



BINAR
ACADEMY

Android Permissions

Silver - Chapter 3 - Topic 1

**Selamat datang di Chapter 3 Topic 1 *online course*
Android Developer dari Binar Academy!**





Selamat Datang di Chapter 3 ✨

Pada Chapter 2 kita udah banyak belajar nih tentang tools Android Studio dan Git.

Di chapter ini, kita akan perdalam pemahaman tentang fitur-fitur yang perlu kamu ketahui dalam pembuatan aplikasi Android.

Pertama bin utama, kita akan bahas apa yang dimaksud dengan **Android Permission**. Yuk, lanjut!



Detailnya, kita bakal bahas hal-hal berikut ini:

- Konsep Security di Android
- Konsep Permission di Android
- Normal Permission
- Dangerous Permission





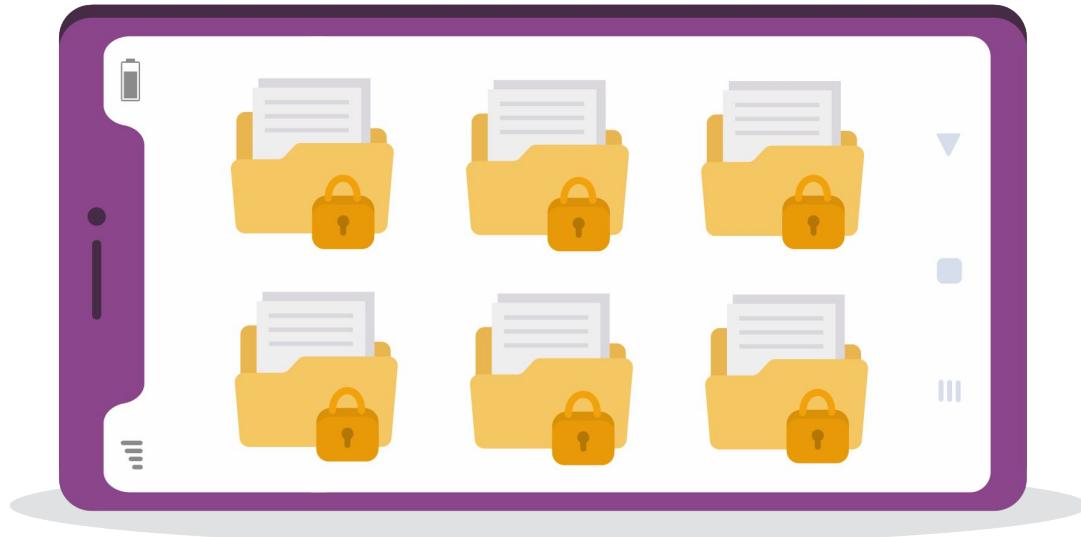
Layaknya satpam yang menjaga rumah kita, Android juga punya sistem **Permissions** dan **Security**.

Namun sebelum ke sana, kita perlu paham dulu nih dengan Konsep **Penjagaan Keamanan (Security)** pada Android a.k.a rumah kita!



Setiap rumah punya banyak ruangan yang memiliki pintu dan knop kunci berbeda untuk menjaga privasi. Ada pintu kamar mandi, pintu kamar tidur, dan sebagainya.

Nah, Android juga sama. Sistem pada Android menginstall setiap aplikasi Android dengan **ID User** dan **ID Grup** yang berbeda-beda.





Kita ambil contoh kamar mandi. Setiap kamar punya satu pintu dan satu kunci untuk melindungi privasi kita supaya orang nggak sembarangan masuk. Pada Android, setiap file aplikasi juga **bersifat pribadi** untuk setiap penggunanya.

Misalnya nih, Data Aplikasi WhatsApp **cuma bisa** diakses oleh aplikasi WhatsApp. Kamar kita cuma bisa diakses oleh kita.



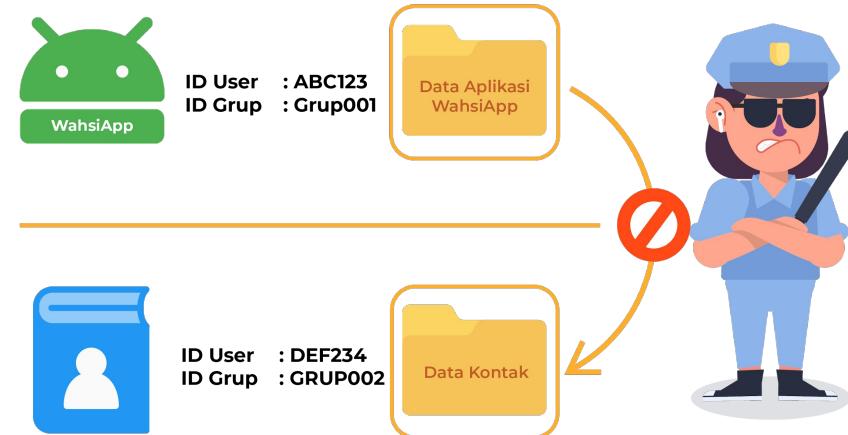
ID User : ABC123
ID Grup : Grup001





Oleh karena itu, melalui kernel Linux yang mendasarinya, setiap aplikasi Android **diisolasi** (dikunci) dari aplikasi yang sedang berjalan.

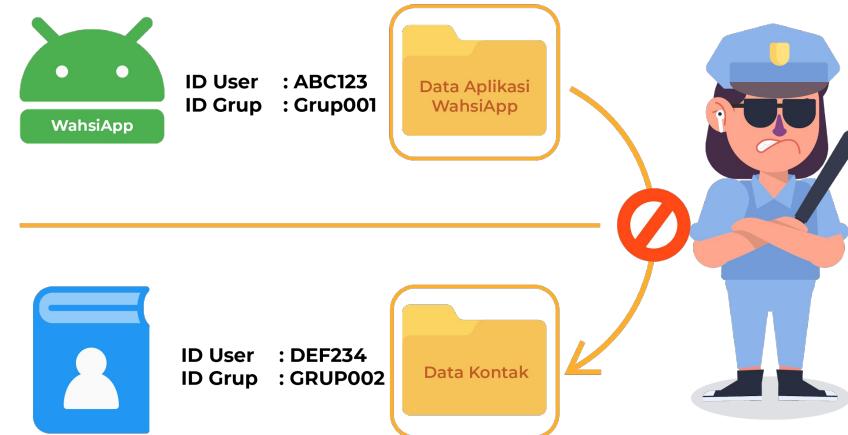
Oh, iya. Buat yang belum tahu, kernel adalah inti dari suatu OS / Inti dari Android. Sedangkan Linux adalah sistem operasi open source yang bisa kita gunakan.





Kenapa diisolasi? supaya aplikasi lain nggak bisa sembarangan akses informasi di aplikasi kita.

Coba bayangin deh kalau tetangga rumah sembarangan keluar-masuk kamar mandi kita. Nggak enak kan?

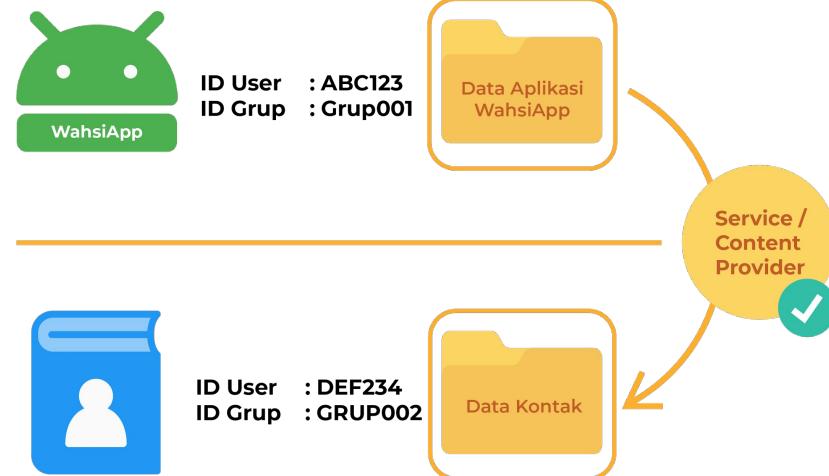




Dengan begitu, kalo aplikasi ingin membagikan data yang bersifat pribadi, aplikasi perlu memanfaatkan komponen yang menangani pembagian data.

Komponen tersebut misalnya **Service** dan **Content Provider**.

Salah satu aplikasi yang pake komponen Content Provider itu WhatsApp. Membuat kita bisa mengakses data nomer orang selama ada di kontak kita.



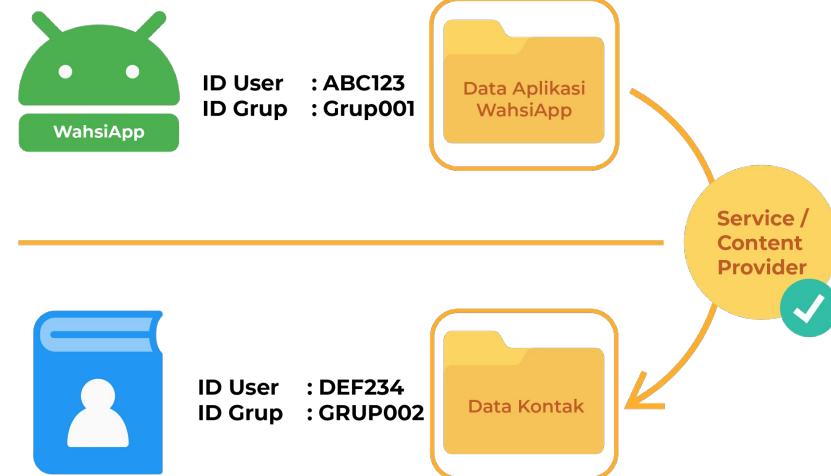


“Kok bisa? bukannya tadi kalo kayak gini (akses data orang) gak boleh sama kernel?”

Tentu bisa dong, asalkan pakai yang namanya Service ataupun Content Provider.

Ibaratnya, komponen ini tuh membuat perizinan akses data tetap sesuai dengan norma masyarakat setempat.

Nah pembahasan Service dan Content Provider, akan dibahas pada topik yang berbeda.





Konsep penjagaan keamanan (**Security**) Android ini kurang lebih mirip kaya keamanan komplek.

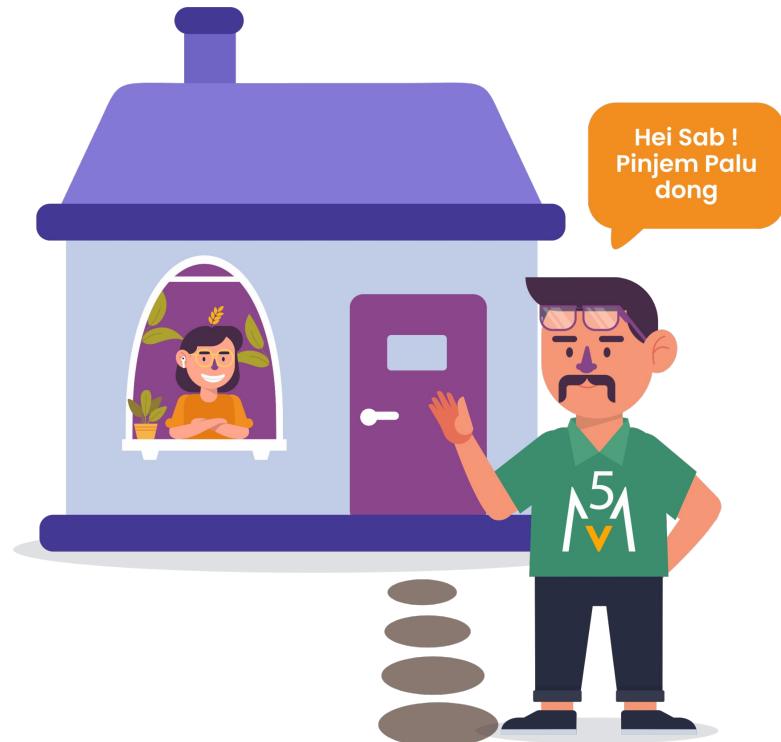
Sekarang, saatnya kita belajar konsep punten alias **Permission** pada Android!



Biar mudah, coba pakai analogi, ya!

Adam levine adalah tetangga rumah kamu yang pingin pinjem palu. Sebelum bisa malu, doi perlu izin dulu dong ke kamu as tuan rumah.

Kalo tiba-tiba tamu langsung nyelonong, bisa bisa dicurigai maling atau dianggap nggak sopan, dan lain sebagainya.





Nah, dari cerita tersebut, coba kondisinya kamu ganti dengan panduan seperti ini:

Tamu : Aplikasi yang ada pada ponsel.

Rumah : Ponsel dengan berbagai fitur yang sudah tersedia.

Tuan rumah : Sistem android yang dikendalikan oleh pengguna untuk mengatur perizinan aplikasi.





Excuse me...



Setiap aplikasi perlu meminta izin untuk menggunakan fitur yang ada pada smartphone.

Nah, izin yang diminta tentunya harus relevan dengan keperluan aplikasi.

Contohnya, supaya smartphone bisa pake koneksi internet, ya perlu memiliki izin dulu untuk menggunakan koneksi internet.

Pardon...

Permisi...

Punten ,slur...



Setiap permission yang dilakukan oleh suatu aplikasi, harus terdaftar pada **AndroidManifest.xml**.

Kalo suatu aplikasi memerlukan koneksi internet, pasti AndroidManifest.xml aplikasi tersebut bakal mencantumkan permission kayak gini:

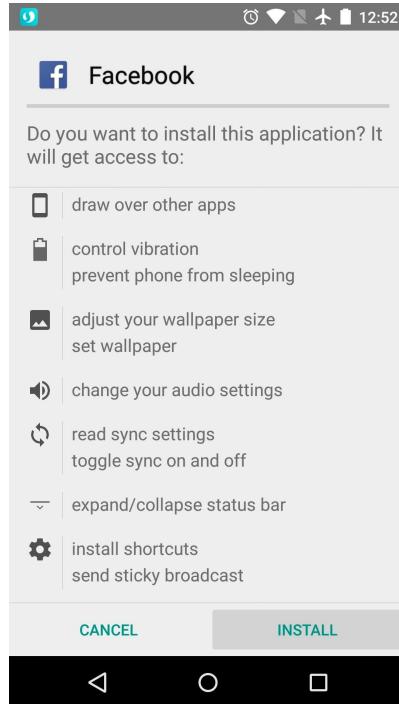
```
<uses-permission  
    android:name="android.permission.INTERNET" />
```





Nah, usut punya usut... konsep Permission telah berubah sejak API Level 23, atau bisa kita sebut ketika geng Android Marshmallow memimpin.

Mari kita bandingin before after-nya setelah ini.

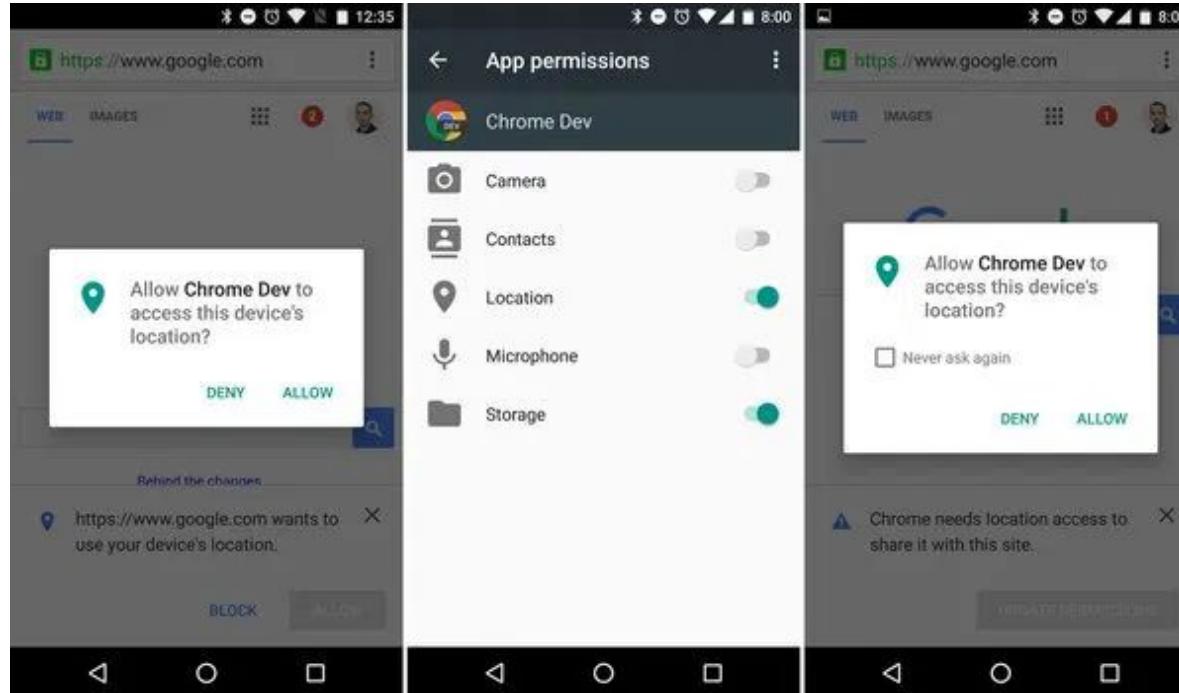


Sebelum geng Marshmallow exist (Lollipop, Kitkat, Jellybean, dst), permission bakal ditampilkan **ketika pengguna menginstall aplikasi**.

Pengguna perlu memutuskan dulu daftar permission yang ada diizinkan atau enggak.

Kalo pengguna menolak permission yang diperlukan, yaa berarti aplikasi tidak terinstall.

Ketika Install Aplikasi Di OS sebelum Marshmallow



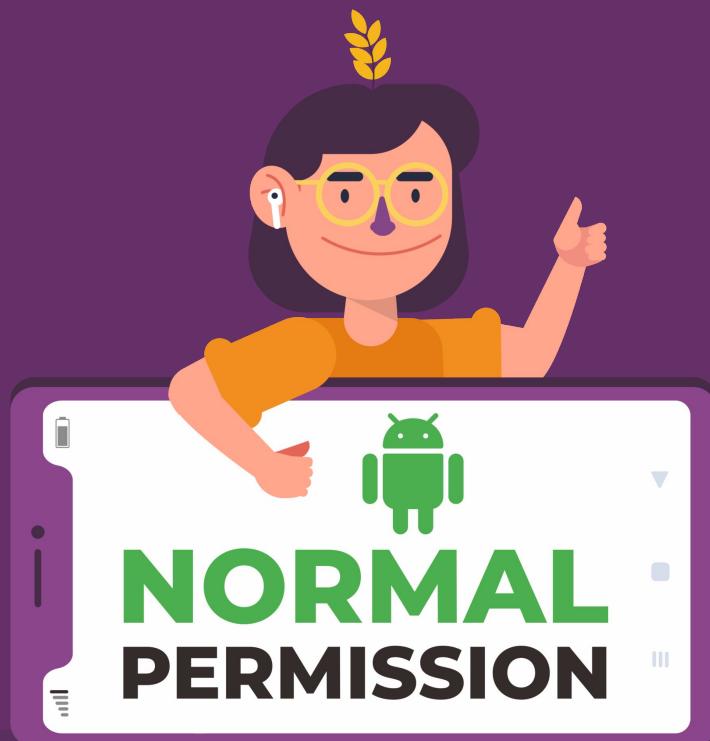
Versi geng Marshmallow hingga saat ini, pengguna akan ditanyai perizinan permission **ketika menggunakan aplikasi** (runtime) kaya gambar diatas itu.



Sehingga, kalo aplikasi yang dibuat itu menargetkan API level 23 (Marshmallow) atau yang lebih baru, kita harus menggunakan model permission yang baru juga.

Nah, model permission yang baru memiliki dua tipe tingkat permission yang perlu diperhatikan, yaitu **Normal Permission** dan **Dangerous Permission**.





Itu dia perbedaan punten yang dilakuin warga desa Android RT 01/RW 3.

Kalo kamu kepo sama dua tingkat permission tadi, tenang~

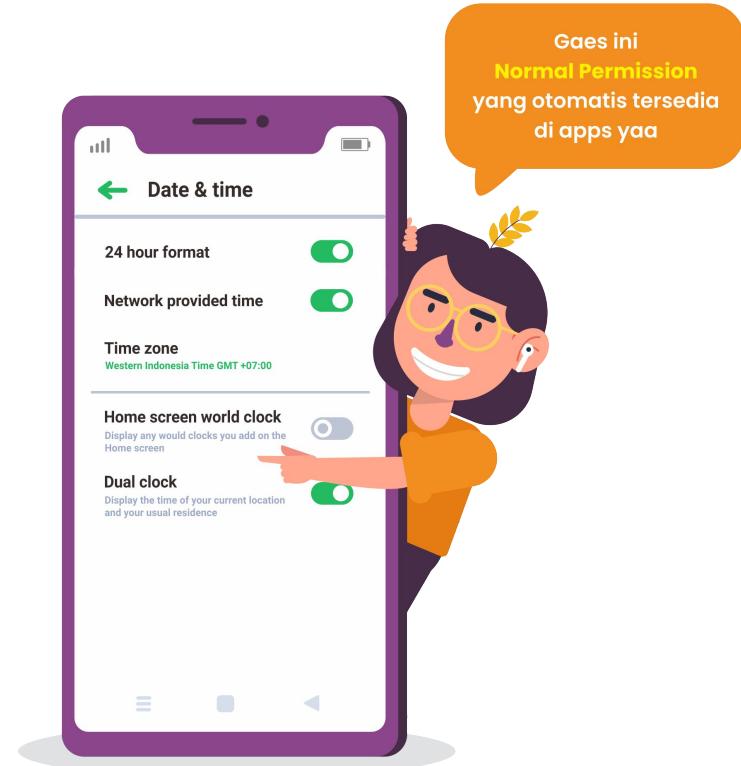
Sekarang kita bedah Permission pertama, yakni **Normal Permission**.



Normal Permission

Normal Permission adalah **permission yang dianggap tidak berbahaya bagi privasi pengguna atau pengoperasian aplikasi lain**. Misalnya, permission untuk mengatur zona waktu.

Normal permission ini diberikan secara otomatis ke aplikasi.





Dibawah ini beberapa contoh normal permission alias izin yang nggak berbahaya :

- ACCESS_NETWORK_STATE
- ACCESS_WIFI_STATE
- BLUETOOTH
- CHANGE_NETWORK_STATE
- SET_ALARM





Bagaimana Implementasi Normal Permission?

Implementasi yang paling sederhana dari penerapan Normal Permission adalah memuat gambar yang ada di internet menggunakan library image loader bernama **Glide**.

Kayak apa tuh? kita coba praktikan~



glide



Langkah 1, Tambahkan Permission

Ingat! Setiap permission yang akan digunakan oleh Aplikasi **harus bisa didefinisikan di file AndroidManifest.xml**, baik itu Normal Permission maupun Dangerous Permission.

Dikarenakan kita membutuhkan akses internet, kita wajib untuk mendefinisikan permission internet dulu dalam AndroidManifest.xml.

Untuk lebih jelasnya bisa dilihat di gambar yes~

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

    <uses-permission android:name="android.permission.INTERNET" />

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



Langkah 2, Tambahkan Dependency Library Glide

Kita akan menampilkan gambar dari internet dengan bantuan dari Library Glide.

Glide adalah library yang dibuat oleh pihak ketiga, sehingga kita perlu menambahkan dependency library di block dependencies dalam file build.gradle level module.

Cara menambahkan library ini, bisa lihat kode disamping yaa~



```
apply plugin: 'com.android.application'  
apply plugin: 'kotlin-android'  
apply plugin: 'kotlin-android-extensions'  
  
android {  
    //Other configuration here  
}  
  
dependencies {  
    //Other implementation here  
  
    //Glide - Image loader Library  
    implementation 'com.github.bumptech.glide:glide:4.11.0'  
    annotationProcessor 'com.github.bumptech.glide:compiler:4.11.0'  
}
```



Langkah 3, Siapkan Layout

Kita akan menyiapkan layout sederhana, yaitu layout anti ribet yang didesain hanya dengan **memuat ImageView sebagai wadah** tempat tampilnya image-nya.

Tampilan image ini bersumber dari internet dan memanfaatkan button sebagai pemicu untuk menjalankan pengambilan dan pemrosesan gambar oleh library Glide.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/ivBinar"
        android:layout_width="match_parent"
        android:layout_height="256dp" />

    <Button
        android:id="@+id/btnLoadImage"
        android:text="Load Image"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```



Langkah 4, Tambahkan Gambar~

Lalu kita tambahin nih code di method onCreate Activity supaya bisa menampilkan Image Logo Binar. Kamu bisa lakukan dengan url berikut :

<https://i.ibb.co/zJHYGBP/binarlogo.jpg>

Atau kita juga bisa menggunakan URL gambar lainnya.



```
btnLoadImage.setOnClickListener {
    Glide.with(this)
        .load("https://i.ibb.co/zJHYGBP/binarlogo.jpg")
        .circleCrop()
        .into(ivBinar)
}
```



Untuk melakukan loading gambar dengan Glide, ada beberapa method yang perlu kamu jalankan, yaitu :

- **with.** Method ini membutuhkan Context. Context Adalah instance dari class yang berhubungan langsung dengan tampilan / View.

Jika Glide dijalankan dari Activity, cukup tambahkan **this**.



```
btnLoadImage.setOnClickListener {
    Glide.with(this)
        .load("https://i.ibb.co/zJHYGBP/binarlogo.jpg")
        .circleCrop()
        .into(ivBinar)
}
```

The code above demonstrates how to use Glide to load an image from a URL and apply circular cropping. The code is enclosed in an `setOnClickListener` block. The `Glide.with(this)` line is annotated with a purple bracket and the label "Context". The `.load("https://i.ibb.co/zJHYGBP/binarlogo.jpg")` line is annotated with a purple bracket and the label "URL Gambar". The `.circleCrop()` and `.into(ivBinar)` lines are grouped together and annotated with a purple bracket and the label "Transformasi".

Context

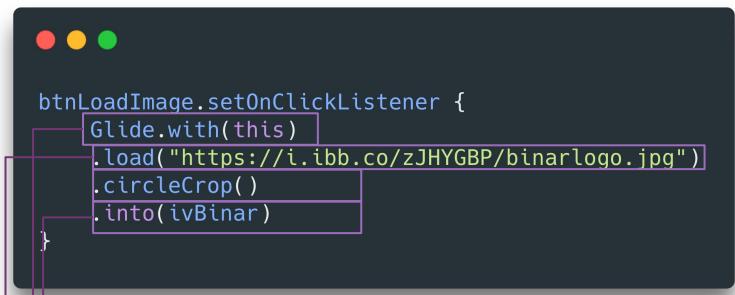
URL Gambar

Transformasi



- **load.** Diproses oleh method load. URL Gambar adalah link ke file Gambar yang ada di internet.

Biasanya diakhiri dengan ekstensi Gambar berupa jpg, png, bmp, dll. Selain loading dari URL Gambar, Glide juga bisa loading dari file bitmap, res, dll.



```
btnLoadImage.setOnClickListener {
    Glide.with(this)
        .load("https://i.ibb.co/zJHYGBP/binarlogo.jpg")
        .circleCrop()
        .into(ivBinar)
}
```

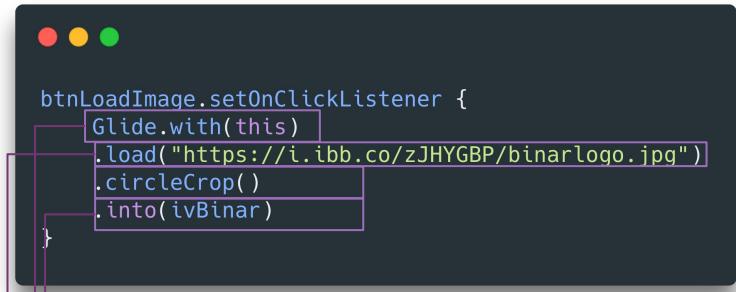
The code shows a click listener for a button. It uses the Glide library to load an image from a URL. The URL is "https://i.ibb.co/zJHYGBP/binarlogo.jpg". The image is processed with ".circleCrop()" and displayed in an ImageView named "ivBinar".

Annotations below the code:

- **Context**: Points to the line ".with(this)".
- **URL Gambar**: Points to the line ".load("https://i.ibb.co/zJHYGBP/binarlogo.jpg")".
- **Transformasi**: Points to the line ".circleCrop()".



- **circleCrop**. Method ini bersifat *additional*. Digunakan untuk mentransformasi gambar menjadi terpotong membentuk lingkaran.
- **into**. Method ini membutuhkan ImageView yang ada di file Layout.



```
btnLoadImage.setOnClickListener {
    Glide.with(this)
        .load("https://i.ibb.co/zJHYGBP/binarlogo.jpg")
        .circleCrop()
        .into(ivBinar)
}
```

The code demonstrates the use of the Glide library to load an image from a URL and apply a circular crop transformation. The code is enclosed in a click listener for a button. The Glide.with(this) call provides the context, the load call specifies the URL, circleCrop is the transformation, and into(ivBinar) identifies the ImageView where the result will be displayed.

→ Context

→ URL Gambar

→ Transformasi

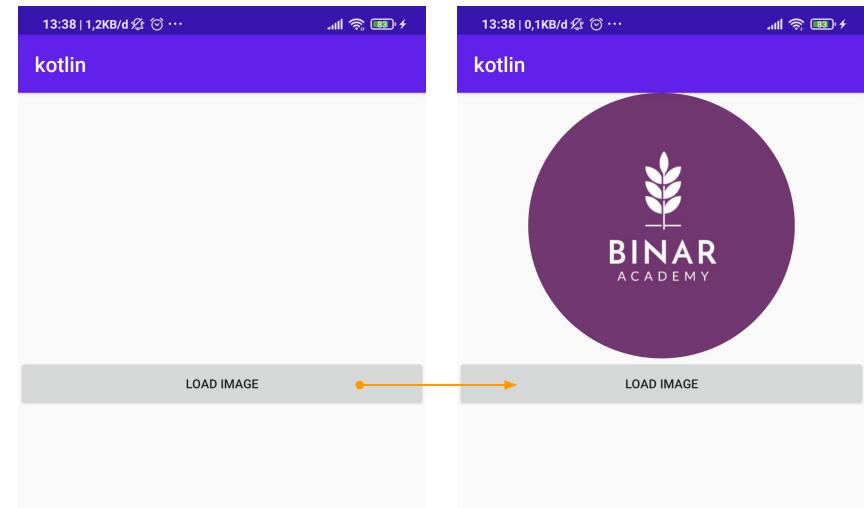


Langkah 5, Run!

Jika kita jalankan aplikasinya, lalu menekan tombol Load Image, maka akan tampil gambar Logo Binar dengan shape lingkaran.

Seperti yang udah kita terapin sebelumnya, gambarnya jadi lingkaran karena kita pake **circleCrop** pas loading gambar menggunakan Glide.

Gampang bukan?





Nah.. barusan itu adalah contoh implementasi dari **Normal Permission** yang tidak membutuhkan konfirmasi pengguna.

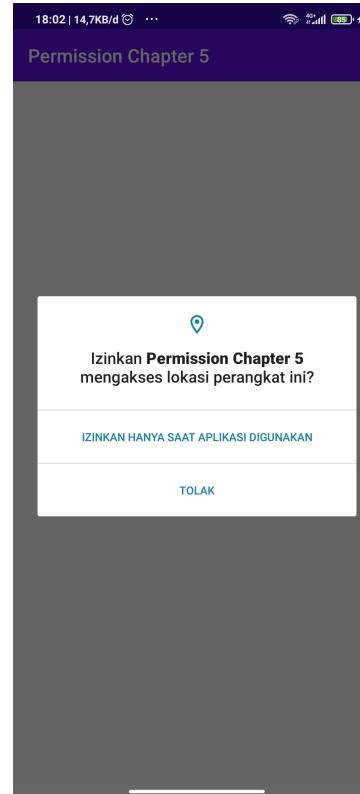
Selanjutnya apakah **Dangerous Permission** memerlukan konfirmasi pengguna? Kita cari tahu~



Dangerous Permission

Pas kamu install aplikasi, pernah liat gambar disamping? Pop-up message atau pesan tersebut merupakan tanda bahwa aplikasi tersebut ingin mengakses data yang bersifat pribadi.

Dangerous Permission mempengaruhi **informasi pribadi pengguna atau berpotensi mempengaruhi datanya atau pengoperasian operasi lain**. Misalnya, kemampuan aplikasi untuk membaca data kontak pengguna.

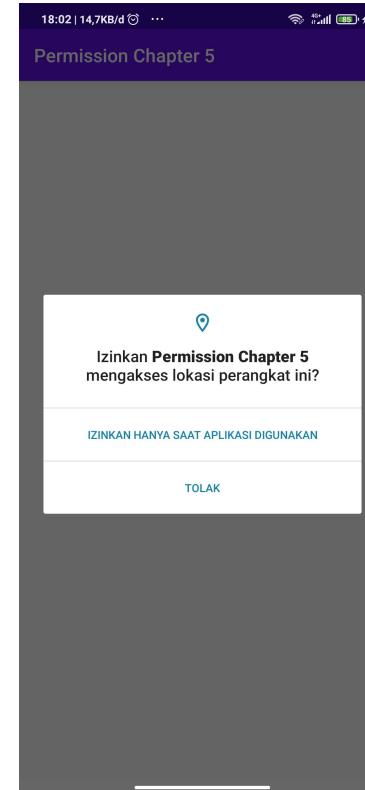




Dangerous permission perlu diinfoin ke pengguna saat menjalankan aplikasi atau runtime ke aplikasi.

Supaya kamu makin familiar, berikut beberapa contoh dangerous permission:

- READ_CALENDAR
- ACCESS_FINE_LOCATION
- RECORD_AUDIO
- USE_SIP
- SEND_SMS

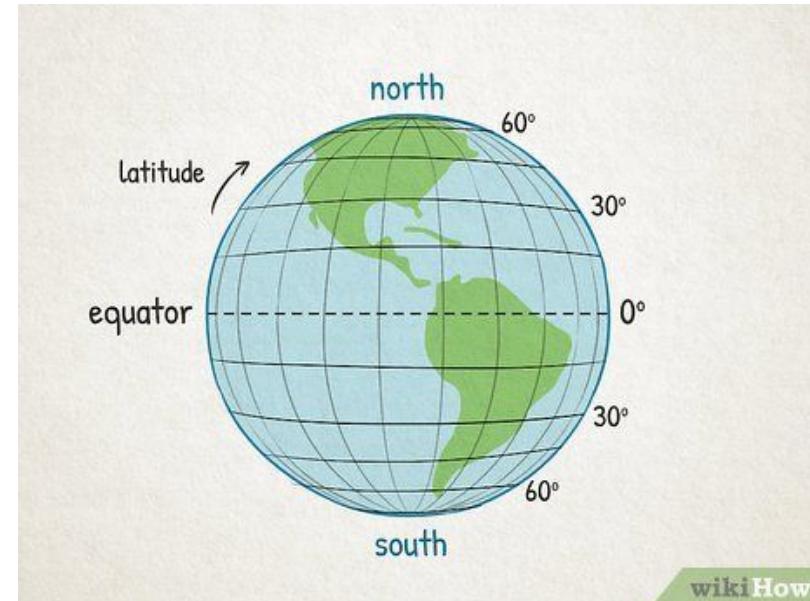




Bagaimana implementasi Dangerous Permission?

Supaya kita tetap menjadi pribadi yang sopan, kita perlu punten dulu nih sebelum mengakses data pribadi orang lain. Kali ini kita bakal coba mengaktifkan **permission lokasi [ACCESS_FINE_LOCATION]** untuk aplikasi kita.

Data yang akan didapatkan adalah berupa koordinat posisi user, dalam bentuk longitude dan latitude.





Check Permission

Siap 86! Code Check Permission dipanggil Activity untuk mengecek apakah akses yang menggunakan `ACCESS_FINE_LOCATION` sudah diizinkan atau belum, yang kemudian hasilnya akan ditampilkan dalam bentuk Toast.

Kalo sudah diizinkan, aplikasi bisa menggunakan fitur `ACCEES_FINE_LOCATION` untuk mendapatkan lokasi berupa Longitude dan Latitude.

Kalo belum diizinkan, aplikasi bisa meminta konfirmasi permission ke user.

```
val permissionCheck = checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION)

if (permissionCheck == PackageManager.PERMISSION_GRANTED){
    Toast.makeText(this,"Permission Location DIIZINKAN",Toast.LENGTH_LONG).show()
    getLongLat()
} else {
    Toast.makeText(this,"Permission Location DITOLAK",Toast.LENGTH_LONG).show()
    requestLocationPermission()
}
```



Mendapatkan Data yang Dibutuhkan

Kalau permission sudah diizinkan, maka aplikasi bisa mendapatkan data yang sebelumnya dibatasi oleh Android. Khusus pada latihan kita, data tersebut berupa lokasi si-pengguna, yaitu dalam bentuk **longitude dan latitude**.

Ini adalah code untuk mendapatkan data Lokasi dalam bentuk longitude dan latitude!

```
@SuppressLint("MissingPermission")
fun getLongLat(){
    val locationManager =
        applicationContext.getSystemService(Context.LOCATION_SERVICE) as LocationManager

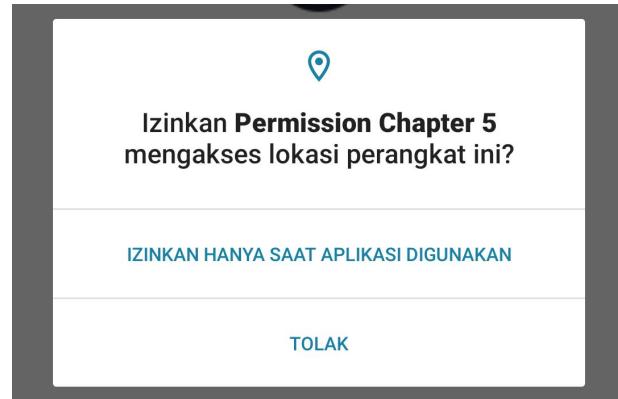
    val location: Location = locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER)
    Toast.makeText(this,"Lat: ${location.latitude} Long : ${location.longitude}",Toast.LENGTH_LONG).show()
}
```



Meminta permission

Kalo belum dapet permission, gimana? Berarti kita perlu mendapatkan permission terlebih dahulu.

Karena permission ‘ACCESS_FINE_LOCATION’ adalah salah satu Dangerous Permission maka ia memerlukan konfirmasi user untuk mendapatkan izin.



```
fun requestLocationPermission(){
    requestPermissions(arrayOf(Manifest.permission.ACCESS_FINE_LOCATION), 201)
}
```



Nah, untuk meminta konfirmasi, kita bisa menggunakan method '**requestPermissions**' yang membutuhkan dua parameter, yaitu:

1. **Array Permissions**

Kita bisa mengajukan lebih dari satu permission dalam sekali konfirmasi. Permission yang banyak itu bisa diborong dalam bentuk Array.



```
fun requestLocationPermission(){
    requestPermissions(arrayOf(Manifest.permission.ACCESS_FINE_LOCATION), 201)
}
```



- 2.** **Request** **Code**
Kode untuk mengidentifikasi permintaan permissions. Kurang lebih, mirip kayak nomor pesanan ketika kita memesan makanan di restoran.



```
fun requestLocationPermission(){  
    requestPermissions(arrayOf(Manifest.permission.ACCESS_FINE_LOCATION), 201)  
}
```



Mendapatkan Keputusan User mengenai Permintaan Permission

Tadi kita udah ngajuin permintaan permission pada User. Lalu user jadi punya dua pilihan, yaitu Mengizinkan (Allow) atau Menolak (Deny).

Setelah user memilih salah satu pilihan tersebut, aplikasi kita dapat mengetahui pilihan apa yang dipilih oleh user melalui method **onRequestPermissionsResult**.

```
override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
    when (requestCode){
        201 ->{
            if(grantResults.size>0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){
                Toast.makeText(this,"ALLOW telah dipilih",Toast.LENGTH_LONG).show()
                getLongLat()
            }else{
                Toast.makeText(this,"DENY telah dipilih",Toast.LENGTH_LONG).show()
            }
        }else->{
            Toast.makeText(this,"Bukan Request yang dikirim",Toast.LENGTH_LONG).show()
        }
    }
}
```



Ada tiga parameter yang kita dapatkan pada method **onRequestPermissionsResult**, yaitu :

- **requestCode**

Adalah bilangan Int request code pengajuan permissions yang sebelumnya telah kita tentukan.

- **permissions**

Array of String daftar permission yang pernah dilakukan.

```
override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
    when (requestCode){
        201 ->{
            if(grantResults.size>0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){
                Toast.makeText(this,"ALLOW telah dipilih",Toast.LENGTH_LONG).show()
                getLongLat()
            }else{
                Toast.makeText(this,"DENY telah dipilih",Toast.LENGTH_LONG).show()
            }
        }else->{
            Toast.makeText(this,"Bukan Request yang dikirim",Toast.LENGTH_LONG).show()
        }
    }
}
```



- **grantResults**

Array yang memuat hasil keputusan yang dipilih oleh user. Urutan array di grantResults sejajar dengan array permissions.

Jadi, misal di permissions[0] adalah permissions untuk 'ACCESS_FINE_LOCATION', maka untuk mengetahui keputusan user terhadap permission tersebut bisa dilihat di grantResults[0].

```
override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
    when (requestCode){
        201 ->{
            if(grantResults.size>0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){
                Toast.makeText(this,"ALLOW telah dipilih",Toast.LENGTH_LONG).show()
                getLongLat()
            }else{
                Toast.makeText(this,"DENY telah dipilih",Toast.LENGTH_LONG).show()
            }
        }else->{
            Toast.makeText(this,"Bukan Request yang dikirim",Toast.LENGTH_LONG).show()
        }
    }
}
```

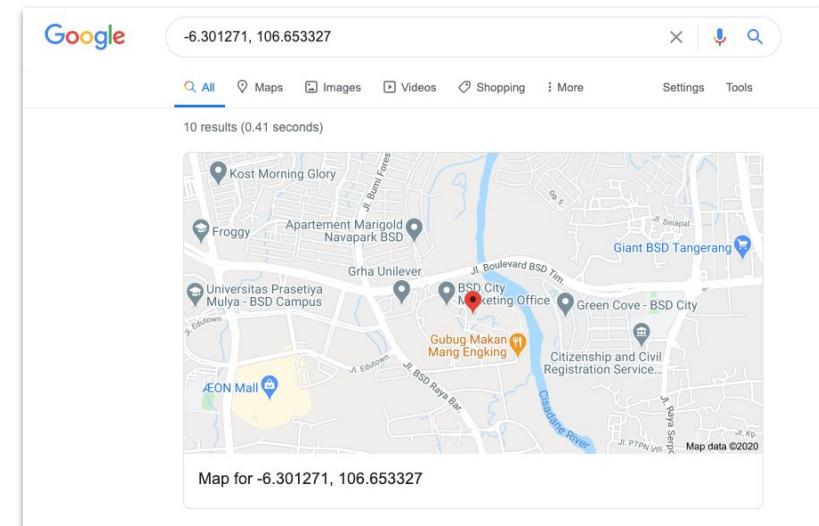


Selesai Deh!

Kita udah mempelajari cara menerapkan Normal Permission dengan contoh mendapatkan gambar dari internet.

Sedangkan untuk Dangerous Permission, kita menerapkan permission untuk mendapatkan Longitude dan Latitude lokasi yang cukup akurat.

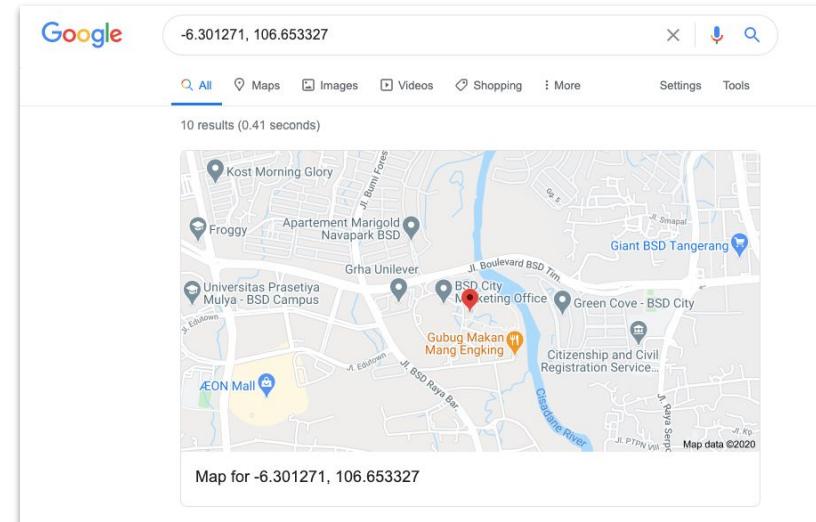
Setelah mendapatkan Longitude dan Latitude dari aplikasi, kita bisa melihat lokasi yang ditunjukkan dari Longitude dan Latitude dengan Google!





Selanjutnya kalo kamu ingin intip project lengkap tentang Permission-nya, bisa disini ya:

<https://github.com/sennohananto/PermissionChapter5>



Saatnya kita Quiz!





1. Suatu Aplikasi bisa mendapatkan akses terhadap data/resource aplikasi lain dengan bantuan dari...

- A. Intent Implicit/Explicit
- B. Content Provider/Service
- C. Gradle



1. Suatu Aplikasi bisa mendapatkan akses terhadap data/resource aplikasi lain dengan bantuan dari...

- A. Intent Implicit/Explicit
- B. Content Provider/Service
- C. Gradle

Content Provider memungkinkan aplikasi untuk berbagi data/resource yang dimilikinya dengan aplikasi lain. Hal yang serupa juga dapat dilakukan oleh Service.



2. Daftar permissions yang diperlukan aplikasi terdapat pada file apa?

- A. AndroidManifest.xml
- B. Build.gradle
- C. Res



2. Daftar permissions yang diperlukan aplikasi terdapat pada file apa?

- A. AndroidManifest.xml
- B. Build.gradle
- C. Res

Yap, benar! Pada file AndroidManifest.xml lah daftar permission yang diperlukan oleh aplikasi wajib dicantumkan seperti contoh berikut :

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

    <uses-permission      android:name="android.permission.SEND_SMS" />

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



3. Apakah perbedaan dari normal permission dan dangerous permission?

- A. Sama saja
- B. Normal permission: Tidak perlu izin saat digunakan
Dangerous permission: Memerlukan izin saat digunakan
- C. Normal permission: Permission yang disediakan oleh device
Dangerous permission: Permission yang tidak boleh diaktifkan



3. Apakah perbedaan dari normal permission dan dangerous permission?

- A. Sama saja
- B. Normal permission: Tidak perlu izin saat digunakan
Dangerous permission: Memerlukan izin saat digunakan
- C. Normal permission: Permission yang disediakan oleh device
Dangerous permission: Permission yang tidak boleh diaktifkan

Yap, normal permission merupakan sebuah izin yang akan langsung diberikan tanpa harus memerlukan konfirmasi lagi dari pengguna. Sedangkan dangerous permission memerlukan konfirmasi lagi dari pengguna ketika aplikasi membutuhkan permission tersebut.



4. Dengan method apa Aplikasi menerima Callback hasil dari permintaan request permissions ke pengguna?

- A. onRequestPermissionsResult()
- B. onPermissionsCheck()
- C. onResultPermissionsRequest()



4. Dengan method apa Aplikasi menerima Callback hasil dari permintaan request permissions ke pengguna?

- A. onRequestPermissionsResult()
- B. onPermissionsCheck()
- C. onResultPermissionsRequest()

Method `onRequestPermissionsResult()` akan menerima Callback hasil dari pengajuan permissions kepada pengguna.

Pada method ini, kita dapat menangani jika pengguna menerima permissions, menolak permissions, hingga penanganan permissions lainnya.



5. Code pada gambar berikut ini dapat kita temui pada?

```
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

<application>
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="Camera"
    android:roundIcon="@mipmap/ic_launcher_round"
    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>
```

- A. LocalProperties
- B. AndroidManifest
- C. Gradle



5. Code pada gambar berikut ini dapat kita temui pada?

```
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

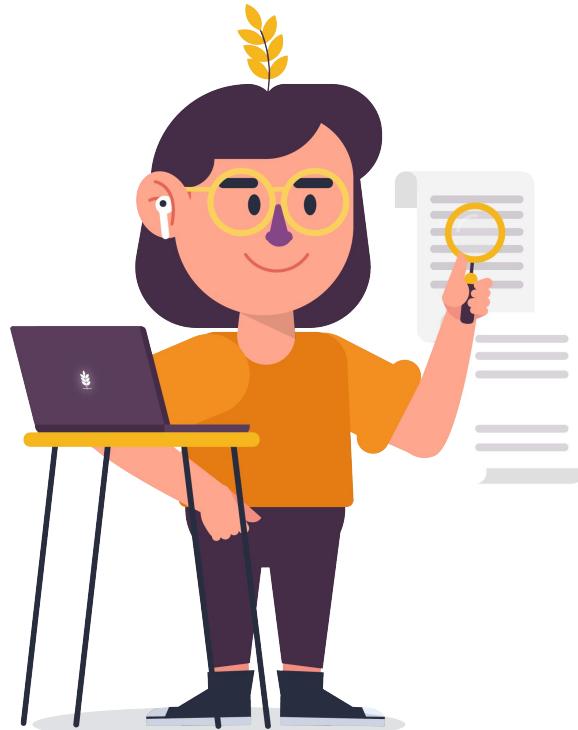
<application>
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="Camera"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

- A. LocalProperties
- B. AndroidManifest
- C. Gradle

Betul sekali! Gambar tersebut adalah code yang ada pada AndroidManifest! Pada file itu juga kita mendeklarasikan semua permission yang akan digunakan oleh aplikasi kita

Referensi dan bacaan lebih lanjut~

1. [Permissions on Android](#)
2. [The Android permissions model - Tutorial](#)





Nah, selesai sudah pembahasan kita di Chapter 3 Topic 1 ini.

Selanjutnya, kita bakal fitur Activity dan Activity Lifecycle.

Penasaran kayak gimana? Cus langsung ke topik selanjutnya~



Terima Kasih!



Next Topic

loading...