, aplikasi berbeda bisa memulai salah satu aktivitas ini (jika aplikasi email mengizinkannya). Misalnya, aplikasi kamera bisa memulai aktivitas dalam aplikasi email yang membuat email baru agar pengguna bisa berbagi gambar. Aktivitas mempermudah interaksi penting berikut di antara sistem dan aplikasi:

- Tetap memantau apa yang penting bagi pengguna saat ini (apa yang ada di layar) untuk memastikan bahwa sistem tetap menjalankan proses yang menjadi host aktivitas.
- Memahami proses yang digunakan sebelumnya berisi sesuatu yang dapat dikembalikan pengguna (aktivitas yang dihentikan), jadi lebih memprioritaskan mempertahankan proses tersebut.
- Membantu menangani aplikasi menghentikan prosesnya sehingga pengguna dapat kembali ke aktivitas dengan status sebelumnya yang dipulihkan.

r

- Memberikan cara bagi aplikasi untuk menerapkan alur antar pengguna, dan bagi sistem untuk mengoordinasikan alur ini. (Contoh yang paling klasik sedang dibagikan di sini).

Anda menerapkan aktivitas sebagai subclass dari class Activity. Untuk informasi selengkapnya tentang class Activity, lihat panduan developer Aktivitas.

Contoh kode program :

```
<activity android:allowEmbedded=["true" | "false"]
          android:allowTaskReparenting=["true" | "false"]
          android:alwaysRetainTaskState=["true" | "false"]
          android:autoRemoveFromRecents=["true" | "false"]
          android:banner="drawable resource"
          android:clearTaskOnLaunch=["true" | "false"]
          android:colorMode=[ "hdr" | "wideColorGamut"]
          android:configChanges=["mcc", "mnc", "locale",
                                 "touchscreen", "keyboard",
"keyboardHidden",
                                 "navigation", "screenLayout",
"fontScale",
                                 "uiMode", "orientation",
"density",
                                 "screenSize",
"smallestScreenSize"]
          android:directBootAware=["true" | "false"]
          android:documentLaunchMode=["intoExisting" | "always"
|
                                  "none" | "never"]
          android:enabled=["true" | "false"]
          android:excludeFromRecents=["true" | "false"]
          android:exported=["true" | "false"]
          android:finishOnTaskLaunch=["true" | "false"]
          android:hardwareAccelerated=["true" | "false"]
          android:icon="drawable resource"
          android:immersive=["true" | "false"]
          android:label="string resource"
          android:launchMode=["standard" | "singleTop" |
                              "singleTask" | "singleInstance"]
          android:lockTaskMode=["normal" | "never" |
                              "if_whitelisted" | "always"]
          android:maxRecents="integer"
          android:maxAspectRatio="float"
          android:multiprocess=["true" | "false"]
          android:name="string"
```

e

r

```
          android:noHistory=["true" | "false"]
          android:parentActivityName="string"
          android:persistableMode=["persistRootOnly" |
                                   "persistAcrossReboots" |
"persistNever"]
          android:permission="string"
          android:process="string"
          android:relinquishTaskIdentity=["true" | "false"]
          android:resizeableActivity=["true" | "false"]
          android:screenOrientation=["unspecified" | "behind" |
                                     "landscape" | "portrait" |
                                     "reverseLandscape" |
"reversePortrait" |
                                     "sensorLandscape" |
"sensorPortrait" |
                                     "userLandscape" |
"userPortrait" |
                                     "sensor" | "fullSensor" |
"nosensor" |
                                     "user" | "fullUser" |
"locked"]
          android:showForAllUsers=["true" | "false"]
          android:stateNotNeeded=["true" | "false"]
          android:supportsPictureInPicture=["true" | "false"]
          android:taskAffinity="string"
          android:theme="resource or theme"
          android:uiOptions=["none" |
"splitActionBarWhenNarrow"]
          android:windowSoftInputMode=["stateUnspecified",
                                       "stateUnchanged",
"stateHidden",
                                       "stateAlwaysHidden",
"stateVisible",
                                       "stateAlwaysVisible",
"adjustUnspecified",
                                       "adjustResize",
"adjustPan"] >
    . . .
</activity>
```

e

r

Layanan

Layanan adalah titik masuk serbaguna untuk menjaga aplikasi tetap berjalan di latar belakang bagi semua jenis alasan. Ini adalah komponen yang berjalan di latar belakang untuk melakukan operasi yang berjalan lama atau untuk melakukan pekerjaan bagi proses jarak jauh. Layanan tidak menyediakan antarmuka pengguna. Misalnya, sebuah layanan bisa memutar musik di latar belakang sementara pengguna berada dalam aplikasi lain, atau layanan bisa menarik data lewat jaringan tanpa memblokir interaksi pengguna dengan aktivitas. Komponen lain, seperti aktivitas, bisa memulai layanan dan membiarkannya berjalan atau mengikat layanan untuk berinteraksi dengannya. Sebenarnya ada dua layanan semantik berbeda yang memberi tahu sistem tentang cara mengelola aplikasi: Layanan yang dimulai memberi tahu sistem agar tetap berjalan hingga pekerjaannya selesai. Hal ini bisa jadi untuk menyinkronkan beberapa data di latar belakang atau memutar musik meskipun setelah pengguna meninggalkan aplikasi tersebut. Menyinkronkan data di latar belakang atau memutar musik juga mewakili dua jenis layanan dimulai yang berbeda, yang memodifikasi bagaimana sistem menanganinya:

- Pemutaran musik adalah sesuatu yang disadari secara langsung oleh pengguna, jadi aplikasi tersebut memberi tahu sistem dengan mengatakan ingin berjalan di latar depan dengan notifikasi untuk memberi tahu pengguna tentang hal ini; dalam kasus ini sistem memahami bahwa harus benar-benar berusaha keras untuk menjaga proses layanan itu tetap berjalan, karena pengguna akan merasa tidak senang jika layanan itu hilang.
- Layanan latar belakang regular bukanlah sesuatu yang disadari pengguna secara langsung sebagai layanan yang berjalan, jadi sistem tersebut memiliki kebebasan lebih dalam mengelola prosesnya. Layanan ini mungkin diperbolehkan untuk dimatikan (lalu memulai ulang nanti) jika layanan ini memerlukan RAM untuk hal yang lebih penting bagi pengguna.

Layanan terikat berjalan karena beberapa aplikasi lain (atau sistem) dikatakan ingin menggunakan layanan tersebut. Pada dasarnya ini adalah layanan yang menyediakan API untuk proses lain. Dengan demikian sistem tersebut mengetahui adanya ketergantungan antar proses-proses ini, jadi jika proses A terikat ke layanan dalam proses B, proses A tahu bahwa harus mempertahankan proses B (beserta layanannya) agar tetap berjalan. Lebih lanjut, jika proses A adalah sesuatu yang dianggap penting bagi pengguna, maka proses A tersebut paham untuk memperlakukan proses B sebagai sesuatu yang juga dianggap penting bagi pengguna. Oleh karena fleksibilitasnya (baik atau buruk), layanan telah menjadi blok bangunan yang benar-benar berguna bagi semua jenis konsep sistem dengan level lebih tinggi. Wallpaper animasi, listener notifikasi, screen saver, metode masukan, layanan aksesibilitas, dan berbagai fitur sistem inti yang lain dibangun sebagai layanan yang menerapkan aplikasi dan mengikat sistem saat harus dijalankan.

Suatu layanan diterapkan sebagai subclass Service. Untuk informasi selengkapnya tentang class Service, lihat panduan developer Layanan.

e

r

Contoh kode program :

```
<service android:description="string resource"
         android:directBootAware=["true" | "false"]
         android:enabled=["true" | "false"]
         android:exported=["true" | "false"]
         android:foregroundServiceType=["connectedDevice" |
"dataSync" |
                                        "location" |
"mediaPlayback" | "mediaProjection" |
                                        "phoneCall"]
         android:icon="drawable resource"
         android:isolatedProcess=["true" | "false"]
         android:label="string resource"
         android:name="string"
         android:permission="string"
         android:process="string" >
    . . .
</service>
```

Penerima siaran

Penerima siaran adalah komponen yang memungkinkan sistem menyampaikan kejadian di luar alur pengguna regular, menjadikan aplikasi tersebut dapat merespons pengumuman siaran seluruh sistem. Oleh karena penerima siaran adalah entri yang didefinisikan dengan baik ke dalam aplikasi, sistem dapat mengirimkan siaran meskipun ke aplikasi yang saat ini tidak berjalan. Jadi, misalnya, suatu aplikasi dapat menjadwalkan alarm untuk mengirimkan notifikasi agar pengguna tahu tentang acara yang akan datang... dan dengan mengirimkan alarm tersebut ke Penerima Siaran aplikasi, aplikasi tersebut tidak perlu untuk tetap berjalan hingga alarm mati. Banyak siaran berasal dari sistem—misalnya, siaran yang mengumumkan bahwa layar sudah dimatikan, baterai lemah, atau gambar sudah diambil. Aplikasi juga dapat mengawali siaran—misalnya, untuk memberi tahu aplikasi lain bahwa beberapa data sudah didownload ke perangkat dan tersedia untuk digunakan. Walaupun penerima siaran tidak menampilkan antarmuka pengguna, penerima bisa membuat notifikasi bilah status untuk memberi tahu pengguna kapan kejadian siaran dilakukan. Meskipun penerima siaran umumnya cuma menjadi *gerbang* untuk komponen lain dan dimaksudkan untuk melakukan pekerjaan dalam jumlah sangat minim. Misalnya, mungkin dijadwalkan JobService melakukan beberapa pekerjaan berdasarkan acara dengan JobScheduler

Penerima siaran diimplementasikan sebagai subclass BroadcastReceiver dan setiap siaran dikirim sebagai objek Intent. Untuk informasi selengkapnya, lihat class BroadcastReceiver.

e

r

Contoh kode program :

```
<receiver android:directBootAware=["true" | "false"]
          android:enabled=["true" | "false"]
          android:exported=["true" | "false"]
          android:icon="drawable resource"
          android:label="string resource"
          android:name="string"
          android:permission="string"
          android:process="string" >
    . . .
</receiver>
```

Penyedia materi

Penyedia materi mengelola set data aplikasi secara bersama-sama, yang dapat Anda simpan di sistem file, di database SQLite, di web, atau di lokasi penyimpanan persisten lain yang dapat diakses aplikasi Anda. Melalui penyedia materi, aplikasi lain bisa melakukan kueri atau memodifikasi data jika penyedia materi mengizinkannya. Misalnya, sistem Android menyediakan penyedia materi yang mengelola informasi kontak pengguna. Karenanya, setiap aplikasi dengan izin yang sesuai bisa melakukan kueri mengenai bagian dari penyedia materi, seperti ContactsContract.Data, untuk membaca dan menulis informasi tentang orang tertentu. Aplikasi ini membujuk agar memikirkan penyedia konten sebagai abstraksi di database, karena terdapat banyak API dan dukungan dibuat untuk kasus umum tersebut. Namun demikian, penyedia konten memiliki beragam tujuan inti untuk perspektif desain-sistem. Bagi sistem, penyedia konten adalah titik masuk ke dalam suatu aplikasi untuk memublikasikan item data bernama, yang diidentifikasi oleh skema URI. Jadi sebuah aplikasi dapat memutuskan bagaimana ia ingin memetakan data yang ada di dalamnya ke ruang nama URI, membagikan URI tersebut ke entitas lain yang dapat menggunakannya guna mengakses data. Ada beberapa hal tertentu yang dapat dilakukan sistem dalam mengelola sebuah aplikasi:

- Menetapkan URI tidak mengharuskan aplikasi tetap berjalan, sehingga URI dapat terus ada setelah aplikasi yang memilikinya keluar. Sistem hanya perlu memastikan bahwa aplikasi yang memilikinya masih berjalan saat harus mengambil data aplikasi tersebut dari URI yang terkait.
- URI ini juga menyediakan model keamanan halus yang penting. Misalnya, sebuah aplikasi dapat menempatkan URO untuk gambar yang ada di papan klip, namun membiarkan penyedia kontennya terkunci sehingga aplikasi lain tidak dapat mengaksesnya secara bebas. Apabila aplikasi kedua berupaya mengakses URI tersebut di papan klip, sistem dapat mengizinkan aplikasi tersebut untuk mengakses data melalui *pemberian izin URI* sementara sehingga diizinkan mengakses data hanya di belakang URI tersebut, namun tidak ada data lainnya di aplikasi kedua itu.

e

r

Penyedia materi juga berguna untuk membaca dan menulis data privat ke aplikasi Anda dan tidak dibagikan.

Penyedia materi diimplementasikan sebagai subclass ContentProvider dan harus mengimplementasikan seperangkat standar API yang memungkinkan aplikasi lain melakukan transaksi. Untuk informasi selengkapnya, lihat panduan developer Penyedia Materi.

Aspek unik dari desain sistem Android adalah aplikasi mana pun bisa memulai komponen aplikasi lain. Misalnya, jika Anda menginginkan pengguna mengambil foto dengan kamera perangkat, bisa saja aplikasi lain yang melakukannya dan aplikasi Anda bisa menggunakannya, sebagai ganti mengembangkan aktivitas sendiri untuk mengambil foto. Anda tidak perlu menggabungkan atau bahkan menghubungkan ke kode dari aplikasi kamera. Sebagai gantinya, Anda dapat memulai aktivitas di aplikasi kamera yang berupa aktivitas mengambil sebuah foto. Bila selesai, foto akan dikembalikan ke aplikasi sehingga Anda bisa menggunakannya. Bagi pengguna, kamera seakan menjadi bagian dari aplikasi Anda.

Saat sistem memulai komponen, sistem akan memulai proses untuk aplikasi itu (jika belum berjalan) dan membuat instance class yang diperlukan untuk komponen. Misalnya, jika aplikasi Anda memulai aktivitas dalam aplikasi kamera yang berupa aktivitas mengambil foto, aktivitas itu akan berjalan dalam proses yang dimiliki oleh aplikasi kamera, bukan dalam proses aplikasi Anda. Karenanya, tidak seperti aplikasi di sebagian besar sistem lain, aplikasi Android tidak memiliki titik masuk tunggal (tidak ada main() fungsi).

Karena sistem menjalankan setiap aplikasi dalam proses terpisah dengan izin file yang membatasi akses ke aplikasi lain, aplikasi Anda tidak bisa langsung mengaktifkan komponen dari aplikasi lain. Namun demikian, sistem Android dapat melakukan hal tersebut. Untuk mengaktifkan komponen dalam aplikasi lain, kirim pesan ke sistem yang menetapkan *intent* Anda untuk memulai komponen tertentu. Selanjutnya sistem akan mengaktifkan komponen untuk Anda.

Contoh kode program :

```
<provider android:authorities="list"
          android:directBootAware=["true" | "false"]
          android:enabled=["true" | "false"]
          android:exported=["true" | "false"]
          android:grantUriPermissions=["true" | "false"]
          android:icon="drawable resource"
          android:initOrder="integer"
          android:label="string resource"
          android:multiprocess=["true" | "false"]
          android:name="string"
          android:permission="string"
          android:process="string"
          android:readPermission="string"
          android:syncable=["true" | "false"]
          android:writePermission="string" >
```

e

r

```
    . . .
</provider>
```

Mengaktifkan komponen

Tiga dari empat tipe komponen—aktivitas, layanan, dan penerima siaran—diaktifkan oleh pesan asinkron yang disebut *intent*. Intents mengikat antar masing-masing komponen di runtime. Anda dapat menganggap intent sebagai messenger yang meminta tindakan dari komponen lain, entah komponen milik aplikasi Anda atau komponen lainnya.

Intent dibuat dengan objek Intent, yang mendefinisikan pesan untuk mengaktifkan komponen tertentu (intent ekspolisit) atau *tipe* komponen spesifik (intent implisit).

Untuk aktivitas dan layanan, intent mendefinisikan aksi yang akan dilakukan (misalnya, untuk *melihat* atau *mengirim* sesuatu) dan mungkin menetapkan URI data untuk ditindaklanjuti, salah satu hal yang mungkin perlu diketahui oleh komponen yang akan dimulai. Misalnya, intent mungkin menyampaikan permintaan suatu aktivitas untuk menampilkan gambar atau membuka halaman web. Dalam beberapa kasus, Anda dapat mengawali aktivitas untuk menerima hasil, dalam kasus ini aktivitas juga mengembalikan hasil dalam Intent. Misalnya, Anda bisa mengeluarkan intent agar pengguna dapat memilih kontak pribadi dan mengembalikannya kepada Anda. Inten yang dikembalikan mencakup URI yang menunjuk ke kontak terpilih.

Untuk penerima siaran, intent hanya mendefinisikan pengumuman yang sedang disiarkan. Misalnya, siaran untuk menunjukkan baterai perangkat hampir habis hanya menyertakan string tindakan yang menunjukkan *baterai hampir habis*.

Tidak seperti aktivitas, layanan, dan penerima siaran, penyedia konten tidak diaktifkan oleh intent. Melainkan, diaktifkan saat ditargetkan oleh permintaan dari ContentResolver. Resolver konten menangani semua transaksi langsung dengan penyedia konten sehingga komponen yang melakukan transaksi dengan penyedia tidak perlu dan sebagai gantinya memanggil metode pada objek ContentResolver. Ini membuat layer abstraksi antara penyedia konten dan komponen yang meminta informasi (demi keamanan).

Ada beberapa metode terpisah untuk mengaktifkan masing-masing tipe komponen:

- Anda bisa memulai aktivitas (atau memberinya pekerjaan baru) dengan meneruskan Intent ke startActivity() atau startActivityForResult() (bila Anda ingin aktivitas mengembalikan hasil).
- Dengan Android 5.0 (API level 21) dan sesudahnya, Anda dapat menggunakan class JobScheduler untuk menjadwalkan tindakan. Untuk versi Android sebelumnya, Anda bisa memulai layanan (atau memberikan petunjuk baru ke layanan yang sedang berlangsung) dengan meneruskan Intent ke startService(). Anda bisa mengikat ke layanan dengan meneruskan Intent ke bindService().
- Anda bisa memulai siaran dengan meneruskan Intent ke metode-metode seperti sendBroadcast(), sendOrderedBroadcast(), atau sendStickyBroadcast().
- Anda bisa melakukan kueri ke penyedia konten dengan memanggil query() pada ContentResolver.

Untuk informasi selengkapnya tentang menggunakan intent, lihat dokumen Intent dan Filter Intent. Dokumen berikut memberikan informasi lebih lanjut tentang mengaktifkan komponen tertentu: Aktivitas, Layanan, BroadcastReceiver, dan Penyedia Konten.

e

r

The first activity, `com.android.notepad.NotesList`, serves as our main entry into the app. It can do three things as described by its three intent templates:

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

- This provides a top-level entry into the NotePad application: the standard MAIN action is a main entry point (not requiring any other information in the Intent), and the LAUNCHER category says that this entry point should be listed in the application launcher.

- ```
  <intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <action android:name="android.intent.action.EDIT" />
    <action android:name="android.intent.action.PICK" />
    <category android:name="android.intent.category.DEFAULT" />
    <data
  android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
  </intent-filter>
  ```

- This declares the things that the activity can do on a directory of notes. The type being supported is given with the <type> tag, where `vnd.android.cursor.dir/vnd.google.note` is a URI from which a Cursor of zero or more items (`vnd.android.cursor.dir`) can be retrieved which holds our note pad data (`vnd.google.note`). The activity allows the user to view or edit the directory of data (via the VIEW and EDIT actions), or to pick a particular note and return it to the caller (via the PICK action). Note also the DEFAULT category supplied here: this is *required* for the `Context#startActivity` method to resolve your activity when its component name is not explicitly specified.

- ```
  <intent-filter>
    <action android:name="android.intent.action.GET_CONTENT" />
    <category android:name="android.intent.category.DEFAULT" />
    <data
  android:mimeType="vnd.android.cursor.item/vnd.google.note" />
  </intent-filter>
  ```

This filter describes the ability to return to the caller a note selected by the user without needing to know where it came from. The data type `vnd.android.cursor.item/vnd.google.note` is a URI from which a Cursor of exactly one (`vnd.android.cursor.item`) item can be retrieved which contains our note pad data (`vnd.google.note`). The GET_CONTENT action is similar to the PICK action, where the activity will return to its caller a piece of data selected by the user. Here, however, the caller specifies the type of data they desire instead of the type of data the user will be picking from.

e

r

2. Komponen File Manifest

Sebelum sistem Android bisa memulai komponen aplikasi, sistem harus mengetahui bahwa komponen ada dengan membaca *file manifes*, `AndroidManifest.xml` aplikasi. Aplikasi Anda harus mendeklarasikan semua komponennya dalam file ini, yang harus menjadi root dari direktori proyek aplikasi.

Manifes melakukan banyak hal selain mendeklarasikan komponen aplikasi, seperti yang berikut:

- Mengidentifikasi izin pengguna yang diperlukan aplikasi, seperti akses Internet atau akses-baca ke kontak pengguna.
- Mendeklarasikan API Level minimum yang diperlukan aplikasi, berdasarkan API yang digunakan aplikasi.
- Mendeklarasikan fitur perangkat keras dan perangkat lunak yang diperlukan aplikasi, seperti kamera, layanan Bluetooth, atau layar multisentuh.
- Mendeklarasikan pustaka API aplikasi perlu ditautkan (selain Android framework API), seperti library Google Maps.

Mendeklarasikan komponen

Tugas utama manifes adalah menginformasikan komponen aplikasi pada sistem. Misalnya, file manifes bisa mendeklarasikan aktivitas sebagai berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity
android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

Dalam elemen `<application>`, atribut `android:icon` menunjuk ke sumber daya untuk ikon yang mengidentifikasi aplikasi.

Dalam elemen `<activity>`, atribut `android:name` menetapkan nama class yang sepenuhnya memenuhi syarat subclass `Activity` dan atribut `android:label` menetapkan string yang akan digunakan sebagai label yang terlihat oleh pengguna untuk aktivitas tersebut.

e

r

Anda harus mendeklarasikan semua komponen aplikasi menggunakan elemen berikut:

- Elemen `<activity>` untuk aktivitas.
- Elemen `<service>` untuk layanan.
- Elemen `<receiver>` untuk penerima siaran.
- Elemen `<provider>` untuk penyedia konten.

Aktivitas, layanan, dan penyedia konten yang Anda sertakan dalam kode sumber, namun tidak dideklarasikan dalam manifes, tidak akan terlihat pada sistem dan, akibatnya, tidak pernah bisa berjalan. Akan tetapi, penerima siaran bisa dideklarasikan dalam manifes atau dibuat secara dinamis dalam kode sebagai objek `BroadcastReceiver` dan didaftarkan pada sistem dengan memanggil `registerReceiver()`.

Untuk informasi selengkapnya tentang cara menstrukturkan file manifes untuk aplikasi Anda, lihat dokumentasi File AndroidManifest.xml.

Mendeklarasikan kemampuan komponen

Seperti telah dibahas di atas, dalam Mengaktifkan omponen, Anda bisa menggunakan `Intent` untuk memulai aktivitas, layanan, dan penerima siaran. Anda bisa menggunakan `Intent` dengan menamai komponen sasaran secara eksplisit (menggunakan nama class komponen) dalam intent. Anda juga dapat menggunakan intent implisit, yang menjelaskan tipe tindakan yang dilakukan dan, secara opsional, data tempat Anda ingin melakukan tindakan. Dengan intent implisit, sistem dapat menemukan komponen di perangkat yang dapat menjalankan tindakan dan memulainya. Jika ada banyak komponen yang bisa melakukan tindakan yang dijelaskan oleh intent, maka pengguna bisa memilih komponen yang akan digunakan.

Sistem mengidentifikasi komponen yang bisa merespons maksud adalah dengan membandingkan intent yang diterima dengan *filter intent* yang disediakan dalam file manifes aplikasi lainnya pada perangkat.

Bila mendeklarasikan aktivitas dalam manifes aplikasi, secara opsional Anda bisa menyertakan filter intent yang mendeklarasikan kemampuan aktivitas agar bisa merespons intent dari aplikasi lain. Anda bisa mendeklarasikan filter intent untuk komponen dengan menambahkan elemen `<intent-filter>` sebagai anak elemen deklarasi komponen.

Misalnya, jika Anda membuat aplikasi email dengan aktivitas untuk menulis email baru, Anda bisa mendeklarasikan filter maksud untuk merespons intent "kirim" (untuk mengirim email baru) seperti yang ditunjukkan di contoh berikut:

```
<manifest ... >
    ...
    <application ... >
        <activity
android:name="com.example.project.ComposeEmailActivity">
            <intent-filter>
                <action
```

e

r

```
android:name="android.intent.action.SEND" />
                <data android:type="*/*" />
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Kemudian, jika aplikasi lain membuat intent dengan aksi ACTION_SEND dan meneruskannya ke startActivity(), sistem mungkin akan memulai aktivitas Anda agar pengguna bisa menulis draf dan mengirim email.

Untuk informasi selengkapnya tentang membuat filter intent, lihat dokumen Intent dan Filter Intent.

Mendeklarasikan persyaratan aplikasi

Ada berbagai macam perangkat yang didukung oleh Android dan tidak semuanya menyediakan fitur dan kemampuan yang sama. Untuk mencegah aplikasi Anda diinstal pada perangkat yang tidak memiliki fitur yang diperlukan aplikasi, Anda harus jelas mendefinisikan profil mengenai tipe perangkat yang didukung aplikasi dengan mendeklarasikan kebutuhan perangkat dan perangkat lunak dalam file manifes. Kebanyakan deklarasi ini hanya bersifat informasi dan sistem tidak membacanya, namun layanan eksternal seperti Google Play akan membacanya untuk menyediakan penyaringan bagi pengguna saat mereka menelusuri aplikasi dari perangkat.

Misalnya, jika aplikasi memerlukan kamera dan menggunakan API yang disediakan dalam Android 2.1 (API Level 7), Anda harus mendeklarasikannya sebagai kebutuhan dalam file manifes seperti ditunjukkan dalam contoh berikut:

```
<manifest ... >
    <uses-feature android:name="android.hardware.camera.any"
                  android:required="true" />
    <uses-sdk android:minSdkVersion="7"
android:targetSdkVersion="19" />
    ...
</manifest>
```

Dengan deklarasi yang ditunjukkan dalam contoh tersebut, perangkat yang *tidak* memiliki kamera dan menggunakan Android versi *lebih rendah* dari 2.1 tidak bisa menginstal aplikasi Anda dari Google Play. Akan tetapi, Anda dapat mendeklarasikan bahwa aplikasi Anda menggunakan kamera, namun tidak *mengharuskannya*. Dalam hal itu, aplikasi Anda harus menyetel atribut `required` ke `false` dan memeriksa pada waktu proses apakah perangkat memiliki kamera dan menonaktifkan setiap fitur kamera yang sesuai.

r

Informasi selengkapnya tentang cara mengelola kompatibilitas aplikasi dengan perangkat yang berbeda disediakan dalam dokumen Kompatibilitas Perangkat.

3. Komponen Sumber Daya Aplikasi

Aplikasi Android tidak hanya terdiri dari kode—aplikasi memerlukan sumber daya yang terpisah dari kode sumber, seperti gambar, file audio, dan apa saja yang berkaitan dengan presentasi visual suatu aplikasi. Misalnya, Anda dapat mendefinisikan animasi, menu, gaya, warna, dan layout antarmuka pengguna aktivitas dengan file XML. Dengan menggunakan sumber daya aplikasi mempermudah dalam mengupdate berbagai karakteristik aplikasi Anda tanpa memodifikasi kode. Menyediakan seperangkat sumber daya alternatif memungkinkan Anda mengoptimalkan aplikasi untuk berbagai konfigurasi perangkat berbeda (seperti bahasa dan ukuran layar yang berbeda).

Untuk setiap sumber daya yang Anda sertakan dalam proyek Android, alat bawaan SDK akan mendefinisikan ID integer unik, yang bisa Anda gunakan untuk mengacu sumber daya dari kode aplikasi atau dari sumber daya lainnya yang didefinisikan dalam XML. Misalnya, jika aplikasi berisi file gambar bernama `logo.png` (disimpan dalam direktori `res/drawable/`), alat SDK akan menghasilkan ID sumber daya bernama `R.drawable.logo`. Peta ID ini untuk integer khusus aplikasi, yang dapat Anda gunakan untuk mereferensikan gambar dan memasukkannya dalam antarmuka pengguna.

Salah satu aspek paling penting dari penyediaan sumber daya yang terpisah dari kode sumber adalah kemampuan untuk menyediakan sumber daya alternatif untuk konfigurasi perangkat yang berbeda. Misalnya, dengan mendefinisikan string UI dalam XML, Anda bisa menerjemahkan string ke dalam bahasa lain dan menyimpan string itu dalam file terpisah. Lalu Android menerapkan string bahasa yang sesuai ke UI Anda berdasarkan *qualifier* bahasa yang Anda tambahkan ke nama direktori sumber daya (seperti `res/values-fr/` untuk nilai string bahasa Prancis) dan setelan bahasa pengguna.

Android mendukung banyak *qualifier* berbeda untuk sumber daya alternatif Anda. Qualifier adalah string pendek yang Anda sertakan dalam nama direktori sumber daya untuk mendefinisikan konfigurasi perangkat yang harus digunakan sumber daya tersebut. Misalnya, Anda harus membuat layout berbeda untuk aktivitas, bergantung pada orientasi layar dan ukuran perangkat. Apabila layar perangkat dalam orientasi potret, Anda mungkin ingin layout tombolnya vertikal (tinggi), tetapi saat layar dalam orientasi lanskap (lebar), tombolnya dapat disejajarkan secara horizontal. Untuk mengubah layout sesuai orientasi, Anda bisa mendefinisikan dua layout berbeda dan menerapkan qualifier yang tepat untuk setiap nama direktori layout. Kemudian, sistem secara otomatis menerapkan layout yang tepat sesuai dengan orientasi perangkat saat ini.

Untuk informasi selengkapnya tentang berbagai jenis sumber daya yang bisa disertakan dalam aplikasi dan cara membuat sumber daya alternatif untuk konfigurasi perangkat berbeda, bacalah Menyediakan Sumber Daya. Untuk mengetahui tentang praktik terbaik dan mendesain aplikasi berkualitas produksi yang kuat, lihat Panduan untuk Arsitektur Aplikasi.

e

r

Direktori	Tipe Sumber Daya
`animator/`	File XML yang menetapkan animasi properti.
`anim/`	File XML yang menetapkan animasi tween. (Animasi properti juga bisa disimpan dalam direktori ini, namun direktori `animator/` lebih disukai bagi animasi properti agar kedua tipe ini dapat dibedakan.)
`color/`	File XML yang mendefinisikan daftar status warna. Lihat Sumber Daya Daftar Status Warna
`drawable/`	File bitmap (`.png`, `.9.png`, `.jpg`, `.gif`) atau file XML yang dikompilasi menjadi subtipe sumber daya yang dapat digambar: • File bitmap • Nine-Patches (bitmap yang dapat diubah ukurannya) • Daftar status • Bentuk • Sumber daya dapat digambar untuk animasi • Sumber daya dapat digambar lainnya Lihat Sumber Daya Drawable.
`mipmap/`	File sumber daya dapat digambar untuk beragam kepadatan ikon peluncur. Untuk informasi selengkapnya tentang mengelola ikon peluncur dengan folder `mipmap/`, lihat Mengelola Ringkasan Project.
`layout/`	File XML yang mendefinisikan layout antarmuka pengguna. Lihat Sumber Daya Layout.
`menu/`	File XML yang menetapkan menu aplikasi, seperti Menu Opsi, Menu Konteks, atau Sub Menu. Lihat Sumber Daya Menu.
`raw/`	File arbitrer yang akan disimpan dalam bentuk mentah. Untuk membuka sumber daya ini dengan `InputStream` mentah, panggil `Resources.openRawResource()` dengan ID sumber daya, yaitu `R.raw.`*`filename`*. Akan tetapi, jika butuh akses ke nama file asal dan hierarki file, Anda bisa mempertimbangkan untuk menyimpan beberapa sumber daya dalam direktori `assets/` (sebagai ganti `res/raw/`). File dalam `assets/` tidak diberi ID sumber daya, jadi Anda bisa membacanya hanya dengan menggunakan `AssetManager`.
`values/`	File XML yang berisi nilai-nilai sederhana, seperti string, integer, dan warna. Karena file sumber daya XML dalam subdirektori `res/` lainnya menetapkan satu sumber daya berdasarkan nama file XML, file dalam direktori `values/` menggambarkan beberapa sumber daya. Untuk file dalam direktori ini, setiap turunan elemen `<resources>` menetapkan satu sumber daya. Misalnya, elemen `<string>` membuat sumber daya `R.string` dan elemen `<color>` membuat sumber daya `R.color`. Karena setiap sumber daya ditetapkan dengan elemen XML-nya sendiri, Anda bisa bebas menamai file ini dan menempatkan tipe sumber daya berbeda dalam satu file.

e

r

Akan tetapi, agar jelas, Anda mungkin perlu menempatkan tipe sumber daya unik dalam file berbeda. Misalnya, berikut ini adalah beberapa ketentuan penamaan file untuk sumber daya yang dapat Anda buat dalam direktori ini:

- arrays.xml untuk array sumber daya tipe (array bertipe).
- colors.xml untuk nilai warna
- dimens.xml untuk nilai dimensi.
- strings.xml untuk nilai string.
- styles.xml untuk gaya.

Lihat Sumber Daya String, Sumber Daya Gaya, dan Tipe Sumber Daya Lainnya.

`xml/`	File XML arbitrer yang bisa dibaca saat waktu proses dengan memanggil `Resources.getXML()`. Berbagai file konfigurasi XML harus disimpan di sini, seperti konfigurasi yang dapat dicari.
`font/`	File font dengan ekstensi seperti `.ttf`, `.otf`, atau `.ttc`, atau file XML yang menyertakan elemen `<font-family>`. Untuk informasi selengkapnya tentang font sebagai sumber daya, buka Font dalam XML.

Dapat digambar

Untuk membuat alias ke sumber daya dapat digambar yang ada, gunakan elemen `<drawable>`. Sebagai contoh program:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <drawable name="icon">@drawable/icon_ca</drawable>
</resources>
```

Jika Anda menyimpan file ini sebagai `drawables.xml` (dalam direktori sumber daya alternatif, seperti `res/values-en-rCA/`), maka file akan dikompilasi menjadi sumber daya yang bisa Anda referensikan sebagai `R.drawable.icon`, namun sebenarnya merupakan alias untuk sumber daya `R.drawable.icon_ca` (yang disimpan dalam `res/drawable/`).

Layout

Untuk membuat alias ke layout yang ada, gunakan elemen , `<include>` yang ada dalam `<merge>`. Sebagai contoh program :

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

e

r

Jika Anda menyimpan file ini sebagai `main.xml`, file akan dikompilasi menjadi sumber daya yang bisa Anda referensikan sebagai `R.layout.main`, namun sebenarnya merupakan alias untuk sumber daya `R.layout.main_ltr`.

String dan nilai-nilai sederhana lainnya

Untuk membuat alias ke string yang ada, cukup gunakan ID sumber daya dari string yang diinginkan sebagai nilai untuk string baru. Sebagai contoh program :

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

Sumber daya `R.string.hi` sekarang merupakan alias untuk `R.string.hello`.

Nilai sederhana lainnya cara kerjanya sama. Misalnya, sebuah warna:

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

Mengakses sumber daya aplikasi Anda

Setelah menyediakan sumber daya dalam aplikasi Anda, Anda bisa menerapkannya dengan mengacu pada ID sumber dayanya. Semua ID sumber daya didefinisikan di class `R` proyek Anda, yang dihasilkan oleh fitur `aapt` secara otomatis.

Bila aplikasi Anda telah dikompilasi, `aapt` akan menghasilkan class `R`, yang berisi ID sumber daya untuk semua sumber daya dalam direktori `res/` Anda. Untuk masing-masing tipe sumber daya, ada subclass `R` (misalnya, `R.drawable` untuk semua sumber daya yang dapat digambar), dan untuk masing-masing sumber daya bertipe itu, ada satu integer statis (misalnya, `R.drawable.icon`). Integer ini adalah ID sumber daya yang bisa Anda gunakan untuk mengambil sumber daya.

Walaupun class `R` adalah tempat menetapkan ID sumber daya, Anda tidak perlu melihat ke sana untuk menemukan ID sumber daya. ID sumber daya selalu terdiri dari:

- *Tipe sumber daya*: Masing-masing sumber daya dikelompokkan menjadi "tipe", misalnya `string`, `drawable`, dan `layout`. Untuk mengetahui selengkapnya tentang berbagai tipe, lihat Tipe Sumber Daya.
- *Nama sumber daya*, bisa berupa: nama file, tidak termasuk ekstensi; atau nilai dalam atribut `android:name` XML, jika sumber daya itu sebuah nilai sederhana (misalnya sebuah string).

e

r

Ada dua cara untuk mengakses sumber daya:

- **Dalam kode:** Menggunakan integer statis dari subclass dari class `R`, misalnya:

  ```
  R.string.hello
  ```

• `string` adalah tipe sumber daya dan `hello` adalah nama sumber daya. Ada banyak Android API yang bisa mengakses sumber daya Anda bila Anda menyediakan ID sumber daya dengan format ini. Lihat Mengakses Sumber Daya dalam Kode.

• **Dalam XML:** Menggunakan sebuah sintaks XML khusus yang juga berkaitan dengan ID sumber daya yang didefinisikan dalam class `R`, misalnya:

```
@string/hello
```

- `string` adalah tipe sumber daya dan `hello` adalah nama sumber daya. Anda bisa menggunakan sintaks ini dalam sumber daya XML di mana saja Anda ingin menyediakan sebuah nilai dalam sebuah sumber daya. Lihat Mengakses Sumber Daya dari XML.

Mengakses sumber daya dalam kode

Anda bisa menggunakan sumber daya dalam kode dengan menyalurkan ID sumber daya sebagai sebuah parameter metode. Misalnya, Anda bisa menyetel `ImageView` agar menggunakan sumber daya res/drawable/myimage.png dengan menggunakan `setImageResource()`:

```
ImageView imageView = (ImageView)
findViewById(R.id.myimageview);
imageView.setImageResource(R.drawable.myimage);
```

Anda juga bisa mengambil tiap sumber daya dengan menggunakan berbagai metode di `Resources`, di mana Anda bisa mendapatkan instance dengan `getResources()`.

Sintaks

Inilah sintaks untuk mengacu sumber daya dalam kode:

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

- *<package_name>* adalah nama paket yang di dalamnya terdapat sumber daya (tidak dibutuhkan bila mereferensikan sumber daya dari paket Anda sendiri).
- *<resource_type>* adalah subclass `R` untuk tipe sumber daya.
- *<resource_name>* bisa berupa nama file sumber daya tanpa ekstensi atau nilai atribut `android:name` dalam elemen XML (untuk nilai sederhana).

Lihat Tipe Sumber Daya untuk informasi selengkapnya tentang masing-masing tipe sumber daya dan cara mengacunya.

e

r

4. Komponen Sumber Daya Tambahan

Mengelola project dan sumber

Berikut adalah beberapa konfigurasi untuk mengelola modul project beserta sumbernya. Untuk mempelajari cara membuat serta mengelola project dan modul lebih lanjut, baca Ringkasan Project.

Mengubah konfigurasi kumpulan sumber default

Anda bisa menggunakan blok `sourceSets` di file `build.gradle` tingkat modul untuk mengubah di mana Gradle mencari untuk mengumpulkan file untuk setiap komponen set sumber.

```
    android {
       ...
       sourceSets {
         // Encapsulates configurations for the main source set.
         main {
           // Changes the directory for Java sources. The default
directory is
           // 'src/main/java'.
           java.srcDirs = ['other/java']

           // When you list multiple directories, Gradle uses all
of them to collect
           // sources. You should avoid specifying a directory
which is a parent to one
           // or more other directories you specify.
           res.srcDirs = ['other/res1', 'other/res2']

           // For each source set, you can specify only one
Android manifest.
           // The following points Gradle to a different manifest
for this source set.
           manifest.srcFile 'other/AndroidManifest.xml'
           ...
         }

         // Create additional blocks to configure other source
sets.
         androidTest {
```

e

r

```
            // If all the files for a source set are located under
a single root
            // directory, you can specify that directory using the
setRoot property.
            // When gathering sources for the source set, Gradle
looks only in locations
            // relative to the root directory you specify. For
example, after applying
            // the configuration below for the androidTest source
set, Gradle looks for
            // Java sources only in the src/tests/java/ directory.
            setRoot 'src/tests'
            ...
        }
    }
}
...
```

Mengonfigurasi properti lingkup project

Untuk project yang mencakup beberapa modul, sebaiknya tentukan properti di level project dan bagikan ke seluruh modul. Anda bisa melakukannya dengan menambahkan properti tambahan ke blok `ext` dalam file `build.gradle` tingkat atas.

```
buildscript {...}
allprojects {...}

// This block encapsulates custom properties and makes them
available to all
// modules in the project.
ext {
    // The following are only a few examples of the types of
properties you can define.
    compileSdkVersion = 28
    // You can also use this to specify versions for
dependencies. Having consistent
    // versions between modules can avoid behavior
conflicts.
    supportLibVersion = "28.0.0"
    ...
}
...
```

Untuk mengakses properti ini dari modul dalam project yang sama, gunakan sintaks berikut pada file `build.gradle` tingkat modul.

e

r

```
    android {
      // Use the following syntax to access properties you
define at the project level:
      // rootProject.ext.property_name
      compileSdkVersion rootProject.ext.compileSdkVersion
      ...
    }
    ...
    dependencies {
        implementation "com.android.support:appcompat-
v7:${rootProject.ext.supportLibVersion}"
        ...
    }
```

Mengelola library dan dependensi

Gradle menyediakan mekanisme yang kuat untuk mengelola dependensi, baik library jarak jauh maupun modul library lokal.

Menargetkan build tertentu dengan konfigurasi dependensi

Jika Anda hanya menginginkan dependensi untuk kumpulan sumber varian build tertentu atau kumpulan sumber pengujian, gunakan huruf kapital untuk nama konfigurasi dependensi dan berikan awalan dengan nama varian build atau kumpulan sumber pengujian.

5. Komponen Perizinan

Meminta Izin Aplikasi

Setiap aplikasi Android berjalan dalam sandbox dengan akses terbatas. Jika harus menggunakan resource atau informasi di luar sandbox miliknya, aplikasi harus meminta *izin* yang sesuai. Untuk mendeklarasikan bahwa aplikasi Anda memerlukan izin, cantumkan izin dalam manifes aplikasi lalu minta pengguna untuk menyetujui setiap izin saat aplikasi diluncurkan (di Android 6.0 dan yang lebih baru).

Halaman ini menjelaskan cara menggunakan Android Support Library untuk memeriksa dan meminta izin. Mulai Android 6.0 (API level 23), framework Android menyediakan metode yang mirip, tetapi dengan support library, penyediaan kompatibilitas dengan versi Android lama menjadi lebih mudah.

e

r

Menambahkan izin ke manifes

Pada semua versi Android, untuk mendeklarasikan bahwa aplikasi Anda memerlukan izin, masukkan elemen `<uses-permission>` dalam manifes aplikasi Anda, sebagai turunan dari elemen `<manifest>` level teratas. Misalnya, aplikasi yang perlu mengakses internet akan memiliki baris berikut dalam manifes:

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

        <uses-permission
android:name="android.permission.INTERNET"/>
        <!-- other permissions go here -->

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

Perilaku sistem setelah Anda mendeklarasikan izin bergantung pada seberapa sensitif izin tersebut. Beberapa izin dianggap "normal" sehingga sistem segera memberikannya setelah aplikasi diinstal. Izin lainnya dianggap "berbahaya" sehingga pengguna harus mengizinkan akses aplikasi secara eksplisit. Untuk informasi selengkapnya tentang berbagai jenis izin, lihat Tingkat perlindungan.

Memeriksa izin

Jika aplikasi membutuhkan izin berbahaya, Anda harus memeriksa apakah Anda memiliki izin tersebut setiap kali melakukan operasi yang membutuhkannya. Mulai dari Android 6.0 (API level 23), pengguna bisa mencabut izin dari aplikasi apa pun kapan saja, bahkan meskipun aplikasi menargetkan level API yang lebih rendah. Jadi, meskipun aplikasi menggunakan kamera kemarin, aplikasi tidak bisa berasumsi izin tersebut masih dimilikinya hari ini.

Untuk memeriksa apakah Anda memiliki izin, panggil metode `ContextCompat.checkSelfPermission()`. Sebagai contoh, cuplikan berikut menunjukkan cara memeriksa apakah aktivitas memiliki izin untuk menulis ke kalender:

```
  if (ContextCompat.checkSelfPermission(thisActivity,
Manifest.permission.WRITE_CALENDAR)
            != PackageManager.PERMISSION_GRANTED) {
        // Permission is not granted
    }
```

e

r

Jika aplikasi tersebut memiliki izin, metode akan menampilkan `PERMISSION_GRANTED`, dan aplikasi dapat melanjutkan operasinya. Jika aplikasi tidak memiliki izin, metode akan menampilkan `PERMISSION_DENIED`, dan aplikasi harus meminta izin kepada pengguna secara eksplisit.

Meminta izin

Saat aplikasi menerima `PERMISSION_DENIED` dari `checkSelfPermission()`, Anda harus meminta izin tersebut dari pengguna. Android menyediakan beberapa metode yang dapat Anda gunakan untuk meminta izin, misalnya `requestPermissions()`, seperti yang ditunjukkan dalam cuplikan kode di bawah. Memanggil metode ini akan memunculkan dialog Android standar yang tidak dapat disesuaikan.

Tampilan yang akan dilihat oleh pengguna bergantung pada versi perangkat Android serta versi target aplikasi Anda, seperti yang dijelaskan dalam Ringkasan Izin.

Menjelaskan mengapa aplikasi membutuhkan izin

Dalam situasi tertentu, bantulah pengguna untuk memahami mengapa aplikasi Anda memerlukan izin. Misalnya, jika pengguna meluncurkan aplikasi fotografi, mereka mungkin tidak akan terkejut jika aplikasi meminta izin untuk menggunakan kamera, tetapi mereka mungkin tidak paham mengapa aplikasi juga menginginkan akses ke lokasi atau kontak pengguna. Sebelum aplikasi meminta izin, Anda harus mempertimbangkan untuk memberikan penjelasan kepada pengguna. Perlu diingat bahwa Anda tidak boleh membingungkan pengguna dengan penjelasan Anda. Jika penjelasan terlalu panjang, pengguna mungkin merasa aplikasi tersebut sulit digunakan dan akan menghapusnya.

Salah satu pendekatan yang bisa Anda gunakan adalah memberikan penjelasan hanya jika pengguna menolak permintaan izin. Android menyediakan metode utilitas, `shouldShowRequestPermissionRationale()`, yang menampilkan `true` jika sebelumnya pengguna telah menolak permintaan itu, dan menampilkan `false` jika pengguna menolak izin dan memilih opsi **Jangan tanya lagi** dalam dialog permintaan izin, atau jika kebijakan perangkat melarang izin tersebut.

Jika pengguna tetap mencoba menggunakan fungsi yang memerlukan izin, tetapi terus menolak permintaan izinnya, kemungkinan mereka tidak memahami mengapa aplikasi memerlukan izin untuk menyediakan fungsionalitas tersebut. Dalam situasi seperti ini, memberikan penjelasan akan menjadi solusi yang tepat.

Saran lebih lanjut tentang cara membuat pengalaman pengguna yang baik saat meminta izin dapat dilihat di Praktik Terbaik Izin Aplikasi.

Meminta untuk menjadi pengendali default, jika diperlukan

Beberapa aplikasi bergantung pada akses ke informasi sensitif pengguna yang terkait dengan log panggilan dan pesan SMS. Jika ingin meminta izin khusus untuk log panggilan dan pesan SMS

e

r

serta memublikasikan aplikasi ke Play Store, Anda harus meminta pengguna menetapkan aplikasi Anda sebagai *pengendali default* untuk fungsi sistem inti sebelum meminta izin waktu proses ini.

Untuk informasi selengkapnya tentang pengendali default, termasuk cara menampilkan permintaan pengendali default kepada pengguna, lihat panduan tentang izin yang hanya digunakan dalam pengendali default.

Meminta izin yang diperlukan

Jika belum memiliki izin yang diperlukan, aplikasi Anda harus memanggil salah satu metode `requestPermissions()` untuk meminta izin yang tepat. Aplikasi Anda meneruskan izin yang diperlukan dan *kode permintaan* integer yang Anda tentukan untuk mengidentifikasi permintaan izin ini. Metode ini berfungsi secara asinkron. Metode langsung menampilkan hasil, dan setelah pengguna merespons permintaan, sistem akan memanggil metode callback aplikasi beserta hasilnya, yang akan meneruskan kode permintaan yang sama dengan yang diteruskan oleh aplikasi ke `requestPermissions()`.

Kode berikut ini memeriksa apakah aplikasi memiliki izin untuk membaca kontak pengguna. Jika tidak, kode akan memeriksa apakah penjelasan tentang alasan diperlukannya izin perlu ditampilkan, dan jika penjelasan tidak diperlukan, izin akan diminta:

```
// Here, thisActivity is the current activity
    if (ContextCompat.checkSelfPermission(thisActivity,
            Manifest.permission.READ_CONTACTS)
            != PackageManager.PERMISSION_GRANTED) {

        // Permission is not granted
        // Should we show an explanation?
        if
(ActivityCompat.shouldShowRequestPermissionRationale(thisActivit
y,
                Manifest.permission.READ_CONTACTS)) {
            // Show an explanation to the user *asynchronously*
-- don't block
            // this thread waiting for the user's response!
After the user
            // sees the explanation, try again to request the
permission.
        } else {
            // No explanation needed; request the permission
            ActivityCompat.requestPermissions(thisActivity,
                    new
String[]{Manifest.permission.READ_CONTACTS},
                    MY_PERMISSIONS_REQUEST_READ_CONTACTS);

            // MY_PERMISSIONS_REQUEST_READ_CONTACTS is an
            // app-defined int constant. The callback method
```

e

r

```
gets the
            // result of the request.
        }
    } else {
        // Permission has already been granted
    }
```

Permintaan yang ditampilkan oleh sistem menjelaskan grup izin yang perlu diakses oleh aplikasi Anda, bukan izin khusus. Menangani respons permintaan izin

Ketika pengguna merespons permintaan izin aplikasi, sistem akan memanggil metode `onRequestPermissionsResult()` aplikasi Anda, yang meneruskan respons pengguna kepadanya. Aplikasi Anda harus mengganti metode tersebut untuk mengetahui apakah izin telah diberikan. Callback dikirimi kode permintaan yang sama dengan yang Anda teruskan ke `requestPermissions()`. Misalnya, jika sebuah aplikasi meminta akses `READ_CONTACTS`, aplikasi tersebut mungkin memiliki metode callback sebagai berikut:

```
@Override
    public void onRequestPermissionsResult(int requestCode,
            String[] permissions, int[] grantResults) {
        switch (requestCode) {
            case MY_PERMISSIONS_REQUEST_READ_CONTACTS: {
                // If request is cancelled, the result arrays
are empty.
                if (grantResults.length > 0
                    && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    // permission was granted, yay! Do the
                    // contacts-related task you need to do.
                } else {
                    // permission denied, boo! Disable the
                    // functionality that depends on this
permission.
                }
                return;
            }

            // other 'case' lines to check for other
            // permissions this app might request.
        }
    }
```

Kotak dialog yang ditampilkan oleh sistem menjelaskan grup izin yang perlu diakses aplikasi Anda; izin khususnya tidak dicantumkan. Misalnya, jika Anda meminta izin `READ_CONTACTS`, kotak dialog sistem hanya akan mengatakan bahwa aplikasi Anda memerlukan akses ke kontak perangkat. Pengguna hanya perlu memberikan izin satu kali

e

r

untuk setiap grup izin. Jika aplikasi Anda meminta izin lain dalam grup tersebut (yang tercantum dalam manifes aplikasi), sistem akan otomatis memberikannya. Ketika Anda meminta izin itu, sistem akan memanggil metode callback `onRequestPermissionsResult()` dan meneruskan `PERMISSION_GRANTED` dengan cara yang sama seperti jika pengguna telah mengizinkan permintaan secara eksplisit melalui kotak dialog sistem.

Misalnya, anggaplah Anda mencantumkan `READ_CONTACTS` dan `WRITE_CONTACTS` dalam manifes aplikasi Anda. Jika Anda meminta `READ_CONTACTS` dan pengguna memberikan izinnya, kemudian Anda meminta `WRITE_CONTACTS`, sistem akan langsung memberikan izin tanpa berinteraksi dengan pengguna.

Jika pengguna menolak permintaan izin, aplikasi Anda harus melakukan tindakan yang tepat. Misalnya, aplikasi mungkin menampilkan dialog yang menjelaskan mengapa tindakan yang diminta pengguna tidak bisa dilakukan karena membutuhkan izin tersebut.

Ketika sistem meminta pengguna untuk memberikan izin, pengguna memiliki opsi untuk memberi tahu sistem agar tidak meminta izin itu lagi. Dalam kasus ini, setiap kali aplikasi menggunakan `requestPermissions()` untuk meminta izin itu lagi, sistem akan langsung menolak permintaan. Sistem memanggil metode callback `onRequestPermissionsResult()` dan meneruskan `PERMISSION_DENIED`, dengan cara yang sama seperti ketika pengguna menolak permintaan Anda secara eksplisit. Metode ini juga akan menampilkan `false` jika kebijakan perangkat melarang aplikasi memiliki izin tersebut. Ini berarti bahwa ketika Anda memanggil `requestPermissions()`, Anda tidak dapat berasumsi bahwa telah terjadi interaksi langsung dengan pengguna.

Untuk memberikan pengalaman pengguna terbaik saat meminta izin aplikasi, lihat juga Praktik Terbaik Izin Aplikasi.

Mendeklarasikan izin berdasarkan level API

Untuk mendeklarasikan izin hanya pada perangkat yang mendukung izin waktu proses, yaitu perangkat yang menjalankan Android 6.0 (API level 23) atau yang lebih tinggi, sertakan tag `uses-permission-sdk-23`, bukan tag `uses-permission`.

Saat menggunakan salah satu tag ini, Anda dapat menyetel atribut `maxSdkVersion` untuk menetapkan bahwa, pada perangkat yang menjalankan versi lebih tinggi, izin tertentu tidak diperlukan.

e